Sun Java System Application Server Enterprise Edition 8.2 Administration Guide

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

Part No: 819–4733–15
January 2010

Copyright 2009 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more U.S. patents or pending patent applications in the U.S. and in other countries.

U.S. Government Rights – Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

This distribution may include materials developed by third parties.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, the Solaris logo, the Java Coffee Cup logo, docs.sun.com, Java, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Products covered by and information contained in this publication are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical or biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED “AS IS” AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2009 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Tous droits réservés.

Sun Microsystems, Inc. détient les droits de propriété intellectuelle relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plusieurs brevets américains ou des applications de brevet en attente aux Etats-Unis et dans d'autres pays.

Cette distribution peut comprendre des composants développés par des tierces personnes.

Certaines composants de ce produit peuvent être dérivées du logiciel Berkeley BSD, licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays; elle est licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, le logo Solaris, le logo Java Coffee Cup, docs.sun.com, Java et Solaris sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui, en outre, se conforment aux licences écrites de Sun.

Les produits qui font l'objet de cette publication et les informations qu'il contient sont régis par la legislation américaine en matière de contrôle des exportations et peuvent être soumis au droit d'autres pays dans le domaine des exportations et importations. Les utilisations finales, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes chimiques ou biologiques ou pour le nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers des pays sous embargo des Etats-Unis, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exclusive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la legislation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement designés, sont rigoureusement interdites.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFACON.

100104@23031

Contents

Figures

Tables

Examples

Preface

The Sun Java System Application Server Enterprise Edition 8.2 *Administration Guide* provides system administration for the Application Server including configuration, monitoring, security, resource management, and web services management.

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

Application Server Documentation Set

The Application Server documentation set describes deployment planning and system installation. The Uniform Resource Locator (URL) for stand-alone Application Server documentation is `http://docs.sun.com/app/docs/coll/1310.4`. The URL for Sun Java Enterprise System (Java ES) Application Server documentation is `http://docs.sun.com/app/docs/coll/1310.3`. For an introduction to Application Server, refer to the books in the order in which they are listed in the following table.

TABLE P–1 Books in the Application Server Documentation Set

Book Title	Description
Release Notes	Late-breaking information about the software and the documentation. Includes a comprehensive, table-based summary of the supported hardware, operating system, Java Development Kit (JDK™), and database drivers.
Quick Start Guide	How to get started with the Application Server product.
Installation Guide	Installing the software and its components.
Deployment Planning Guide	Evaluating your system needs and enterprise to ensure that you deploy the Application Server in a manner that best suits your site. General issues and concerns that you must be aware of when deploying the server are also discussed.
Developer's Guide	Creating and implementing Java 2 Platform, Enterprise Edition (J2EE™ platform) applications intended to run on the Application Server that follow the open Java standards model for J2EE components and APIs. Includes information about developer tools, security, debugging, deployment, and creating lifecycle modules.

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

Book Title	Description
J2EE 1.4 Tutorial	Using J2EE 1.4 platform technologies and APIs to develop J2EE applications.
Administration Guide	Configuring, managing, and deploying Application Server subsystems and components from the Administration Console.
High Availability Administration Guide	Post-installation configuration and administration instructions for the high-availability database.
Administration Reference	Editing the Application Server configuration file, `domain.xml`.
Upgrade and Migration Guide	Migrating your applications to the new Application Server programming model, specifically from Application Server 6.x and 7. This guide also describes differences between adjacent product releases and configuration options that can result in incompatibility with the product specifications.
Performance Tuning Guide	Tuning the Application Server to improve performance.
Troubleshooting Guide	Solving Application Server problems.
Error Message Reference	Solving Application Server error messages.
Reference Manual	Utility commands available with the Application Server; written in man page style. Includes the `asadmin` command line interface.

Related Documentation

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

For other Sun Java System server documentation, go to the following:

- Message Queue documentation
- Directory Server documentation
- Web Server documentation

Additionally, the following resources might be useful:

- The J2EE 1.4 Specifications (`http://java.sun.com/j2ee/1.4/docs/index.html`)
- The J2EE 1.4 Tutorial (`http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`)
- The J2EE Blueprints (`http://java.sun.com/reference/blueprints/index.html`)

Default Paths and File Names

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

TABLE P–2 Default Paths and File Names

Placeholder	Description	Default Value

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

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

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

Typographic Conventions

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

TABLE P–3 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	A placeholder to be replaced with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized (note that some emphasized items appear bold online)	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file.

Symbol Conventions

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

TABLE P–4 Symbol Conventions

Symbol	Description	Example	Meaning
`[ ]`	Contains optional arguments and command options.	`ls [-l]`	The `-l` option is not required.
`{ | }`	Contains a set of choices for a required command option.	`-d {y|n}`	The `-d` option requires that you use either the `y` argument or the `n` argument.
`${ }`	Indicates a variable reference.	`${com.sun.javaRoot}`	References the value of the `com.sun.javaRoot` variable.
-	Joins simultaneous multiple keystrokes.	Control-A	Press the Control key while you press the A key.
+	Joins consecutive multiple keystrokes.	Ctrl+A+N	Press the Control key, release it, and then press the subsequent keys.

TABLE P–4 Symbol Conventions *(Continued)*

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

Documentation, Support, and Training

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

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

Searching Sun Product Documentation

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

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

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

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

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

Third-Party Web Site References

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

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

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. To share your comments, go to `http://docs.sun.com` and click Send Comments. In the online form, provide the full document title and part number. The part number is a 7-digit or 9-digit number that can be found on the book's title page or in the document's URL. For example, the part number of this book is 819-4733.

◆ ◆ ◆ CHAPTER 1

Getting Started

This chapter describes the Sun Java System Application Server administration. Application Serveradministration includes many tasks such as deploying applications, creating and configuring domains, server instances and resources, controlling (starting and stopping) domains and server instances, monitoring and managing performance, and diagnosing and troubleshooting problems. This chapter contains following sections:

About the Sun Java System Application Server

The Sun Java System Application Server provides a Java 2 Platform, Enterprise Edition (J2EE platform) 1.4 compatible platform for developing and delivering server side Java applications and Web services. Key features include scalable transaction management, container-managed persistence runtime, performant web services, clustering, high availability, security, and integration capabilities.

The Application Server is available in the following editions:

- The Platform edition is free and is intended for software development and department-level production environments.
- The Enterprise edition is designed for mission-critical services and large-scale production environments. It supports horizontal scalability and service continuity with a load balancer plug-in and cluster management. The Enterprise edition also supports reliable session state management with the high-availability database (HADB).

This section contains the following topics:

■ "Tools for Administration" on page 28

What is the Application Server?

The Application Server is a platform that supports services from Web publishing to enterprise-scale transaction processing, while enabling developers s to build applications based on JavaServer Pages (JSP), Java servlets, and Enterprise JavaBeans (EJB) technology.

The Application Server Platform Edition is *free* for development, production deployment, and redistribution. For more information on redistribution, please visit `http://www.sun.com/software/products/appsrvr/appsrvr_oem.xml`.

The Application Server Enterprise Edition provides advanced clustering and failover technologies. The Application Server infrastructure supports the deployment of many types of distributed applications and is an ideal foundation for building applications based on Service Oriented Architectures (SOA). SOA is a design methodology aimed at maximizing the reuse of application services. These features enable you to run scalable and highly available J2EE applications.

- **Scalability** - Scalability is achieved through clustering. A cluster is a group of application server instances that work together as one logical entity. Each Application Server instance in a cluster has the same configuration and the same applications deployed to it.

  Horizontal scaling is achieved by adding Application Server instances to a cluster, thereby increasing the capacity of the system. It is possible to add Application Server instances to a cluster without disrupting service. The HTTP, RMI/IIOP, and JMS load balancing systems distribute requests to healthy Application Server instances in the cluster.
- **High Availability** - Availability refers to the failover capability. If one server instance goes down, another server instance in a cluster takes over the sessions of the failed instance and continues to serve the clients in a seamless manner. Session information is stored in the high-availability database (HADB). HADB supports the persistence of HTTP sessions and stateful session beans.

Application Server Architecture

This section describes Figure 1–1, which shows the high-level architecture of the Application Server.

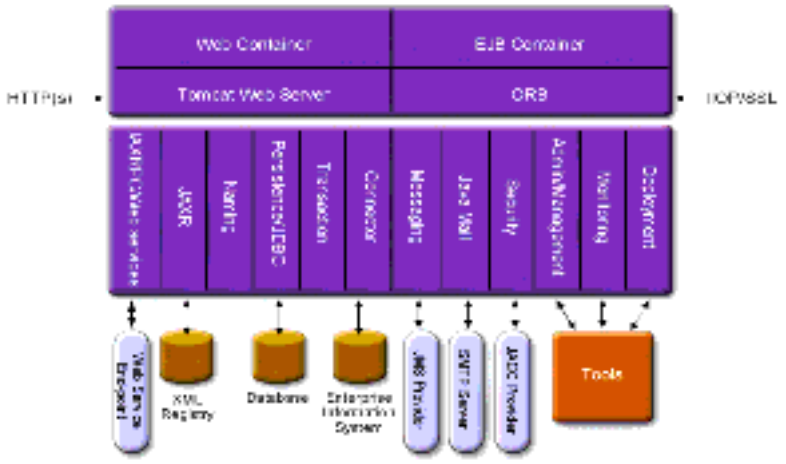

FIGURE 1–1 Application Server Architecture

- **Containers** - A container is a runtime environment that provides services such as security and transaction management to J2EE components. Figure 1–1 shows the two types of J2EE containers: Web and EJB. Web components, such as JSP pages and servlets, run within the Web container. Enterprise Java Beans run within the EJB container.
- **Client Access** - At runtime, browser clients access Web applications by communicating with the Web server via HTTP, the protocol used throughout the internet. The HTTPS protocol is for applications that require secure communication. Enterprise Java Bean clients communicate with the Object Request Broker (ORB) through the IIOP or IIOP/SSL (secure) protocols. The Application Server has separate listeners for the HTTP, HTTPS, IIOP, and IIOP/SSL protocols. Each listener has exclusive use of a specific port number.
- **Web Services** - On the J2EE platform, it is possible to deploy a Web application that provides a Web service implemented by Java API for XML-Based RPC (JAX-RPC). A J2EE application or component can also be a client to other Web services. Applications access XML registries through the Java API for XML Registries (JAXR).
- **Services for Applications** - The J2EE platform was designed so that the containers provide services for applications. Figure 1–1 shows the following services:
  - Naming - A naming and directory service binds objects to names. A J2EE application locates an object by looking up its JNDI name. JNDI stands for the Java Naming and Directory Interface API.
  - Security - The Java Authorization Contract for Containers (JACC) is a set of security contracts defined for the J2EE containers. Based on the client's identity, the containers restrict access to the container's resources and services.
- **Transaction management** - A transaction is an indivisible unit of work. For example, transferring funds between bank accounts is a transaction. A transaction management service ensures that a transaction either completes fully or is rolled back.

Access to External Systems

The J2EE platform enables applications to access systems that are outside of the application server. Applications connect to these systems through objects called resources. One of the responsibilities of an administrator is resource configuration. The J2EE platform enables access to external systems through the following APIs and components:

- JDBC - A database management system (DBMS) provides facilities for storing, organizing, and retrieving data. Most business applications store data in relational databases, which applications access via the JDBC API. The information in databases is often described as persistent because it is saved on disk and exists after the application ends. The Application Server bundle includes the Java DB database management system.
- Messaging - Messaging is a method of communication between software components or applications. A messaging client sends messages to, and receives messages from, any other client. Applications access the messaging provider through the Java Messaging Service (JMS) API. The Application Server includes a JMS provider.
- Connector - The J2EE Connector architecture enables integration between J2EE applications and existing Enterprise Information Systems (EIS). An application accesses an EIS through a portable J2EE component called a connector or resource adapter.
- JavaMail - Through the JavaMail API, applications connect to an SMTP server in order to send and receive email.
- Server Administration - The lower right-hand corner of Figure 1-1 shows administration interfaces of the Application Server. Administration tools communicate with the Application Server using these interfaces.

Tools for Administration

Various different tools and APIs are available to administer the Sun Java System Application Server:

- "Administration Console" on page 28
- " Command-line Interface (asadmin utility)" on page 29
- "JConsole" on page 29
- "Application Server Management Extension (AMX)" on page 30

Administration Console

The Administration Console is a browser-based tool that features an easy-to-navigate interface and online help. The administration server (also called the Domain Administration Server or DAS) must be running to use the Administration Console. You need the administration server hostname and port number to launch the Administration Console. The default administration server port number for the default administration server is 4849. You also need the administration username and password to login to the Administration Console. Consult the section for more detailed information.

To start the Administration Console, in a web browser type:

```
https://hostname:port
```

For example:

```
https://kindness.sun.com:4849
```

If the Administration Console is running on the machine on which the administration server is running, you can specify `localhost` for the host name.

On Windows, start the Application Server Administration Console from the Start menu.

Command-line Interface (asadmin utility)

The `asadmin` utility is a command-line interface for the Sun Java System Application Server. You can perform the same set of administrative tasks offered by the Administration Console. The `asadmin` utility can be invoked from a command prompt in the shell, or called from other scripts or programs. The `asadmin` utility is installed in *install-dir*`/bin` directory. The default Sun Java System Application Server installation root directory on Solaris is `/opt/SUNWappserver`.

To start the `asadmin` utility, go to the *install-dir*`/bin` directory and enter:

```
$ asadmin
```

To list the commands available within `asadmin`:

```
asadmin> help
```

It is also possible to issue an `asadmin` command at the shell's command prompt:

```
$ asadmin help
```

To view a command's syntax and examples, type `help` followed by the command name. For example:

```
asadmin> help create-jdbc-resource
```

The `asadmin help` information for a given command displays the UNIX man page of the command. These man pages are also available in HTML on the Web at *Sun Java System Application Server Enterprise Edition 8.2 Reference Manual*.

JConsole

In the Java 2, Platform Standard Edition 5.0, the Java Monitoring and Management Console (JConsole) was introduced. JConsole is used to monitor the Sun Java System Application Server. You can use either the JConsole remote tab, or the advanced tab to connect to the Application Server.

- Remote Tab: identify the username, password, administration server host, and JMS port number (8686 by default), and select Connect.
- Advanced Tab: identify the `JMXServiceURL` as service:`jmx:rmi:///jndi/rmi://host:jms-port/jmxrmi` and select Connect. The `JMXServerURL` is printed in the `server.log` file as well as output in the command window of the domain creation command.

Application Server Management Extension (AMX)

The **A**pplication Server **M**anagement e**X**tension is an API that exposes all of the Application Server configuration and monitoring JMX managed beans as easy-to-use client-side dynamic proxies implementing the AMX interfaces.

For more information on using the Application Server Management Extension, see Chapter 16, "Using the Java Management Extensions (JMX) API," in *Sun Java System Application Server Enterprise Edition 8.2 Developer's Guide* .

Application Server Commands and Concepts

The Sun Java System Application Serverconsists of one or more domains. A domain is an administrative boundary or context. Each domain has an administration server (also called Domain Administration Server or DAS) associated with it and consists of zero or more standalone instances and/or clusters. Each cluster has one or more homogeneous server instances. A server instance is a single Java Virtual Machine (JVM) that runs the Application Server on a single physical machine. Server instances (whether standalone or clustered) in a domain can run on different physical hosts.

This section contains following topics:

Domain

A domain is a group of instances that are administered together. However, an application server instance can belong to just one domain. In addition to administration boundary, domain provide a basic security structure whereby different administrators can administer specific groups (domains) of application server instances. By grouping the server instances into separate domains, different organizations and administrators can share a single Application Server

installation. Each domain has its own configuration, log files, and application deployment areas that are independent of other domains. If the configuration is changed for one domain, the configurations of other domains are not affected.

The Sun Java System Application Server installer creates the default administrative domain (named `domain1`). It also creates an associated domain administration server (named `server`). You must provide the administration server port number. The default administration server port is 4849. The installer also queries for the administration username and password. After installation, additional administration domains can be created.

Domain Administrative Server (DAS)

Each domain has its own Domain Administration Server (DAS) with a unique port number. The Administration Console communicates with a specific DAS to administer the associated domain. Each Administration Console session allows you to configure and manage the specific domain.

The Domain Administration Server (DAS), is a specially-designated application server instance that hosts the administrative applications. The DAS authenticates the administrator, accepts requests from administration tools, and communicates with server instances in the domain to carry out the requests. The DAS is sometimes referred to as the admin server or default server. It is referred to as the default server because it is the only server instance that gets created on Sun Java System Application Server installation and can be used for deployments. The DAS is simply a server instance with additional administration capabilities.

Each Administration Console session allows you to configure and manage a single domain. If you created multiple domains, you must start an additional Administration Console session to manage the other domains. When specifying the URL for the Administration Console, be sure to use the port number of the DAS associated with the domain to be administered.

Cluster

A cluster is a named collection of server instances sharing the same set of applications, resources, and configuration information. A server instance can belong to exactly one cluster. A cluster facilitates server instance load-balancing through distribution of a load across multiple machines. A cluster facilitates high availability through instance-level failover. From an administrative perspective, a cluster represents a virtualized entity in which operations on a cluster (e.g. deployment of an application) act on all instances that make up the cluster.

Node Agent

A lightweight agent (e.g. hosting a JMX runtime only) is required on each node in the domain to facilitate remote lifecycle management of instances. Its primary purpose is to start, stop, and create server instances as instructed by the DAS. The Node Agent also acts as a watchdog and

restarts failed processes. Like the DAS, the Node Agent should only be required for certain administrative operations and should not be expected to be highly available. However, the Node Agent is an “always on” component, and must be configured to be started by the native O/S node bootstrap (e.g. Solaris/Linux `inetd`, or as a Windows service). A Node Agent is not required for the DAS.

Server Instance

The server instance is a single J2EE compatible Java Virtual Machine hosting a J2EE 1.4 Application Server on a single node. Each server instance has a unique name in the domain. A clustered server instance is a member of a cluster and receives all of its application, resource, and configuration from its parent cluster; ensuring that all instances in the cluster are homogeneous. An unclustered server instance does not belong to a cluster and as such has an independent set of applications, resources, and configuration.

Application server instances form the basis of an application deployment. Each instance belongs to a single domain. Every server instance, other than the DAS, must contain a reference to a node agent name defining the machine on which the instance will reside.

If your topology includes remote server instances (server instances other than the DAS), create node agents to manage and facilitate remote server instances. It is the responsibility of the node agent to create, start, stop, and delete a server instance. Use the command line interface commands to set up node agents. Figure 1–2 shows an application server instance in detail.

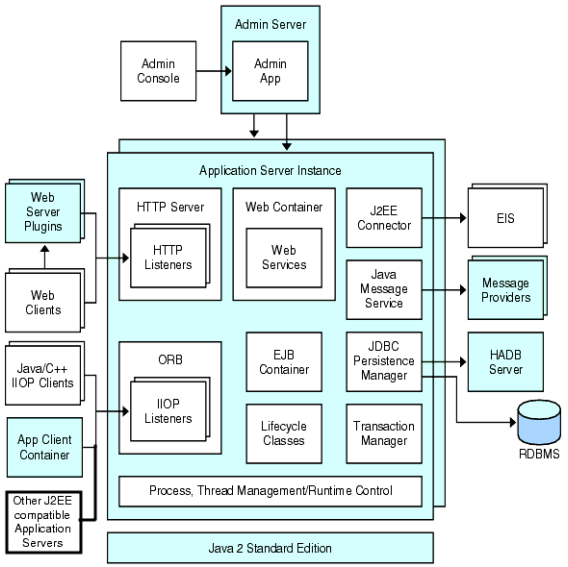

FIGURE 1–2 Application Server Instance

The Sun Java System Application Server creates one application server instance, called `server` at the time of installation. For many users, one application server instance meets their needs. However, depending upon your environment, you might want to create one or more additional application server instances. For example, in a development environment you can use different application server instances to test different Application Server configurations, or to compare and test different application deployments. Because you can easily add or delete an application server instance, you can use them to create temporary sandbox area for experimentation purposes.

In addition, for each application server instance, you can also create virtual servers. Within a single installed application server instance you can offer companies or individuals domain names, IP Addresses, and some administration capabilities. For the users, it is almost as if they have their own web server, without the hardware and basic server maintenance. These virtual servers do not span application server instances. For more information about virtual servers, see Chapter 12, “Configuring the HTTP Service.”

In operational deployments, for many purposes you can use virtual servers instead of multiple application server instances. However, if virtual servers do not meet your needs, you can also use multiple application server instances. On stopping, application server instance stops accepting new connections, then waits for all outstanding connections to complete. If your machine crashes or is taken offline, the server quits and any requests it was servicing may be lost.

Application Server Commands

Administration of the Application Server includes tasks such as creation, configuration, control and management of domains, clusters, node agents, and server instances. This section contains the following topics:

Creating a Domain

Domains are created using the `create-domain` command. The following example command creates a domain named `mydomain`. The administration server listens on port 1234 and the administrative user name is `hanan`. The command prompts for the administrative and master passwords.

```
$ asadmin create-domain --adminport 80 --adminuser hanan mydomain
```

To start the Administration Console for `mydomain` domain, in a browser, enter the following URL:

`http://`*hostname*`:80`

For the preceding `create-domain` example, the domain's log files, configuration files, and deployed applications now reside in the following directory:

domain-root-dir`/mydomain`

To create the domain's directory in another location, specify the `--domaindir` option. For the full syntax of the command, type `asadmin help create-domain`.

Deleting a Domain

Domains are deleted using the `asadmin delete-domain` command. Only the operating system user (or root) who can administer the domain can execute this command successfully. To delete a domain named `mydomain`, for example, type the following command:

```
$ asadmin delete-domain mydomain
```

Listing Domains

The domains created on a machine can be found using the `asadmin list-domains` command. To list the domains in the default *domain-root-dir* directory, type this command:

```
$ asadmin list-domains
```

To list domains that were created in other directories, specify the `--domaindir` option.

Starting the Domain

When starting a domain, the administration server and application server instance are started. Once the application server instance is started it runs constantly, listening for and accepting requests. Each domain must be started separately.

To start a domain, type the `asadmin start-domain` command and specify the domain name. For example, to start the default domain (`domain1`), type the following:

```
$ asadmin start-domain --user admin domain1
```

If there is only one domain, omit the domain name. For the full command syntax, type `asadmin help start-domain`. If the password data is omitted, you are prompted to supply it.

Starting the default domain on Windows

From the Windows Start Menu, select Programs -> Sun Microsystems -> Application Server -> Start Admin Server.

Stopping the Domain

Stopping a domain shuts down its administration server and application server instance. When stopping a domain, the server instance stops accepting new connections and then waits for all outstanding connections to complete. This process takes a few seconds because the server instance must complete its shutdown process. While the domain is stopped, the Administration Console or most `asadmin` commands cannot be used.

To stop a domain, type the `asadmin stop-domain` command and specify the domain name. For example, to stop the default domain (`domain1`), type the following:

```
$ asadmin stop-domain domain1
```

If there is only one domain, then the domain name is optional. For the full syntax, type `asadmin help stop-domain`.

Stopping the default domain on Windows

From the Start menu select Programs -> Sun Microsystems -> Application Server-> Stop Admin Server.

Restarting the Domain

Restarting the server is the same as restarting the domain. To restart the domain or server, stop and start the domain.

Creating a Cluster

A cluster is created using the `create-cluster` command. The following example creates a cluster named `mycluster`. The administration server host is `myhost`, the server port is `1234`, and the administrative username is `admin`. The command prompts for the administrative passwords.

```
$ asadmin create-cluster --host myhost --port 1234 --user admin mycluster
```

For the full syntax, type `asadmin help create-cluster`.

Starting a Cluster

A cluster is started using the `start-cluster` command. The following example starts the cluster named `mycluster`. The command prompts for the administrative password.

```
$ asadmin start-cluster --host myhost --port 1234 --user admin mycluster
```

`myhost` is the administrative server host, `1234` is the administrative port, `admin` is the administrative username.

For the full syntax, type `asadmin help start-cluster`. When a cluster is started, all the server instances in the cluster get started. A cluster without instances cannot be started.

Stopping a Cluster

A cluster is stopped using the `stop-cluster` command. The following example stops the cluster named `mycluster`. The command prompts for the administrative passwords.

```
$ asadmin stop-cluster --host myhost --port 1234 --user admin mycluster
```

`myhost` is the administrative server host, `1234` is the administrative port, `admin` is the administrative username.

For the full syntax, type `asadmin help stop-cluster`. When a cluster is stopped, all the server instances in the cluster get stopped. A cluster without instances cannot be stopped.

Creating a Node Agent

A node agent is created using the `create-node-agent` command. The following example creates node agent named `mynodeagent`. The administration server host is `myhost`, the administration server port is `1234`, and the administrative username is `admin`. The command prompts for the administrative passwords.

```
$ asadmin create-node-agent --host myhost --port 1234 --user admin mynodeagent
```

For the full syntax, type `asadmin help create-node-agent`.

Starting a Node Agent

A node agent is started using the `start-node-agent` command and specifying the node agent name. For example, to start the node agent `mynodeagent`, type the following:

```
$ asadmin start-node-agent --user admin mynodeagent
```

For the full syntax, type `asadmin help start-node-agent`.

Stopping a Node Agent

A node agent is stopped using the `stop-node-agent` command and specifying the node agent name. For example, to stop the node agent `mynodeagent`, type the following:

```
$ asadmin stop-node-agent mynodeagent
```

For the full syntax, type `asadmin help stop-node-agent`.

Creating an Instance

A server instance is created using the `create-instance` command. The following example creates an instance named `myinstance`. The administration server host is `myhost`, the administration server port is `1234`, and the administrative username is `admin`. The command prompts for the administrative passwords.

The following example creates a clustered server instance named `myinstance`. The command prompts for the administrative passwords.

```
$ asadmin create-instance --host myhost --port 1234
--user admin --cluster mycluster --nodeagent mynodeagent myinstance
```

myhost is the administrative server host, the administrative port is 1234, the administrative username is admin, the cluster this server instance belongs to is mycluster, the node agent managing this server instance is mynodeagent.

For the full syntax, type asadmin help create-instance.

To create a standalone server instance do not specify the --cluster option.

The following example creates a standalone server instance named myinstance managed by the node agent named mynodeagent.

```
$ asadmin create-instance --host myhost --port 1234
--user admin --nodeagent mynodeagent myinstance
```

Starting an Instance

A server instance is started using the start-instance command. The following example starts the server instance named myinstance. The command prompts for the administrative passwords.

```
$ asadmin start-instance --host myhost --port 1234 --user admin  myinstance
```

The administrative server host is myhost, the administrative port is 1234, the administrative username is admin. The server instance myinstance can be clustered or standalone.

For the full syntax, type asadmin help start-instance.

Stopping an Instance

A server instance is stopped using the stop-instance command. The following example stops the instance named myinstance. The command prompts for the administrative passwords.

```
$ asadmin stop-instance --host myhost --port 1234 --user admin  myinstance
```

The administrative server host is myhost, the administrative port is 1234, the administrative username is admin. The server instance myinstance can be clustered or standalone.

For the full syntax, type asadmin help stop-instance.

Restarting an Instance

To restart server instance, stop and start the instance.

Recreating the Domain Administration Server

For mirroring purposes, and to provide a working copy of the Domain Administration Server (DAS), you must have:

- One machine (machine1) that contains the original DAS.

- A second machine (machine2) that contains a cluster with server instances running applications and catering to clients. The cluster is configured using the DAS on the first machine.
- A third backup machine (machine3) where the DAS needs to be recreated in case the first machine crashes.

Note – You must maintain a backup of the DAS from the first machine. Use `asadmin backup-domain` to backup the current domain.

▼ To migrate the DAS

The following steps are required to migrate the Domain Administration Server from the first machine (machine1) to the third machine (machine3).

1 **Install the application server on the third machine just as it is installed on the first machine.**

This is required so that the DAS can be properly restored on the third machine and there are no path conflicts.

a. **Install the application server administration package using the command-line (interactive) mode. To activate the interactive command-line mode, invoke the installation program using the `console` option:**

./bundle-filename `-console`

You must have root permission to install using the command-line interface.

b. **Deselect the option to install default domain.**

Restoration of backed up domains is only supported on two machines with same architecture and **exactly** the same installation paths (use same *install-dir* and *domain-root-dir* on both machines).

2 **Copy the backup ZIP file from the first machine into the *domain-root-dir* on the third machine. You can also FTP the file.**

3 **Execute `asadmin restore-domain` command to restore the ZIP file onto the third machine:**

```
asadmin restore-domain --filename domain-root-dir/sjsas_backup_v00001.zip domain1
```

You can backup any domain. However, while recreating the domain, the domain name should be same as the original.

4 **Change *domain-root-dir*`/domain1/generated/tmp` directory permissions on the third machine to match the permissions of the same directory on first machine.**

The default permissions of this directory are: `?drwx------?` (or 700).

For example:

```
chmod 700 domain-root-dir/domain1/generated/tmp
```

The example above assumes you are backing up `domain1`. If you are backing up a domain by another name, you should replace `domain1` above with the name of the domain being backed up.

5 **Change the host values for the properties in the `domain.xml` file for the third machine:**

6 **Update the *domain-root-dir*`/domain1/config/domain.xml` on the third machine.**

For example, search for `machine1` and replace it with `machine3`. So, you can change:

```
<jmx-connector><property name=client-hostname value=machine1/>...
```

to:

```
<jmx-connector><property name=client-hostname value=machine3/>...
```

7 **Change:**

```
<jms-service... host=machine1.../>
```

to:

```
<jms-service... host=machine3.../>
```

8 **Start the restored domain on machine3:**

```
asadmin start-domain --user admin-user --password admin-password domain1
```

9 **Change the DAS host values for properties under node agent on machine2.**

10 **Change `agent.das.host` property value in *install-dir*`/nodeagents/`*nodeagent*`/agent/config/das.properties` on machine2.**

11 **Restart the node agent on machine2.**

Note – Start the cluster instances using the `asadmin start-instance` command to allow them to synchronize with the restored domain.

Changing the Administrator Password

To reset the administrator password, you must stop all the node agents in the domain. This stops all the associated server instances. All the server instances and node agents are now stopped and only the Domain Administration Server (DAS) is running.

Now you can change the administrative user's password as follows:

1. Change the admin password using the Command-line interface:

`asadmin update-file-user --authrealmname admin-realm ... --userpassword newpassword <admin-user-name>`

2. Change the admin password using the Admin console:

   Select the admin-server's configuration node > Security > Realms > admin-realm > Edit File Realm User.

   A message indicating that you have successfully changed the administrator password is displayed.

3. Restart the Domain Admin Server (DAS) with the new password as follows:

   Using the Command-line interface: `asadmin start-domain --user admin --password newpassword domain1`

   Assuming the following configuration: a domain with two node-agents (`i1na, c1-na`), and three instances (`c1i1` and `c1i2` are in the same cluster named `c1` and a standalone server instance `i1`).

4. Restart Node Agent without starting the instances with the new password.

   For example:

   `asadmin start-node-agent --user admin --password newpassword --startinstances=false i1-na asadmin`

   `asadmin start-node-agent --user admin --password newpassword --startinstances=false c1-na`

5. Restart the Servers and Clusters.

   `asadmin start-node-agent --user admin --password newpassword ... c1`

   `asadmin start-node-agent --user admin --password newpassword i1`

Application Server Configuration

The Sun Java System Application Server configuration is stored in the `domain.xml` file. The `domain.xml` is a document that represents the configuration state of the Application Server. It is the central repository for a given administrative domain. The document contains an XML representation of the Application Server domain model. The contents of `domain.xml` are governed by the specification expressed in the form of the domain DTD.

This section contains the following topics:

Changing Application Server Configuration

When making any of the following configuration changes, restarting the server is required for the changes to take effect:

- Changing JVM options
- Changing port numbers
- Managing HTTP, IIOP, and JMS services
- Managing thread pools

For instructions, see "Restarting the Domain" on page 36.

When making any of the following configuration changes, if dynamic configuration is enabled, restarting the server is **NOT** required for the changes to take effect:

- Deploying and undeploying applications
- Adding or removing JDBC, JMS, and Connector resources and pools
- Changing logging levels
- Adding file realm users
- Changing monitoring levels
- Enabling and disabling resources and applications

Note that the `asadmin reconfig` command has been deprecated and is no longer necessary. Configuration changes are applied to the server dynamically.

Ports in the Application Server

The following table describes the port listeners of the Application Server.

TABLE 1–1 Application Server Listeners that Use Ports

Listener	Default Port Number	Description
Administrative server	4849	A domain's administrative server is accessed by the Administration Console and the `asadmin` utility. For the Administration Console, specify the port number in the URL of the browser. When executing an `asadmin` command remotely, specify the port number with the --port option.
HTTP	8080	The Web server listens for HTTP requests on a port. To access deployed Web applications and services, clients connect to this port.
HTTPS	8181	Web applications configured for secure communications listen on a separate port.

TABLE 1–1 Application Server Listeners that Use Ports *(Continued)*

Listener	Default Port Number	Description
IIOP	3700	Remote clients of enterprise beans (EJB components) access the beans through the IIOP listener.
IIOP, SSL	3820/3890	Another port is used by the IIOP listener configured for secure communications.
IIOP, SSL and mutual authentication		Another port is used by the IIOP listener configured for mutual (client and server) authentication.
JMX	8686	Another port is used by the JMX connector to communicate with the DAS.

Changing the J2SE Software

The Application Server relies on the Java 2 Standard Edition (J2SE) software. When the Application Server was installed, the directory for the J2SE software was specified. For instructions on changing the J2SE software, see Chapter 17, "Java Virtual Machine and Advanced Settings."

◆ ◆ ◆ CHAPTER 2

Deploying Applications

This chapter explains how to deploy (install) J2EE applications on the Application Server. This chapter contains following sections:

The Deployment Life Cycle

After installing the Application Server and starting a domain, you can deploy (install) J2EE applications and modules. During deployment and as the application is changed, an application or module can go through the following stages:

1. Initial Deployment

   Before deploying an application or module, start the domain.

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

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

   You can also deploy to the domain using the `asadmin deploy` command, not the Administration Console. If you deploy the application or module only to the domain, it exists in the domain's central repository, but no server instances or clusters reference it until you add reference.

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

2. Enabling or Disabling

   By default, a deployed application or module is enabled, which means that it is runnable and can be accessed by clients if it has been deployed to an accessible server instance or cluster. To prevent access, disable the application or module. A disabled application or module is not uninstalled from the domain and can be easily enabled after deployment.

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

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

   To change which server instances and clusters reference an application or module after it is deployed, change an application or module's targets using the Administration Console, or change the application references using the `asadmin` tool. Because the application itself is stored in the central repository, adding or deleting targets adds or deletes the same version of an application on different targets. However, an application deployed to more than one target can be enabled on one and disabled on another, so even if an application is referenced by a target, it is not available to users unless it is enabled on that target.

4. Redeployment

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

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

   For production environments, use rolling upgrade, which upgrades application without interruption in service.

5. Undeployment

   To uninstall an application or module, undeploy it.

Autodeployment

The auto deploy feature enables you to deploy a prepackaged application or module by copying it to the *domain-dir*`/autodeploy` directory.

For example, copy a file named `hello.war` to the *domain-dir*`/autodeploy` directory. To undeploy the application, remove the `hello.war` file from the `autodeploy` directory.

You can also use the Administration Console or `asadmin` tool to undeploy the application. In that case, the archive file remains.

Note – Auto deploy is only available for the default server instance.

The auto deploy feature is intended for development environments. It is incompatible with session persistence, a production environment feature. Do not enable session persistence if auto deployment is enabled

Deploying An Unpackaged Application

This feature is for advanced developers.

Use directory deployment only to deploy to the default server instance (server). You cannot use it to deploy to clusters or stand-alone server instances.

A directory containing an unpackaged application or module is sometimes called an exploded directory. The contents of the directory must match the contents of a corresponding J2EE archive file. For example, if you deploy a Web application from a directory, the contents of the directory must be the same as a corresponding WAR file. For information about the required directory contents, see the appropriate specifications.

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

If your environment is configured to use dynamic reloading, you can also dynamically reload applications deployed from the directory. For more information, see "Configuring Advanced Settings" on page 196.

Using a Deployment Plan

This feature is for advanced developers.

A deployment plan is an JAR file that contains only the deployment descriptors that are specific to the Application Server. These deployment descriptors, for example `sun-application.xml`, are described in the *Application Server Developer's Guide*. The deployment plan is part of the implementation of *JSR 88: J2EE Application Deployment*. Use a deployment plan to deploy an application or module that does not contain the deployment descriptors that are specific to the Application Server.

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

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

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

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

In the deployment plan for a web application or a module file, the deployment descriptor that is specific to the Application Server is at the root level. If a stand-alone EJB module contains a CMP bean, the deployment plan includes the `sun-cmp-mappings.xml` and `.dbschema` files at the root level. In the following listing, the deployment plan describes a CMP bean.

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

Using the `deploytool` Utility

Designed for software developers, the `deploytool` utility packages and deploys J2EE applications and modules. For instructions on how to use `deploytool`, see *The J2EE 1.4 Tutorial*.

Types of J2EE Archive Files

A software provider packages an application or module into a archive file. To deploy the application or module, specify the archive file name. The content and structure of the archive file is defined by the specifications of the J2EE platform. Types of J2EE archive files are as follows:

- Web Application Archive (WAR): A WAR file consists of Web components such as servlets and JSPs, as well as static HTML pages, JAR files, tag libraries and utility classes. A WAR file name has the `.war` extension.

- EJB JAR: The EJB JAR file contains one or more enterprise beans, the components used for EJB technology. The EJB JAR file also includes any utility classes needed by the enterprise beans. The name of an EJB JAR file has the `.jar` extension.
- J2EE Application Client JAR: This JAR file contains the code for a J2EE application client, which accesses server-side components such as enterprise beans via RMI/IIOP. In the Administration Console, a J2EE application client is referred to as an "application client." The name of the J2EE application client JAR file has the `.jar` extension.
- Resource Adapter Archive (RAR): A RAR file holds a resource adapter. Defined by the J2EE Connector Architecture specifications, a resource adapter is a portable component that enables enterprise beans, Web components, and application clients to access resources and foreign enterprise systems. A resource adapter is often referred to as a connector. A RAR file name has the `.rar` extension.
- Enterprise Application Archive (EAR): An EAR file holds one or more WAR, EJB JAR, RAR or J2EE Application Client JAR files. An EAR file name has the `.ear` extension.

The software provider might assemble an application into a single EAR file or into separate WAR, EJB JAR, and application client JAR files. In the administration tools, the deployment pages and commands are similar for all types of files.

Naming Conventions

In a given domain, the names of deployed applications and modules must be unique.

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

Modules of different types can have the same name within an application. When the application is deployed, the directories holding the individual modules are named with `_jar`, `_war` and `_rar` suffixes. Modules of the same type within an application must have unique names. In addition, database schema file names must be unique within an application.

Using a Java package-like naming scheme is recommended for module filenames, EAR filenames, module names as found in the `<module-name>` portion of the `ejb-jar.xml` files, and EJB names as found in the `<ejb-name>` portion of the `ejb-jar.xml` files. The use of this package-like naming scheme ensures that name collisions do not occur. The benefits of this naming practice apply not only to the Application Server, but to other J2EE application servers as well.

JNDI lookup names for EJB components must also be unique. Establishing a consistent naming convention might help. For example, appending the application name and the module name to

the EJB name is one way to guarantee unique names. In this case, `mycompany.pkging.pkgingEJB.MyEJB` would be the JNDI name for an EJB in the module `pkgingEJB.jar`, which is packaged in the application `pkging.ear`.

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

◆ ◆ ◆ CHAPTER 3

JDBC Resources

A JDBC resource (data source) provides applications with a means of connecting to a database. Typically, the administrator creates a JDBC resource for each database accessed by the applications deployed in a domain. (However, more than one JDBC resource can be created for a database.)

This chapter contains the following sections:

Creating a JDBC Resource

To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API.

A JDBC resource (data source) provides applications with a means of connecting to a database. Before creating a JDBC resource, first create a JDBC connection pool.

To create a JDBC resource, specify a unique JNDI name that identifies the resource. Expect to find the JNDI name of a JDBC resource in `java:comp/env/jdbc` subcontext. For example, the JNDI name for the resource of a payroll database could be `java:comp/env/jdbc/payrolldb`. Because all resource JNDI names are in the `java:comp/env` subcontext, when specifying the JNDI name of a JDBC resource in the Administration Console, enter only `jdbc/`*name*. For example, for a payroll database specify `jdbc/payrolldb`.

To create a JDBC resource using the Admin Console, select Resources > JDBC Resources. Specify the resources settings as follows:

- JNDI Name: Specify a unique name. The JNDI name organizes and locates components within a distributed computing environment similarly to the way that card catalogs organize and represent locations of books in a library. Consequently, the JNDI name becomes an important method of accessing the JDBC resource. By convention, the name begins with the `jdbc/` string. For example: `jdbc/payrolldb`. Don't forget the forward slash.
- Pool Name: Choose the connection pool to be associated with the new JDBC resource.
- Description: Type a short description of the resource.
- Status: If you want the resource to be unavailable, deselect the Enabled checkbox. By default, the resource is available (enabled) as soon as it is created.

To create a JDBC resource using the command-line utility, use the `create-jdbc-resource` command.

Creating a JDBC Connection Pool

To create a JDBC resource, specify the connection pool with which it is associated. Multiple JDBC resources can specify a single connection pool.

A JDBC connection pool is a group of reusable connections for a particular database. Because creating each new physical connection is time consuming, the server maintains a pool of available connections to increase performance. When an application requests a connection, it obtains one from the pool. When an application closes a connection, the connection is returned to the pool.

When creating a connection pool , you are actually defining the aspects of a connection to a specific database. Before creating the pool, you must first install and integrate the JDBC driver. The properties of connection pools can vary with different database vendors. Some common properties are the database's name (URL), user name, and password.

Certain data specific to the JDBC driver and the database vendor must be entered. Before proceeding, gather the following information:

- Database vendor name
- Resource type, such as `javax.sql.DataSource` (local transactions only) `javax.sql.XADataSource` (global transactions)
- Data source class name: If the JDBC driver has a Datasource class for the resource type and database, then the value of the Datasource Classname field is required.
- Required properties, such as the database name (URL), user name, and password

A JDBC connection pool is a group of reusable connections for a particular database. When creating the pool with the Administration Console, the Administrator is actually defining the aspects of a connection to a specific database.

Before creating the pool, you must first install and integrate the JDBC driver. When building the Create Connection Pool pages, certain data specific to the JDBC driver and the database vendor must be entered. Before proceeding, gather the following information:

- Database vendor name
- Resource type, such as `javax.sql.DataSource` (local transactions only) `javax.sql.XADataSource` (global transactions)
- Data source class name
- Required properties, such as the database name (URL), user name, and password

Define the general settings values as specified by the JDBC driver you installed. These settings are the names of classes or interfaces in the Java programming language.

Parameter	Description
`DataSource Class Name`	The vendor-specific class name that implements the `DataSource` and / or `XADataSource` APIs. This class is in the JDBC driver.
`Resource Type`	Choices include `javax.sql.DataSource` (local transactions only), `javax.sql.XADataSource` (global transactions), and `java.sql.ConnectionPoolDataSource` (local transactions, possible performance improvements).

Additionally, you must define a set of physical database connections that reside in the pool. When an application requests a connection, the connection is removed from the pool, and when the application releases the connection, it is returned to the pool.

Parameter	Description
Initial and Minimum Pool Size	The minimum number of connections in the pool. This value also determines the number of connections placed in the pool when the pool is first created or when application server starts.
Maximum Pool Size	The maximum number of connections in the pool.
Pool Resize Quantity	When the pool shrinks toward the minimum pool size it is resized in batches. This value determines the number of connections in the batch. Making this value too large delays connection recycling; making it too small will be less efficient.
Idle Timeout	The maximum time in seconds that a connection can remain idle in the pool. After this time expires, the connection is removed from the pool.
Max Wait Time	The amount of time the application requesting a connection will wait before getting a connection timeout. Because the default wait time is long, the application might appear to hang indefinitely.

Optionally, the application server can validate connections before they are passed to applications. This validation allows the application server to automatically reestablish database connections if the database becomes unavailable due to network failure or database server crash. Validation of connections incurs additional overhead and slightly reduces performance.

Parameter	Description
Connection Validation	Select the Required checkbox to enable connection validation.
Validation Method	The application server can validate database connections in three ways: auto-commit, metadata, and table. auto-commit and metadata - The application server validates a connection by calling the `con.getAutoCommit()` and `con.getMetaData()` methods. However, because many JDBC drivers cache the results of these calls, they do not always provide reliable validations. Check with the driver vendor to determine whether these calls are cached or not. auto-commit validation makes use of two methods for validating the connection. `getAutoCommit()` is used to retrieve the current state of auto-commit and `setAutoCommit()` to change the state of auto-commit. This allows actual contact with the database to take place. `getAutomCommit()` might or might not contact the database, depending on the implementation. Actual physical connection will be wrapped for different purposes, such as a connection pool. **Note –** Databases such as Oracle do some caching for `setAutoCommit()`. For such databases, actual connection validation might not happen, so table-based validation is recommended. table - The application queries a database table that are specified. The table must exist and be accessible, but it doesn't require any rows. Do not use an existing table that has a large number of rows or a table that is already frequently accessed.
Table Name	If you selected table from the Validation Method combo box, then specify the name of the database table here.
On Any Failure	If you select the checkbox labelled Close All Connections, if a single connection fails, then the application server closes all connections in the pool and then reestablish them. If you do not select the checkbox, then individual connections are reestablished only when they are used.

Because a database is usually accessed by many users concurrently, one transaction might update data while another attempts to read the same data. The isolation level of a transaction defines the degree to which the data being updated is visible to other transactions. For details on isolation levels, refer to the documentation of the database vendor.

Parameter	Description
Transaction Isolation	Makes it possible to select the transaction isolation level for the connections of this pool. If left unspecified, the connections operate with default isolation levels provided by the JDBC driver.
Guaranteed Isolation Level	Only applicable if the isolation level has been specified. If you select the Guaranteed checkbox, then all connections taken from the pool have the same isolation level. For example, if the isolation level for the connection is changed programmatically (with `con.setTransactionIsolation`) when last used, this mechanism changes the status back to the specified isolation level.

How JDBC Resources and Connection Pools Work Together

To store, organize, and retrieve data, most applications use relational databases. Java EE applications access relational databases through the JDBC API. Before an application can access a database, it must get a connection.

At runtime, here's what happens when an application connects to a database:

1. The application gets the JDBC resource (data source) associated with the database by making a call through the JNDI API.

   Given the resource's JNDI name, the naming and directory service locates the JDBC resource. Each JDBC resource specifies a connection pool.

2. Via the JDBC resource, the application gets a database connection.

   Behind the scenes, the application server retrieves a physical connection from the connection pool that corresponds to the database. The pool defines connection attributes such as the database name (URL), user name, and password.

3. Now that it's connected to the database, the application can read, modify, and add data to the database.

   The applications access the database by making calls to the JDBC API. The JDBC driver translates the application's JDBC calls into the protocol of the database server.

4. When it's finished accessing the database, the application closes the connection.

   The application server returns the connection to the connection pool. Once it's back in the pool, the connection is available for the next application.

Setting Up Database Access

To set up a database access, you must first install the supported database product. For a list of database products supported by the Application Server, see the *Release Notes*. Then, you must install a JDBC driver for the database product and make the driver's JAR file accessible to the domain's server instance. Next, create the database, the connection pool for the database, and the JDBC resource that points to the connection pool.

After installing the JDBC driver, you must integrate it so that the driver translates an application's JDBC calls into the protocol of the database server. To integrate the JDBC driver into an administrative domain, do either of the following:

- Make the driver accessible to the common class loader.
  1. Copy the driver's JAR and ZIP files into the *domain-dir*`/lib` directory or copy its class files into the *domain-dir*`/lib/ext` directory.
  2. Restart the domain.
- Make the driver accessible to the system class loader.
  1. In the Administration Console, select the JVM Settings for the desired configuration.
  2. Identify the fully-qualified path name for the driver's JAR file.
  3. Save the settings and restart the server.

Persistence Manager Resources

This feature is needed for backward compatibility. To run on version 7 of the Application Server, a persistent manager resource was required for applications with container-managed persistence beans (a type of EJB component). Using a JDBC resource instead is recommended.

To create a persistence manager resource using the Admin console, select the Resources node . Persistence Managers node > Persistence Manager page. To use the command-line utility, use the `create-persistence-resource` command.

◆ ◆ ◆ CHAPTER 4

Configuring Java Message Service Resources

This chapter describes how to configure resources for applications that use the Java Message Service (JMS) API. It contains the following sections:

About JMS Resources

The JMS Provider in the Application Server

The Application Server implements the Java Message Service (JMS) API by integrating the Sun Java System Message Queue (formerly Sun ONE Message Queue) software into the Application Server. For basic JMS API administration tasks, use the Application Server Administration Console. For advanced tasks, including administering a Message Queue cluster, use the tools provided in the *MQ-install-dir*`/imq/bin` directory.

For details about administering Message Queue, see the *Message Queue Administration Guide*.

JMS Resources

The Java Message Service (JMS) API uses two kinds of administered objects:

- Connection factories, objects that allow an application to create other JMS objects programmatically
- Destinations, which serve as the repositories for messages

These objects are created administratively, and how they are created is specific to each implementation of JMS. In the Application Server, perform the following tasks:

- Create a connection factory by creating a connection factory resource
- Create a destination by creating two objects:
  - A physical destination
  - A destination resource that refers to the physical destination

JMS applications use the JNDI API to access the connection factory and destination resources. A JMS application normally uses at least one connection factory and at least one destination. To learn what resources to create, study the application or consult with the application developer.

There are three types of connection factories:

- `QueueConnectionFactory` objects, used for point-to-point communication
- `TopicConnectionFactory` objects, used for publish-subscribe communication
- `ConnectionFactory` objects, which can be used for both point-to-point and publish-subscribe communications; these are recommended for new applications

There are two kinds of destinations:

- `Queue` objects, used for point-to-point communication
- `Topic` objects, used for publish-subscribe communication

The chapters on JMS in the *J2EE 1.4 Tutorial* provide details on these two types of communication and other aspects of JMS (see `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`).

The order in which the resources are created does not matter.

For a J2EE application, specify connection factory and destination resources in the Application Server deployment descriptors as follows:

- Specify a connection factory JNDI name in a `resource-ref` or an `mdb-connection-factory` element.
- Specify a destination resource JNDI name in the `ejb` element for a message-driven bean and in the `message-destination` element.
- Specify a physical destination name in a `message-destination-link` element, within either a `message-driven` element of an enterprise bean deployment descriptor or a `message-destination-ref` element. In addition, specify it in the `message-destination` element. (The `message-destination-ref` element replaces the `resource-env-ref`

element, which is deprecated in new applications.) In the `message-destination` element of an Application Server deployment descriptor, link the physical destination name with the destination resource name.

The Relationship Between JMS Resources and Connector Resources

The Application Server implements JMS by using a system resource adapter named `jmsra`. When a user creates JMS resources, the Application Server automatically creates connector resources that appear under the Connectors node in the Administration Console's tree view.

For each JMS connection factory that a user creates, the Application Server creates a connector connection pool and connector resource. For each JMS destination a user creates, the Application Server creates an admin object resource. When the user deletes the JMS resources, the Application Server automatically deletes the connector resources.

It is possible to create connector resources for the JMS system resource adapter by using the Connectors node of the Administration Console instead of the JMS Resources node. See Chapter 7, "Connector Resources," for details.

JMS Connection Factories

JMS connection factories are objects that allow an application to create other JMS objects programmatically. These administered objects implement the `ConnectionFactory`, `QueueConnectionFactory`, and `TopicConnectionFactory` interfaces. Using the Application Server Admin Console, you can create, edit, or delete a JMS Connection Factory. The creation of a new JMS connection factory also creates a connector connection pool for the factory and a connector resource.

To manage JMS connection factories using the command-line utility, use `create-jms-resource`, `list-jms-resources`, or `delete-jms-resource` command.

JMS Destination Resources

JMS destinations serve as the repositories for messages. Using the Admin Console, you can create, modify or delete JMS Destination Resources. To create a new JMS Destination Resource, select Resources>JMS Resources>Destination Resources. In the Destination Resources page, you can specify the following:

- JNDI Name for the resource. It is a recommended practice to use the naming subcontext prefix `jms/` for JMS resources. For example: `jms/Queue`.

- The resource type, which can be `javax.jms.Topic` or `javax.jms.Queue`.
- Additional properties for the destination resource. For more details about all these settings and the additional properties, refer to the Admin Console Online Help.

To manage JMS destinations using the command-line utility, use `create-jms-resource`, or `delete-jms-resource` command.

Tip – To specify the `addresslist` property (in the format `host:mqport,host2:mqport,host3:mqport`) for `asadmin create-jms-resource` command, escape the `:` by using `\\`. For example, `host1\\:mqport,host2\\:mqport,host3\\:mpqport`.

For more information on using escape characters, see the `asadmin`(8) man page.

JMS Physical Destinations

For production purposes, always create physical destinations. During the development and testing phase, however, this step is not required.

▼ To Create a JMS Physical Destination

The first time an application accesses a destination resource, Message Queue automatically creates the physical destination specified by the Name property of the destination resource. The physical destination is temporary and expires after a period specified by a Message Queue configuration property. Physical destinations are attached to Message Queue instances and not to Application Server instances. Any Application Server instance can talk to any Message Queue instance because there is no tight coupling between Message Queue and Application Server.

1. **In the Admin Console, select Configuration >Physical Destinations.**
2. **In the Create Physical Destinations page, specify a name for the physical destination.**
3. **Choose the type of destination, which can be `topic` or `queue`.**

See Also To manage JMS physical destinations with the `asadmin` command-line utility, use the `create-jmsdest`, `flush-jmsdest`, or `delete-jmsdest` subcommands.

For more details about the fields and properties of the Physical Destinations page, see the Admin Console online help.

▼ To Create a Configuration-Specific JMS Physical Destination

By default, all the instances in Application Server share the same Default JMS Host which is used by the domain administration server (DAS). In other words, all physical destinations are the same. If your environment requires configuration-specific physical destinations, you must create a separate JMS host for each instance or cluster as described in the following steps:

1. **In the Edit JMS Hosts page of the Admin Console, create a new JMS host with its own port.**
2. **In the JMS Service page, create a JMS Host as follows:**
   a. **Set the Type to LOCAL.**
   b. **In the Default JMS Host field, specify the host that you created in Step 1.**

   You can also set other options on this page if needed.
3. **Save the changes and restart the instance or cluster.**

   Now you are ready to create configuration-specific physical destinations.

JMS Providers

Configuring General Properties for the JMS Provider

Use the JMS Service page in the Admin Console to configure properties to be used by all JMS connections. In the Admin Console, select Configurations>Java Message Service. In the JMS Service page, you can control the following general JMS settings.

- Select Startup Timeout interval, which indicates the time that Application Server waits for the JMS service to start before aborting the startup.
- Select JMS Service type, which decides whether you manage a JMS Service on a local or a remote host.
- Specify Start Arguments to customize the JMS service startup.
- Select Reconnect checkbox to specify whether the JMS service attempts to reconnect to a message server (or the list of addresses in the `AddressList`) when a connection is lost.
- Specify Reconnect Interval in terms of number of seconds. This applies for attempts on each address in the `AddressList` and for successive addresses in the list. If it is too short, this time interval does not give a broker time to recover. If it is too long, the reconnect might represent an unacceptable delay.

- Specify the number of reconnect attempts. In the field, type the number of attempts to connect (or reconnect) for each address in the `AddressList` before the client runtime tries the next address in the list.
- Choose the default JMS host.
- In the Address List Behavior drop-down list, choose whether connection attempts are in the order of addresses in the `AddressList` (`priority`) or in a random order (`random`).
- In the Address List Iterations field, type the number of times the JMS service iterates through the `AddressList` in an effort to establish (or reestablish) a connection.
- In the MQ Scheme and MQ Service fields, type the Message Queue address scheme name and the Message Queue connection service name if a non-default scheme or service is to be used.

Values of all these properties can be updated at run time too. However, only those connection factories that are created after the properties are updated, will get the updated values. The existing connection factories will continue to have the original property values. Also, for almost all of the values to take effect, the application server needs to be restarted. The only property that can be updated without having to restart the application server is the default JMS host.

To manage JMS providers using the command-line utility, use the `set` or `jms-ping` commands.

Accessing Remote Servers

Changing the provider and host to a remote system causes all JMS applications to run on the remote server. To use both the local server and one or more remote servers, create a connection factory resource with the AddressList property to create connections that access remote servers.

Foreign JMS Providers

Generic Resource Adapter for JMS is a Java EE Connector 1.5 resource adapter that can wrap the JMS client library of external JMS providers such as IBM Websphere MQ, Tibco EMS, and Sonic MQ among others, and thus integrate any JMS provider with a Java EE 1.4 application server such as Sun Java System Application Server. The adapter is a `.rar` archive that can be deployed and configured using a Java EE 1.4 application server's administration tools.

Configuring the Generic Resource Adapter for JMS

Application server's administration tools can be used to deploy and configure the generic resource adapter for JMS. This section explains how to configure Generic Resource Adapter for JMS with Sun Java System Application Server. Overall, the Resource Adapter can be configured to indicate whether the JMS provider supports XA or not. It is also possible to indicate what mode of integration is possible with the JMS provider. Two modes of integration are supported

by the resource adapter. First one uses JNDI as the means of integration. In this case, administered objects are set up in JMS provider's JNDI tree and will be looked up for use by the generic resource adapter. If that mode is not suitable for integration, it is also possible to use Java reflection of JMS administered object javabean classes as the mode of integration. You can use the Sun Java System Application Server's Administration Console or the CLI to configure the resource adapter. This is not different from configuring any other resource adapter.

Configuring the Generic Resource Adapter

Prior to deploying the resource adapter, JMS client libraries should be made available to the application server. For some JMS providers, client libraries may also include native libraries. In such cases, these native libraries should also be made available to the application server JVM(s).

1. Deploy the generic resource adapter the same way you would deploy a connector module.

   For steps to do this, refer to the Admin Console Online Help. During deployment, make sure that you specify `install-dir/lib/addons/resourceadapters/genericjmsra/genericra.rar` as the location of the generic resource adapter. Also, you must specify the properties explained in the section "Resource Adapter Properties" on page 64.

2. Create a connector connection pool.

   For steps to do this, refer to the Admin Console Online help. In the New Connector Connection Pool page, from the Resource Adapter combo box, select `genericra`. Also, in the Connection Definition combo box, select `javax.jms.QueueConnectionFactory`. Additionally, specify the properties explained in the section "`ManagedConnectionFactory` Properties" on page 67.

3. Create a connector resource.

   For detailed procedure to do this, you can refer the Admin Console Online Help. In the New Connector Resource page, select the pool you created in the previous step.

4. Create an administered object resource.

   For detailed procedure to do this, you can refer the Admin Console Online Help. In the New Admin Object Resource page, select `genericra` as the Resource Adapter and `javax.jms.Queue` as the Resource Type. Click Next and in the second page, click Add Property. In the Additional Properties table, specify a new property called `DestinationProperties` with the value `Name\\=clientQueue`. For information on more properties, see the section "Administered Object Resource Properties" on page 67.

5. Make the following changes to the security policy in Sun Java System Application Server.

   - Modify `sjsas_home/domains/domain1/config/server.policy` to add `java.util.logging.LoggingPermission "control"`
   - Modify `sjsas_home/lib/appclient/client.policy` to add `permission javax.security.auth.PrivateCredentialPermission "javax.resource.spi.security.PasswordCredential * \"*\"","read";`

Resource Adapter Properties

The following table presents the properties to be used while creating the resource adapter.

Property Name	*Valid Values*	*Default value*	*Description*
`ProviderIntegrationMode`	`javabean`/jndi	`javabean`	Decides the mode of integration between the resource adapter and the JMS client.
`ConnectionFactoryClassName`	Name of the class available in the application server classpath, for example: `com.sun.messaging.ConnectionFactory`	none	Class name of `javax.jms.ConnectionFactory` implementation of the JMS client. Used if `ProviderIntegrationMode` is `javabean`.
`QueueConnectionFactoryClassName`	Name of the class available in the application server classpath, for example: `com.sun.messaging.QueueConnectionFactory`	none	Class name of `javax.jms.QueueConnectionFactory` implementation of the JMS client. This is used if `ProviderIntegrationMode` is `javabean`.
`TopicConnectionFactoryClassName`	Name of the class available in the application server classpath , for example: `com.sun.messaging.TopicConnectionFactory`	none	Class name of `javax.jms.TopicConnectionFactory` implementation of the JMS client. This is used if `ProviderIntegrationMode` is specified as `javabean`.
`XAConnectionFactoryClassName`	Name of the class available in application server classpath , for example: `com.sun.messaging.XAConnectionFactory`	none	Class name of `javax.jms.ConnectionFactory` implementation of the JMS client. This is used if `ProviderIntegrationMode` is specified as `javabean`.
`XAQueueConnectionFactoryClassName`	Name of the class available in application server classpath , for example: `com.sun.messaging.XAQueueConnectionFactory`	none	Class name of `javax.jms.XAQueueConnectionFactory` implementation of the JMS client. This is used if `ProviderIntegrationMode` is specified as `javabean`.

Property Name	*Valid Values*	*Default value*	*Description*
`XATopicConnectionFactoryClassName`	Name of the class available in application server classpath , for example: `com.sun.messaging.XATopicConnectionFactory`	none	Class name of `javax.jms.XATopicConnectionFactory` implementation of the JMS client. This is used if `ProviderIntegrationMode` is `javabean`.
`TopicClassName`	Name of the class available in application server classpath, for example : `com.sun.messaging.Topic`	none	Class Name of `javax.jms.Topic` implementation of the JMS client. This is used if `ProviderIntegrationMode` is `javabean`.
`QueueClassName`	Name of the class available in application server classpath, for example: `com.sun.messaging.Queue`	none	Class Name of `javax.jms.Queue` implementation of the JMS client. This is used if `ProviderIntegrationMode` is specified as a `javabean`.
`SupportsXA`	`True/false`	`FALSE`	Specifies whether the JMS client supports XA or not.
`ConnectionFactoryProperties`	Name value pairs separated by comma.	none	This specifies the `javabean` property names and values of the `ConnectionFactory` of the JMS client. This is required only if `ProviderIntegrationMode` is `javabean`.
`JndiProperties`	Name value pairs separated by comma.	none	This specifies the JNDI provider properties to be used for connecting to the JMS provider's JNDI. This is used only if `ProviderIntegrationMode` is `jndi`.
`CommonSetterMethodName`	Method name	none	This specifies the common setter method name that some JMS vendors use to set the properties on their administered objects. This property is used only if `ProviderIntegrationMode` is `javabean`. In the case of Sun Java System Message Queue, this property is named `setProperty`.
`UserName`	Name of the JMS user	none	User name to connect to the JMS Provider.

Property Name	*Valid Values*	*Default value*	*Description*
`Password`	Password for the JMS user.	none	Password to connect to the JMS provider.
`RMPolicy`	`ProviderManaged` or `OnePerPhysicalConnection`	`ProviderManaged`	The `isSameRM` method on an `XAResource` is used by the Transaction Manager to determine if the Resource Manager instance represented by two `XAResources` are the same. When `RMPolicy` is set to `ProviderManaged`(the default value), the JMS provider is responsible for determining the `RMPolicy` and the `XAResource` wrappers in the Generic Resource Adapter merely delegate the `isSameRM` call to the message queue provider's XA resource implementations. This should ideally work for most message queue products. Some `XAResource` implementations, such as IBM MQ Series, rely on a resource manager per physical connection. This causes issues when there is inbound and outbound communication to the same queue manager in a single transaction (for example, when an MDB sends a response to a destination). When `RMPolicy` is set to `OnePerPhysicalConnection`, the `XAResource` wrapper implementation's `isSameRM` in Generic Resource Adapter would check if both the `XAResources` use the same physical connection, before delegating to the wrapped objects.

`ManagedConnectionFactory` Properties

`ManagedConnectionFactory` properties are specified when a connector-connection-pool is created. All the properties specified while creating the resource adapter can be overridden in a `ManagedConnectionFactory`. Additional properties available only in `ManagedConnectionFactory` are given below.

Property Name	*Valid Value*	*Default Value*	*Description*
`ClientId`	A valid client ID	none	`ClientID` as specified by JMS 1.1 specification.
`ConnectionFactoryJndiName`	JNDI Name	none	JNDI name of the connection factory bound in the JNDI tree of the JMS provider. The administrator should provide all connection factory properties (except `clientID`) in the JMS provider itself. This property name will be used only if `ProviderIntegratinMode` is `jndi`.
`ConnectionValidationEnabled`	true/false	FALSE	If set to true, the resource adapter will use an exception listener to catch any connection exception and will send a `CONNECTION_ERROR_OCCURED` event to application server.

Administered Object Resource Properties

Properties in this section are specified when an administered object resource is created. All the resource adapter properties can be overridden in an administered resource object. Additional properties available only in the administered object resource are given below.

Property Name	*Valid Value*	*Default Value*	*Description*
`DestinationJndiName`	JNDI Name	none	JNDI name of the destination bound in the JNDI tree of the JMS provider. Administrator should provide all properties in the JMS provider itself. This property name will be used only if `ProviderIntegrationMode` is `jndi`.

Property Name	*Valid Value*	*Default Value*	*Description*
`DestinationProperties`	Name value pairs separated by comma	none	This specifies the `javabean` property names and values of the destination of the JMS client. Required only if `ProviderIntegrationMode` is `javabean`.

Activation Spec Properties

Properties in this section are specified in the sun specific deployment descriptor of MDB as activation-config-properties. All the resource adapter properties can be overridden in an Activation Spec. Additional properties available only in ActivationSpec are given below.

Property Name	*Valid Value*	*Default Value*	*Description*
`MaxPoolSize`	An integer	8	Maximum size of server session pool internally created by the resource adapter for achieving concurrent message delivery. This should be equal to the maximum pool size of MDB objects.
`MaxWaitTime`	An integer	3	The resource adapter will wait for the time in seconds specified by this property to obtain a server session from its internal pool. If this limit is exceeded, message delivery will fail.
`SubscriptionDurability`	Durable or Non-Durable	Non-Durable	`SubscriptionDurability` as specified by JMS 1.1 specification.
`SubscriptionName`		none	`SubscriptionName` as specified by JMS 1.1 specification.
`MessageSelector`	A valid message selector	none	`MessageSelector` as specified by JMS 1.1 specification.
`ClientID`	A valid client ID	none	`ClientID` as specified by JMS 1.1 specification.

Property Name	*Valid Value*	*Default Value*	*Description*
`ConnectionFactoryJndiName`	A valid JNDI Name	none	JNDI name of connection factory created in JMS provider. This connection factory will be used by resource adapter to create a connection to receive messages. Used only if `ProviderIntegrationMode` is configured as `jndi`.
`DestinationJndiName`	A valid JNDI Name	none	JNDI name of destination created in JMS provider. This destination will be used by resource adapter to create a connection to receive messages from. Used only if `ProviderIntegrationMode` is configured as `jndi`.
`DestinationType`	`javax.jms.Queue` or `javax.jms.Topic`	null	Type of the destination the MDB will listen to.
`DestinationProperties`	Name-value pairs separated by comma	none	This specifies the `javabean` property names and values of the destination of the JMS client. Required only if `ProviderIntegrationMode` is `javabean`.
`RedeliveryAttempts`	integer		Number of times a message will be delivered if a message causes a runtime exception in the MDB.
`RedeliveryInterval`	time in seconds		Interval between repeated deliveries, if a message causes a runtime exception in the MDB.
`SendBadMessagesToDMD`	true/false	false	Indicates whether the resource adapter should send the messages to a dead message destination, if the number of delivery attempts is exceeded.
`DeadMessageDestinationJndiName`	a valid JNDI name.	none	JNDI name of the destination created in the JMS provider. This is the target destination for dead messages. This is used only if `ProviderIntegrationMode` is `jndi`.
`DeadMessageDestinationClassName`	class name of destination object.	none	Used if `ProviderIntegrationMode` is `javabean`.

Property Name	*Valid Value*	*Default Value*	*Description*
`DeadMessageDestinationPropertiesName`	Name Value Pairs separated by comma	none	This specifies the `javabean` property names and values of the destination of the JMS client. This is required only if `ProviderIntegrationMode` is `javabean`.
`ReconnectAttempts`	integer		Number of times a reconnect will be attempted in case exception listener catches an error on connection.
`ReconnectInterval`	time in seconds		interval between reconnects.

◆ ◆ ◆ CHAPTER 5

Configuring JavaMail Resources

The Application Server includes the JavaMail API. The JavaMail API is a set of abstract APIs that model a mail system. The API provides a platform-independent and protocol-independent framework to build mail and messaging applications. The JavaMail API provides facilities for reading, composing, and sending electronic messages. Service providers implement particular protocols. Using the JavaMail API you can add email capabilities to your applications. JavaMail provides access from Java applications to Internet Message Access Protocol (IMAP)- and Simple Mail Transfer Protocol (SMTP)-capable mail servers on your network or the Internet. It does not provide mail server functionality; you must have access to a mail server to use JavaMail.

To learn more about the JavaMail API, consult the JavaMail web site at `http://java.sun.com/products/javamail/index.html`

This chapter contains the following section:

Creating a JavaMail Session

To configure JavaMail for use in Application Server, create a Mail Session in the Application Server Admin Console. This allows server-side components and applications to access JavaMail services with JNDI, using the Session properties you assign for them. When creating a Mail Session, you can designate the mail hosts, transport and store protocols, and the default mail user in the Admin Console so that components that use JavaMail do not have to set these properties. Applications that are heavy email users benefit because the Application Server creates a single Session object and makes it available via JNDI to any component that needs it.

To create a JavaMail session using the Admin Console, select Resources > JavaMail Sessions. Specify the JavaMail settings as follows:

- JNDI Name: The unique name for the mail session. Use the naming sub-context prefix `mail/` for JavaMail resources. For example: `mail/MySession`.

- Mail Host: The host name of the default mail server. The connect methods of the Store and Transport objects use this value if a protocol-specific host property is not supplied. The name must be resolvable to an actual host name.
- Default User: The user name to provide when connecting to a mail server. The connect methods of the Store and Transport objects use this value if a protocol-specific username property is not supplied.
- Default Return Address: The email address of the default user, in the form: *username*@*host*.*domain*.
- Description: Provide a descriptive statement for the component.
- Session: Deselect the Enabled checkbox if you do not want to enable the mail session at this time.

Additionally, define the following Advanced settings only if the mail provider has been re-configured to use a non-default store or transport protocol:

- Store Protocol: Defines the Store object communication method to be used. By default, the Store Protocol is `imap`.
- Store Protocol Class: Provides the Store communication method class that implements the desired Store protocol. By default, the Store Protocol Class is `com.sun.mail.imap.IMAPStore`.
- Transport Protocol: Identifies the transport communication method. By default, the Transport Protocol is `smtp`.
- Transport Protocol Class: Defines the communication method for the transport class. By default, the Transport Protocol Class is `com.sun.mail.smtp.SMTPTransport`
- Debug: Select this checkbox to enable extra debugging output, including a protocol trace, for this mail session. If the JavaMail log level is set to `FINE` or finer, the debugging output is generated and is included in the system log file. See "Setting Custom Log Levels" on page 146 for information about setting the log level.
- Additional Properties: Create properties required by applications, such as a protocol-specific host or username property. The JavaMail API documentation lists the available properties.

◆ ◆ ◆ CHAPTER 6

JNDI Resources

The Java Naming and Directory Interface (JNDI) is an application programming interface (API) for accessing different kinds of naming and directory services. Java EE components locate objects by invoking the JNDI lookup method.

JNDI is the acronym for the Java Naming and Directory Interface API. By making calls to this API, applications locate resources and other program objects. A resource is a program object that provides connections to systems, such as database servers and messaging systems. (A JDBC resource is sometimes referred to as a data source.) Each resource object is identified by a unique, people-friendly name, called the JNDI name. A resource object and its JNDI name are bound together by the naming and directory service, which is included with the Application Server. To create a new resource, a new name-object binding is entered into the JNDI.

This chapter contains the following sections:

J2EE Naming Services

A JNDI name is a people-friendly name for an object. These names are bound to their objects by the naming and directory service that is provided by a J2EE server. Because J2EE components access this service through the JNDI API, the object usually uses its JNDI name. For example, the JNDI name of the PointBase database is `jdbc/Pointbase`. When it starts up, the Application Server reads information from the configuration file and automatically adds JNDI database names to the name space.

J2EE application clients, enterprise beans, and web components are required to have access to a JNDI naming environment.

The application component's naming environment is a mechanism that allows customization of the application component's business logic during deployment or assembly. Use of the application component's environment allows the application component to be customized without the need to access or change the application component's source code.

A J2EE container implements the application component's environment, and provides it to the application component instance as a JNDI naming context. The application component's environment is used as follows:

- The application component's business methods access the environment using the JNDI interfaces. The application component provider declares in the deployment descriptor all the environment entries that the application component expects to be provided in its environment at runtime.
- The container provides an implementation of the JNDI naming context that stores the application component environment. The container also provides the tools that allow the deployer to create and manage the environment of each application component.
- A deployer uses the tools provided by the container to initialize the environment entries that are declared in the application component's deployment descriptor. The deployer sets and modifies the values of the environment entries.
- The container makes the environment naming context available to the application component instances at runtime. The application component's instances use the JNDI interfaces to obtain the values of the environment entries.

Each application component defines its own set of environment entries. All instances of an application component within the same container share the same environment entries. Application component instances are not allowed to modify the environment at runtime.

Naming References and Binding Information

A resource reference is an element in a deployment descriptor that identifies the component's coded name for the resource. More specifically, the coded name references a connection factory for the resource. In the example given in the following section, the resource reference name is `jdbc/SavingsAccountDB`.

The JNDI name of a resource and the name of the resource reference are not the same. This approach to naming requires that you map the two names before deployment, but it also decouples components from resources. Because of this de-coupling, if at a later time the component needs to access a different resource, the name does not need to change. This flexibility also makes it easier for you to assemble J2EE applications from preexisting components.

The following table lists JNDI lookups and their associated references for the J2EE resources used by the Application Server.

TABLE 6–1 JNDI Lookups and Their Associated References

JNDI Lookup Name	Associated Reference
`java:comp/env`	Application environment entries
`java:comp/env/jdbc`	JDBC DataSource resource manager connection factories
`java:comp/env/ejb`	EJB References
`java:comp/UserTransaction`	UserTransaction references
`java:comp/env/mail`	JavaMail Session Connection Factories
`java:comp/env/url`	URL Connection Factories
`java:comp/env/jms`	JMS Connection Factories and Destinations
`java:comp/ORB`	ORB instance shared across application components

Using Custom Resources

A custom resource accesses a local JNDI repository and an external resource accesses an external JNDI repository. Both types of resources need user-specified factory class elements, JNDI name attributes, etc. In this section, we will discuss how to configure JNDI connection factory resources, for J2EE resources, and how to access these resources.

Within the Application Server, you can create, delete, and list resources, as well as `list-jndi-entities`.

Using External JNDI Repositories and Resources

Often applications running on the Application Server require access to resources stored in an external JNDI repository. For example, generic Java objects could be stored in an LDAP server as per the Java schema. External JNDI resource elements let users configure such external resource repositories. The external JNDI factory must implement `javax.naming.spi.InitialContextFactory` interface.

An example of the use of an external JNDI resource is:

```
<resources>
 <!-- external-jndi-resource element specifies how to access J2EE resources
 -- stored in an external JNDI repository. The following example
 -- illustrates how to access a java object stored in LDAP.
 -- factory-class element specifies the JNDI InitialContext factory that
 -- needs to be used to access the resource factory. property element
 -- corresponds to the environment applicable to the external JNDI context
```

```
 -- and jndi-lookup-name refers to the JNDI name to lookup to fetch the
 -- designated (in this case the java) object.
 -->
  <external-jndi-resource jndi-name="test/myBean"
      jndi-lookup-name="cn=myBean"
      res-type="test.myBean"
      factory-class="com.sun.jndi.ldap.LdapCtxFactory">
    <property name="PROVIDER-URL" value="ldap://ldapserver:389/o=myObjects" />
    <property name="SECURITY_AUTHENTICATION" value="simple" />
    <property name="SECURITY_PRINCIPAL", value="cn=joeSmith, o=Engineering" />
    <property name="SECURITY_CREDENTIALS" value="changeit" />
  </external-jndi-resource>
</resources>
```

◆ ◆ ◆ CHAPTER 7

Connector Resources

A connector module (also called a resource adapter), is a Java component that enables applications to interact with enterprise information systems (EISs). EIS software includes various types of systems: enterprise resource planning (ERP), mainframe transaction processing, and non-relational databases, among others. To install a connector module you deploy it, as you do for other Java modules.

A connector connection pool is a group of reusable connections for a particular EIS. To create a connector connection pool, specify the connector module (resource adapter) that is associated with the pool.

A connector resource is a program object that provides an application with a connection to an EIS. To create a connector resource, specify its JNDI name and its associated connection pool. Multiple connector resources can specify a single connection pool. The application locates the resource by looking up its JNDI name. The JNDI name of a connector resource for an EIS is usually in the `java:comp/env/`*eis-specific* subcontext. Application Server 9 implements JMS by using a connector module (resource adapter)

This chapter covers:

Connector Connection Pools

The following table describes connection pool settings:

Parameter	Description
Initial and Minimum Pool Size	The minimum number of connections in the pool. This value also determines the number of connections placed in the pool when the pool is first created or when application server starts.
Maximum Pool Size	The maximum number of connections in the pool.
Pool Resize Quantity	When the pool shrinks toward the minimum pool size it is resized in batches. This value determines the number of connections in the batch. Making this value too large will delay connection recycling; making it too small will be less efficient.
Idle Timeout	The maximum time in seconds that a connection can remain idle in the pool. After this time expires, the connection will be removed from the pool.
Max Wait Time	The amount of time the application that has requested a connection will wait before getting a connection time-out. Because the default wait time is long, the application might appear to hang indefinitely.
On Any Failure	If you select the checkbox labelled Close All Connections, if a single connection fails, then the application server will close all connections in the pool and then reestablish them. If you do not select the checkbox, then individual connections will be reestablished only when they are used.
Transaction Support	Use the Transaction Support list to select the type of transaction support for the connection pool. The chosen transaction support overrides the transaction support attribute in the resource adapter associated with this connection pool in a downward compatible way. In other words, it can support a lower transaction level than that specified in the resource adapter or the same transaction level as that specified in resource adapter, but it cannot specify a higher level. The None selection from the Transaction Support menu indicates that the resource adapter does not support resource manager local or JTA transactions and does not implement `XAResource` or `LocalTransaction` interfaces. For JAXR resource adapters, you need to choose None from the Transaction Support menu. JAXR resource adapters do not support local or JTA transactions. Local transaction support means that the resource adapter supports local transactions by implementing the `LocalTransaction` interface. Local transactions are managed internal to a resource manager and involve no external transaction managers. XA transaction support means that the resource adapter supports resource manager local and JTA transactions by implementing the `LocalTransaction` and `XAResource` interfaces. XA transactions are controlled and coordinated by a transaction manager external to a resource manager. Local transactions are managed internal to a resource manager and involve no external transaction managers.
Connector Validation	Select the Enabled checkbox if you want the connection pool to be validated before being passed on to the application.

Before creating a connection pool, you need to deploy the connector module (resource adapter) associated with the pool. You can deploy a connector module using the Admin Console or by using the `asadmin` command. For information about the `asadmin`command see, `asadmin`(1M)

To view, create, edit, or delete connection pools in the Admin Console, click Resources > Connectors > Connector Connection Pools. You can add properties (a name-value pair) to a connector connection pool. Alternatively, you can use the following `asadmin` commands to create and delete connection pools:

- `create-connector-connection-pool`(1)
- `delete-connector-connection-pool`(1)

Connector Resources

A connector resource provides an application with a connection to an Enterprise Information System (EIS). Every connector resource is associated with a connection pool. To view, create, edit, or delete connector resources, click Resources > Connectors > Connector Resources in the Admin Console. Alternatively, you can use the following `asadmin` commands to create and delete connection resources:

- `create-connector-resource`(1)
- `delete-connector-resource`(1)

Administered Object Resources

An administered object provides an application with specialized functionality, such as access to a parser specific to the resource adapter and its associated EIS. To view, create, edit, or delete administered objects, click Resources > Connectors > Admin Object Resources in the Admin Console.

- `create-admin-object`(1)
- `delete-admin-object-1`(1)

◆ ◆ ◆ CHAPTER 8

J2EE Containers

This chapter explains how to configure the J2EE containers included in the server. This chapter contains following sections:

Types of J2EE Containers

J2EE containers provide runtime support for J2EE application components. J2EE application components use the protocols and methods of the container to access other application components and services provided by the server. The Application Server provides an application client container, an applet container, a Web container, and an EJB container. For a diagram that shows the containers, see the section "Application Server Architecture" on page 26.

The Web Container

The Web Container is a J2EE container that hosts web applications. The web container extends the web server functionality by providing developers the environment to run servlets and JavaServer Pages (JSP files).

The EJB Container

Enterprise beans (EJB components) are Java programming language server components that contain business logic. The EJB container provides local and remote access to enterprise beans.

There are three types of enterprise beans: session beans, entity beans, and message-driven beans. Session beans represent transient objects and processes and typically are used by a single

client. Entity beans represent persistent data, typically maintained in a database. Message-driven beans are used to pass messages asynchronously to application modules and services.

The container is responsible for creating the enterprise bean, binding the enterprise bean to the naming service so other application components can access the enterprise bean, ensuring only authorized clients have access to the enterprise bean's methods, saving the bean's state to persistent storage, caching the state of the bean, and activating or passivating the bean when necessary.

Configuring J2EE Containers

Configuring the General Web Container Settings

In this release, there are no container-wide settings for the Web container in the Administration Console.

Configuring Web Container Sessions

This section describes the HTTP session settings in the Web container. HTTP sessions are unique web sessions that have their state data written to a persistent store.

Configuring Session Timeout Value

Use the Administration Console to set the HTTP session timeout value. The session timeout value represents the duration for which an HTTP session is valid.

In the Administration Console, go to Configuration > Web Container > Session Properties. In the Session Timeout field, enter the number of seconds that a session is valid.

For detailed instructions on setting the session timeout value, Click Help in the Administration Console.

Configuring Manager Properties

The session manager provides the means to configure how sessions are created and destroyed, where session state is stored, and the maximum number of sessions.

To change the session manager settings in the Administration Console, go to Configuration > Web Container > Manager Properties.

- Select the instance to configure:

  To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.

  To configure the default settings for all instances, select the `default-config` node.

In the Manager Properties tab, set the following properties:

- Reap Interval value. The Reap Interval field is the number of seconds before the inactive session data is deleted from the store.
- Max Sessions value. The Max Sessions field is the maximum number of sessions allowed.
- Set the Session Filename value. The Session Filename field is the file that contains the session data.
- Session ID Generator Classname value.

  The Session ID Generator Classname field allows you to specify a custom class for generating unique session IDs. Only one session ID generator class per server instance is permitted, and all instances in a cluster must use the same session ID generator to prevent session key collision.

  Custom session ID generator classes must implement the `com.sun.enterprise.util.uuid.UuidGenerator` interface:

```
package com.sun.enterprise.util.uuid;

public interface UuidGenerator {

    public String generateUuid();
    public String generateUuid(Object obj);  //obj is the session object
}
```

  The class must be in the Application Server classpath.

For detailed instructions on setting the manager properties, Click Help in the Administration Console.

Configuring Store Properties

To specify where the session store data will be saved, in the Administration Console, go to Configuration > Web Container > Store Properties.

- Select the instance to configure:

To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.

To configure the default settings for all instances, select the `default-config` node.

- Set the Reap Interval field

  The Reap Interval field is the number of seconds before the inactive session data is deleted from the store.
- Specify the directory where session data will be stored.

For detailed instructions on setting the session store properties, Click Help in the Administration Console.

Configuring the Virtual Server Settings

When you install Application Server, a default virtual server for the Application Server instance is created. The default `docroot` for this virtual server is created at *instance-dir*`domains/domain1/docroot`, which is synchronized to the *instance_name*`/docroot`. A virtual server is created for each additional Application Server instance you create.

Configuring the General EJB Settings

This section describes the following settings, which apply to all enterprise bean containers on the server:

- "Session Store Location" on page 84
- "Configuring EJB Pool Settings" on page 85
- "Configuring EJB Cache Settings" on page 85

To override the defaults on a per-container basis, adjust the values in the enterprise bean's `sun-ejb-jar.xml` file. For details, see the *Application Server Developer's Guide*.

Session Store Location

The Session Store Location field specifies the directory where passivated beans and persisted HTTP sessions are stored on the file system.

Passivated beans are enterprise beans that have had their state written to a file on the file system. Passivated beans typically have been idle for a certain period of time, and are not currently being accessed by clients.

Similar to passivated beans, persisted HTTP sessions are individual web sessions that have had their state written to a file on the file system.

The Commit Option field specifies how the container caches passivated entity bean instances between transactions.

Option B caches entity bean instances between transactions, and is selected by default. Option C disables caching.

Configuring EJB Pool Settings

The container maintains a pool of enterprise beans in order to respond to client requests without the performance hit that results from creating the beans. These settings only apply to stateless session beans and entity beans.

If you experience performance problems in an application that uses deployed enterprise beans, creating a pool, or increasing the number of beans maintained by an existing pool, can help increase the application's performance.

By default, the container maintains a pool of enterprise beans.

Configuring EJB Cache Settings

The container maintains a cache of enterprise bean data for the most used enterprise beans. This allows the container to respond more quickly to requests from other application modules for data from the enterprise beans. This section applies only to stateful session beans and entity beans.

Cached enterprise beans are in one of three states: active, idle, or passivated. An active enterprise bean is currently being accessed by clients. An idle enterprise bean's data is currently in the cache, but no clients are accessing the bean. A passivated bean's data is temporarily stored, and read back into the cache if a client requests the bean.

Configuring the Message-Driven Bean Settings

The pool for message-driven beans is similar to the pool for session beans described in "Configuring EJB Pool Settings" on page 85. By default, the container maintains a pool of message—driven beans.

To adjust the configuration of this pool:

Configuring the EJB Timer Service Settings

The timer service is a persistent and transactional notification service provided by the enterprise bean container used to schedule notifications or events used by enterprise beans. All enterprise beans except stateful session beans can receive notifications from the timer service. Timers set by the service are not destroyed when the server is shut down or restarted.

◆ ◆ ◆ CHAPTER 9

Configuring Security

Security is about protecting data: how to prevent unauthorized access or damage to it in storage or transit. The Application Server; has a dynamic, extensible security architecture based on the J2EE standard. Built in security features include cryptography, authentication and authorization, and public key infrastructure. The Application Server is built on the Java security model, which uses a sandbox where applications can run safely, without potential risk to systems or users. The following topics are discussed:

Understanding Application and System Security

Broadly, there are two kinds of application security:

- In *programmatic security*, application code written by the developer handles security chores. As an administrator, you don't have any control over this mechanism. Generally, programmatic security is discouraged since it hard-codes security configurations in the application instead of managing it through the J2EE containers.
- In *declarative security*, the container (the Application Server) handles security through an application's deployment descriptors. You can control declarative security by editing deployment descriptors directly or with a tool such as `deploytool`. Because deployment descriptors can change after an application is developed, declarative security allows for more flexibility.

In addition to application security, there is also *system security*, which affects all the applications on an Application Server system.

Programmatic security is controlled by the application developer, so this document does not discuss it; declarative security is somewhat less so, and this document touches on it occasionally. This document is intended primarily for system administrators, and so focuses on system security.

Tools for Managing Security

The Application Server provides the following tools for managing security:

- Administration Console, a browser-based tool used to configure security for the entire server, to manage users, groups, and realms, and to perform other system-wide security tasks. For a general introduction to the Administration Console, see "Tools for Administration" on page 28. For an overview of the security tasks you can perform with the Administration Console, see "Managing Security With the Administration Console" on page 100.
- `asadmin`, a command-line tool that performs many of the same tasks as the Administration Console. You may be able to do some things with `asadmin` that you cannot do with Administration Console. You perform `asadmin` commands from either a command prompt or from a script, to automate repetitive tasks. For a general introduction to `asadmin`, see "Tools for Administration" on page 28.
- `deploytool`, a graphical packaging and deployment tool for editing application deployment descriptors to control individual application's security. Because `deploytool` is intended for application developers, this document does not describe its use in detail. For instructions on using `deploytool`, see the tool's online help and *The J2EE 1.4 Tutorial* at `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`.

The Java 2 Platform, Standard Edition (J2SE) provides two tools for managing security:

- `keytool`, a command-line utility for managing digital certificates and key pairs. Use `keytool` to manage users in the `certificate` realm.
- `policytool`, a graphical utility for managing system-wide Java security policies. As an administrator, you will rarely need to use `policytool`.

For more information on using `keytool`, `policytool`, and other Java security tools, see *Java 2 SDK Tools and Utilities* at `http://java.sun.com/j2se/1.4.2/docs/tooldocs/tools.html#security`.

In the Enterprise Edition, two other tools that implement Network Security Services (NSS) are available for managing security. For more information on NSS, go to `http://www.mozilla.org/projects/security/pki/nss/`. The tools for managing security include the following:

- `certutil`, a command-line utility for managing certificates and key databases.
- `pk12util`, a command-line utility used to import and export keys and certificates between the certificate/key databases and files in PKCS12 format.

For more information on using `certutil`, `pk12util`, and other NSS security tools, see *NSS Security Tools* at `http://www.mozilla.org/projects/security/pki/nss/tools`.

Managing Security of Passwords

In this release of the Application Server, the file `domain.xml`, which contains the specifications for a particular domain, initially contains the password of the Sun Java SystemMessage Queue broker in clear text. The element in the `domain.xml` file that contains this password is the `admin-password` attribute of the `jms-host` element. Because this password is not changeable at installation time, it is not a significant security impact.

However, use the Administration Console to add users and resources and assign passwords to these users and resources. Some of these passwords are written to the `domain.xml` file in clear text, for example, passwords for accessing a database. Having these passwords in clear text in the `domain.xml` file can present a security hazard. You can encrypt any password in `domain.xml`, including the `admin-password` attribute or a database password. Instructions for managing the security passwords is included in the following topics:

Encrypting a Password in the `domain.xml` File

To encrypt a password in the `domain.xml` file. Follow these steps:

1. From the directory where the `domain.xml` file resides (*domain-dir*`/config` by default), run the following `asadmin` command:

   ```
   asadmin create-password-alias --user admin alias-name
   ```

   For example,

   ```
   asadmin create-password-alias --user admin jms-password
   ```

   A password prompt appears (`admin` in this case). Refer to the man pages for the `create-password-alias`, `list-password-aliases`, `delete-password-alias` commands for more information.
2. Remove and replace the password in `domain.xml`. This is accomplished using the `asadmin set` command. An example of using the `set` command for this purpose is as follows:

```
asadmin set --user admin server.jms-service.jms-host.
default_JMS_host.admin-password='${ALIAS=jms-password}'
```

Note – Enclose the alias password in single quotes as shown in the example.

3. Restart the Application Server for the relevant domain.

Protecting Files with Encoded Passwords

Some files contain encoded passwords that need protecting using file system permissions. These files include the following:

- *domain-dir*`/master-password`

  This file contains the encoded master password and should be protected with file system permissions 600.
- Any password file created to pass as an argument using the `--passwordfile` argument to `asadmin` should be protected with file system permissions 600.

Changing the Master Password

The master password (MP) is an overall shared password. It is never used for authentication and is never transmitted over the network. This password is the choke point for overall security; the user can choose to enter it manually when required, or obscure it in a file. It is the most sensitive piece of data in the system. The user can force prompting for the MP by removing this file. When the master password is changed, it is re-saved in the master-password keystore, which is a Java JCEKS type keystore.

To change the master password, follow these steps:

1. Stop the Application Server for the domain. Use the `asadmin change-master-password` command, which prompts for the old and new passwords, then re-encrypts all dependent items. For example:

   ```
   asadmin change-master-password>
   Please enter the master password>
   Please enter the new master password>
   Please enter the the new master password again>
   ```
2. Restart the Application Server.

Caution – At this point in time, server instances that are running must not be started and running server instances must not be restarted until the SMP on their corresponding node agent has been changed. If a server instance is restarted before changing its SMP, it will fail to come up.

3. Stop each node agent and its related servers one at a time. Run the `asadmin change-master-password` command again, and then restart the node agent and its related servers.
4. Continue with the next node agent until all node agents have been addressed. In this way, a rolling change is accomplished.

Working with the Master Password and Keystores

The master password is the password for the secure keystore . When a new application server domain is created, a new self-signed certificate is generated and stored in the relevant keystore, which is locked using the master password. If the master password is not the default, the `start-domain` command prompts you for the master password. Once the correct master password is entered, the domain starts.

When a node agent associated with the domain is created, the node agent synchronizes the data with domain. While doing so, the keystore is also synchronized. Any server instance controlled by this node agent needs to open the keystore. Since the store is essentially identical to the store that was created by the domain creation process, it can only be opened by an identical master password. But the master password itself is never synchronized, meaning it is not transmitted to the node agent during the synchronization, but needs to be available with the node agent locally. This is why creation and/or starting of a node agent prompts you for the master password and you need to enter the same password that you entered while creating/starting the domain. If the master password is changed for a domain, you will have to perform the same step to change it at every node agent that is associated with this domain.

Changing the Admin Password

Encrypting the admin password is discussed in "Managing Security of Passwords" on page 89. Encrypting the admin password is strongly encouraged. If you want to change the admin password before encrypting it, use the `asadmin set` command. An example of using the `set` command for this purpose is as follows:

```
asadmin set --user admin server.jms-service.jms-host.default_JMS_host.admin-password=new_pwd
```

It is also possible to change the admin password using the Administration Console as in the following procedure.

To change the admin password using the Admin Console, select Configuration node >Instance to configure > Security node > Realms node > `admin-realm` node and edit the realm page as desired.

About Authentication and Authorization

Authentication and authorization are central concepts of application server security. The following topics are discussed related to authentication and authorization:

Authenticating Entities

Authentication is the way an entity (a user, an application, or a component) determines that another entity is who it claims to be. An entity uses *security credentials* to authenticate itself. The credentials may be a user name and password, a digital certificate, or something else.

Typically, authentication means a user logging in to an application with a user name and password; but it might also refer to an EJB providing security credentials when it requests a resource from the server. Usually, servers or applications require clients to authenticate; additionally, clients can require servers to authenticate themselves, too. When authentication is bidirectional, it is called mutual authentication.

When an entity tries to access a protected resource, the Application Server uses the authentication mechanism configured for that resource to determine whether to grant access. For example, a user can enter a user name and password in a Web browser, and if the application verifies those credentials, the user is authenticated. The user is associated with this authenticated security identity for the remainder of the session.

The Application Server supports four types of authentication, as outlined in "Authenticating Entities" on page 92. An application specifies the type of authentication it uses within its deployment descriptors. For more information on using `deploytool` to configure the authentication method for an application, see *The J2EE 1.4 Tutorial* at `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`.

TABLE 9–1 Application Server Authentication Methods

Authentication Method	Communication Protocol	Description	User Credential Encryption
Basic	HTTP (SSL optional)	Uses the server's built-in pop-up login dialog box.	None, unless using SSL.
Form-based	HTTP (SSL optional)	Application provides its own custom login and error pages.	None, unless using SSL.
Client Certificate	HTTPS (HTTP over SSL)	Server authenticates the client using a public key certificate.	SSL

Verifying Single Sign-On

Single sign-on enables multiple applications in one virtual server instance to share user authentication state. With single sign-on, a user who logs in to one application becomes implicitly logged in to other applications that require the same authentication information.

Single sign-on is based on groups. All Web applications whose deployment descriptor defines the same *group* and use the same authentication method (basic, form, digest, certificate) share single sign-on.

Single sign-on is enabled by default for virtual servers defined for the Application Server.

Authorizing Users

Once a user is authenticated, the level of *authorization* determines what operations can be performed. A user’s authorization is based on his *role*. For example, a human resources application may authorize managers to view personal employee information for all employees, but allow employees to view only their own personal information. For more on roles, see "Understanding Users, Groups, Roles, and Realms" on page 94.

Specifying JACC Providers

JACC (Java Authorization Contract for Containers) is part of the J2EE 1.4 specification that defines an interface for pluggable authorization providers. This enables the administrator to set up third-party plug-in modules to perform authorization.

By default, the Application Server provides a simple, file-based authorization engine that complies with the JACC specification. It is also possible to specify additional third-party JACC providers.

JACC providers use the Java Authentication and Authorization Service (JAAS) APIs. JAAS enables services to authenticate and enforce access controls upon users. It implements a Java technology version of the standard Pluggable Authentication Module (PAM) framework.

Auditing Authentication and Authorization Decisions

The Application Server can provide an audit trail of all authentication and authorization decisions through *audit modules*. The Application Server provides a default audit module, as well as the ability to customize the audit modules. For information on developing custom audit modules, see the Application Server *Developer's Guide Configuring an Audit Module* section.

Configuring Message Security

Message Security enables a server to perform end-to-end authentication of web service invocations and responses at the message layer. The Application Server implements message security using message security providers on the SOAP layer. The message security providers provide information such as the type of authentication that is required for the request and response messages. The types of authentication that are supported include the following:

- Sender authentication, including username-password authentication.
- Content authentication, including XML Digital Signatures.

Two message security providers are included with this release. The message security providers can be configured for authentication for the SOAP layer. The providers that can be configured include `ClientProvider` and `ServerProvider`.

Support for message layer security is integrated into the Application Server and its client containers in the form of (pluggable) authentication modules. By default, message layer security is disabled on the Application Server.

Message level security can be configured for the entire Application Server or for specific applications or methods. Configuring message security at the Application Server level is discussed in Chapter 10, "Configuring Message Security." Configuring message security at the application level is discussed in the *Developer's Guide* chapter titled *Securing Applications*.

Understanding Users, Groups, Roles, and Realms

The Application Server enforces its authentication and authorization policies upon the following entities:

- "Users" on page 95: An individual identity *defined in the Application Server*. In general, a user is a person, a software component such as an enterprise bean, or even a service. A user who has been authenticated is sometimes called a *principal*. Users are sometimes referred to as *subjects*.
- "Groups" on page 95: A set of users *defined in the Application Server*, classified by common traits.

- "Roles" on page 95: A named authorization level *defined by an application*. A role can be compared to a key that opens a lock. Many people might have a copy of the key. The lock doesn't care who seeks access, only that the right key is used.
- "Realms" on page 96: A repository containing user and group information and their associated security credentials. A realm is also called a *security policy domain*.

Note – Users and groups are designated for the entire Application Server, whereas each application defines its own roles. When the application is being packaged and deployed, the application specifies mappings between users/groups and roles, as illustrated in the following figure.

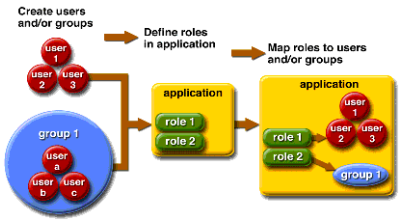

FIGURE 9–1 Role Mapping

Users

A *user* is an individual (or application program) identity that has been defined in the Application Server. A user can be associated with a group. The Application Server authentication service can govern users in multiple realms.

Groups

A *J2EE group* (or simply group) is a category of users classified by common traits, such as job title or customer profile. For example, users of an e-commerce application might belong to the `customer` group, but the big spenders would belong to the `preferred` group. Categorizing users into groups makes it easier to control the access of large numbers of users.

Roles

A *role* defines which applications and what parts of each application users can access and what they can do. In other words, roles determine users' authorization levels.

For example, in a personnel application all employees might have access to phone numbers and email addresses, but only managers would have access to salary information. The application might define at least two roles: `employee` and `manager`; only users in the `manager` role are allowed to view salary information.

A role is different from a user group in that a role defines a function in an application, while a group is a set of users who are related in some way. For example, in the personnel application there might be groups such as `full-time`, `part-time`, and `on-leave`, but users in all these groups would still be in the `employee` role.

Roles are defined in application deployment descriptors. In contrast, groups are defined for an entire server and realm. The application developer or deployer maps roles to one or more groups for each application in its deployment descriptor.

Realms

A *realm*, also called a *security policy domain* or *security domain*, is a scope over which the server defines and enforces a common security policy. In practical terms, a realm is a repository where the server stores user and group information.

The Application Server comes preconfigured with three realms: `file` (the initial default realm), `certificate`, and `admin-realm`. It is possible to also set up `ldap`, `solaris`, or custom realms. Applications can specify the realm to use in their deployment descriptor. If they do not specify a realm, the Application Server uses its default realm.

In the `file` realm, the server stores user credentials locally in a file named `keyfile`. You can use the Administration Console to manage users in the `file` realm. .

In the `certificate` realm, the server stores user credentials in a certificate database. When using the `certificate` realm, the server uses certificates with the HTTPS protocol to authenticate Web clients. For more information about certificates, see "Introduction to Certificates and SSL" on page 97.

The `admin-realm` is also a `FileRealm` and stores administrator user credentials locally in a file named `admin-keyfile`. Use the Administration Console to manage users in this realm in the same way you manage users in the `file` realm.

In the `ldap` realm the server gets user credentials from a Lightweight Directory Access Protocol (LDAP) server such as the Sun Java System Directory Server. LDAP is a protocol for enabling anyone to locate organizations, individuals, and other resources such as files and devices in a network, whether on the public Internet or on a corporate intranet. Consult your LDAP server documentation for information on managing users and groups in the `ldap` realm.

In the `solaris` realm the server gets user credentials from the Solaris operating system. This realm is supported on the Solaris 9 OS and later. Consult your Solaris documentation for information on managing users and groups in the `solaris` realm.

A custom realm is any other repository of user credentials, such as a relational database or third-party component. For more information, see the Admin Console online help.

Introduction to Certificates and SSL

The following topics are discussed in this section:

About Digital Certificates

Digital certificates (or simply certificates) are electronic files that uniquely identify people and resources on the Internet. Certificates also enable secure, confidential communication between two entities.

There are different kinds of certificates, such as personal certificates, used by individuals, and server certificates, used to establish secure sessions between the server and clients through secure sockets layer (SSL) technology. For more information on SSL, see "About Secure Sockets Layer" on page 98.

Certificates are based on *public key cryptography*, which uses pairs of digital *keys* (very long numbers) to *encrypt*, or encode, information so it can be read only by its intended recipient. The recipient then *decrypts* (decodes) the information to read it.

A key pair contains a public key and a private key. The owner distributes the public key and makes it available to anyone. But the owner never distributes the private key; it is always kept secret. Because the keys are mathematically related, data encrypted with one key can be decrypted only with the other key in the pair.

A certificate is like a passport: it identifies the holder and provides other important information. Certificates are issued by a trusted third party called a *Certification Authority* (CA). The CA is analogous to passport office: it validates the certificate holder's identity and signs the certificate so that it cannot be forged or tampered with. Once a CA has signed a certificate, the holder can present it as proof of identity and to establish encrypted, confidential communications.

Most importantly, a certificate binds the owner's public key to the owner's identity. Like a passport binds a photograph to personal information about its holder, a certificate binds a public key to information about its owner.

In addition to the public key, a certificate typically includes information such as:

- The name of the holder and other identification, such as the URL of the Web server using the certificate, or an individual's email address.
- The name of the CA that issued the certificate.

- An expiration date.

Digital Certificates are governed by the technical specifications of the X.509 format. To verify the identity of a user in the `certificate` realm, the authentication service verifies an X.509 certificate, using the common name field of the X.509 certificate as the principal name.

About Certificate Chains

Web browsers are preconfigured with a set of *root* CA certificates that the browser automatically trusts. Any certificates from elsewhere must come with a *certificate chain* to verify their validity. A certificate chain is series of certificates issued by successive CA certificates, eventually ending in a root CA certificate.

When a certificate is first generated, it is a *self-signed* certificate. A self-signed certificate is one for which the issuer (signer) is the same as the subject (the entity whose public key is being authenticated by the certificate). When the owner sends a certificate signing request (CSR) to a CA, then imports the response, the self-signed certificate is replaced by a chain of certificates. At the bottom of the chain is the certificate (reply) issued by the CA authenticating the subject's public key. The next certificate in the chain is one that authenticates the CA's public key. Usually, this is a self-signed certificate (that is, a certificate from the CA authenticating its own public key) and the last certificate in the chain.

In other cases, the CA can return a chain of certificates. In this case, the bottom certificate in the chain is the same (a certificate signed by the CA, authenticating the public key of the key entry), but the second certificate in the chain is a certificate signed by a different CA, authenticating the public key of the CA to which you sent the CSR. Then, the next certificate in the chain is a certificate authenticating the second CA's key, and so on, until a self-signed *root* certificate is reached. Each certificate in the chain (after the first) thus authenticates the public key of the signer of the previous certificate in the chain.

About Secure Sockets Layer

Secure Sockets Layer (SSL) is the most popular standard for securing Internet communications and transactions. Web applications use HTTPS (HTTP over SSL), which uses digital certificates to ensure secure, confidential communications between server and clients. In an SSL connection, both the client and the server encrypt data before sending it, then decrypt it upon receipt.

When a Web browser (client) wants to connect to a secure site, an *SSL handshake* happens:

- The browser sends a message over the network requesting a secure session (typically, by requesting a URL that begins with `https` instead of `http`).
- The server responds by sending its certificate (including its public key).
- The browser verifies that the server's certificate is valid and is signed by a CA whose certificate is in the browser's database (and who is trusted). It also verifies that the CA certificate has not expired.

- If the certificate is valid, the browser generates a one time, unique *session key* and encrypts it with the server's public key. The browser then sends the encrypted session key to the server so that they both have a copy.
- The server decrypts the message using its private key and recovers the session key.

After the handshake, the client has verified the identity of the Web site, and only the client and the Web server have a copy of the session key. From this point forward, the client and the server use the session key to encrypt all their communications with each other. Thus, their communications are ensured to be secure.

The newest version of the SSL standard is called TLS (Transport Layer Security). The Application Server supports the Secure Sockets Layer (SSL) 3.0 and the Transport Layer Security (TLS) 1.0 encryption protocols.

To use SSL, the Application Server must have a certificate for each external interface, or IP address, that accepts secure connections. The HTTPS service of most Web servers will not run unless a digital certificate has been installed. Use the procedure described in "Generating a Certificate Using the `keytool` Utility" on page 105 to set up a digital certificate that your Web server can use for SSL.

About Ciphers

A *cipher* is a cryptographic algorithm used for encryption or decryption. SSL and TLS protocols support a variety of ciphers used to authenticate the server and client to each other, transmit certificates, and establish session keys.

Some ciphers are stronger and more secure than others. Clients and servers can support different cipher suites. Choose ciphers from the SSL3 and TLS protocols. During a secure connection, the client and the server agree to use the strongest cipher they both have enabled for communication, so it is usually sufficient to enable all ciphers.

Using Name-based Virtual Hosts

Using name-based virtual hosts for a secure application can be problematic. This is a design limitation of the SSL protocol itself. The SSL handshake, where the client browser accepts the server certificate, must occur before the HTTP request is accessed. As a result, the request information containing the virtual host name cannot be determined prior to authentication, and it is therefore not possible to assign multiple certificates to a single IP address.

If all virtual hosts on a single IP address need to authenticate against the same certificate, the addition of multiple virtual hosts probably will not interfere with normal SSL operations on the server. Be aware, however, that most browsers will compare the server's domain name against the domain name listed in the certificate, if any (applicable primarily to official, CA-signed certificates). If the domain names do not match, these browsers display a warning. In general, only address-based virtual hosts are commonly used with SSL in a production environment.

About Firewalls

A *firewall* controls the flow of data between two or more networks, and manages the links between the networks. A firewall can consist of both hardware and software elements. This section describes some common firewall architectures and their configuration. The information here pertains primarily to the Application Server. For details about a specific firewall technology, refer to the documentation from your firewall vendor.

In general, configure the firewalls so that clients can access the necessary TCP/IP ports. For example, if the HTTP listener is operating on port 8080, configure the firewall to allow HTTP requests on port 8080 only. Likewise, if HTTPS requests are setup for port 8181, you must configure the firewalls to allow HTTPS requests on port 8181.

If direct Remote Method Invocations over Internet Inter-ORB Protocol (RMI-IIOP) access from the Internet to EJB modules are required, open the RMI-IIOP listener port as well, but this is strongly discouraged because it creates security risks.

In double firewall architecture, you must configure the outer firewall to allow for HTTP and HTTPS transactions. You must configure the inner firewall to allow the HTTP server plug-in to communicate with the Application Server behind the firewall.

Managing Security With the Administration Console

The Administration Console provides the means to manage the following aspects of security:

Server Security Settings

On the Security Settings page, set properties for the entire server, including specifying the default realm, the anonymous role, and the default principal user name and password.

Realms and file Realm Users

The concept of realms is introduced in "Understanding Users, Groups, Roles, and Realms" on page 94.

- Create a new realm
- Delete an existing realm
- Modify the configuration of an existing realm
- Add, modify, and delete users in the `file` realm
- Set the default realm

JACC Providers

JACC providers is introduced in "Specifying JACC Providers" on page 93. Use the Administration Console to perform the following tasks:

- Add a new JACC provider
- Delete or modify an existing JACC provider

Audit Modules

Audit modules is introduced in "Auditing Authentication and Authorization Decisions" on page 94. Auditing is the method by which significant events, such as errors or security breaches, are recorded for subsequent examination. All authentication events are logged to the Application Server logs. A complete access log provides a sequential trail of Application Server access events.

Use the Administration Console to perform the following tasks:

- Add a new audit module
- Delete or modify an existing audit module

Message Security

The concept of message security is introduced in "Configuring Message Security" on page 94. Use the Administration Console to perform the following tasks:

- Enable message security
- Configure a message security provider
- Delete or configure an existing message security configuration or provider

See the Administration Console online help for details on these tasks.

HTTP and IIOP Listener Security

Each virtual server in the HTTP service provides network connections through one or more *HTTP listeners*.

The Application Server supports CORBA (Common Object Request Broker Architecture) objects, which use the Internet Inter-Orb Protocol (IIOP) to communicate across the network. An *IIOP listener* accepts incoming connections from remote clients of EJB components and from other CORBA-based clients. For general information on IIOP listeners, see "IIOP Listeners" on page 140.

With the Administration Console, perform the following tasks:

- Create a new HTTP or IIOP listener, and specify the security it uses.
- Modify the security settings for an existing HTTP or IIOP listener.

Admin Service Security

The Admin Service determines whether the server instance is a regular instance, a domain administration server (DAS), or a combination. Use the Admin Service to configure a JSR-160 compliant remote JMX connector, which handles communication between the domain administration server and the node agents, which manage server instances on a host machine, for remote server instances.

With the Administration Console, perform the following tasks:

- Manage the Admin Service
- Edit the JMX connector
- Modify the security settings of the JMX connector

Security Maps

Use the Administration Console to perform the following security mapping tasks:

- Add a security map to an existing connector connection pool
- Delete or configure an existing security map

Working with Certificates and SSL

About Certificate Files

Installation of the Application Server generates a digital certificate in JSSE (Java Secure Socket Extension) or NSS (Network Security Services) format suitable for internal testing. By default, the Application Server stores its certificate information in a certificate database in the *domain-dir*`/config` directory:

- **Keystore file**, `key3.db`, contains the Application Server's certificate, including its private key. The keystore file is protected with a password. Change the password using the `asadmin change-master-password` command. For more information about `certutil`, read "Using the `certutil` Utility" on page 108.

  Each keystore entry has a unique alias. After installation, the Application Server keystore has a single entry with alias `s1as`.
- **Truststore file**, `cert8.db`, contains the Application Server's trusted certificates, including public keys for other entities. For a trusted certificate, the server has confirmed that the public key in the certificate belongs to the certificate's owner. Trusted certificates generally include those of certification authorities (CAs).

  In the Platform Edition, on the server side, the Application Server uses the JSSE format, which uses `keytool` to manage certificates and key stores. In the Enterprise Edition, on the server side, the Application Server uses NSS, which uses `certutil` to manage the NSS database which stores private keys and certificates. In both editions, the client side (appclient or stand-alone), uses the JSSE format.

  By default, the Application Server is configured with a keystore and truststore that will work with the example applications and for development purposes. For production purposes, you may wish to change the certificate alias, add other certificates to the truststore, or change the name and/or location of the keystore and truststore files.

Changing the Location of Certificate Files

The keystore and truststore files provided for development are stored in the *domain-dir*`/config` directory.

Use the Admin Console to expand the `server-config` node > JVM Settings >JVM Options tab to add or modify the value field for the new location of the certificate files.

`-Dcom.sun.appserv.nss.db=${com.sun.aas.instanceRoot}/`*NSS-database-directory*

where *NSS-database-directory* is the location of the NSS database.

Using Java Secure Socket Extension (JSSE) Tools

Use `keytool` to set up and work with JSSE (Java Secure Socket Extension) digital certificates. In both the Platform Edition and Enterprise Edition, the client side (appclient or stand-alone) uses the JSSE format.

The J2SE SDK ships with `keytool`, which enables the administrator to administer public/private key pairs and associated certificates. It also enables users to cache the public keys (in the form of certificates) of their communicating peers.

To run `keytool`, the shell environment must be configured so that the J2SE `/bin` directory is in the path, or the full path to the tool must be present on the command line. For more information on `keytool`, see the `keytool` documentation at `http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html`.

Using the keytool Utility

The following examples demonstrate usage related to certificate handling using JSSE tools:

- Create a self-signed certificate in a keystore of type JKS using an RSA key algorithm. RSA is public-key encryption technology developed by RSA Data Security, Inc. The acronym stands for Rivest, Shamir, and Adelman, the inventors of the technology.

  ```
  keytool  -genkey -noprompt -trustcacerts -keyalg RSA -alias ${cert.alias}
  -dname  ${dn.name} -keypass ${key.pass} -keystore ${keystore.file}
  -storepass ${keystore.pass}
  ```

  Another example of creating a certificate is shown in "Generating a Certificate Using the `keytool` Utility" on page 105.

- Create a self-signed certificate in a keystore of type JKS using the default key algorithm.

  ```
  keytool -genkey -noprompt -trustcacerts -alias ${cert.alias} -dname
  ${dn.name} -keypass ${key.pass} -keystore ${keystore.file} -storepass
  ${keystore.pass}
  ```

  An example of signing a certificate is shown in "Signing a Digital Certificate Using the `keytool` Utility" on page 106

- Display available certificates from a keystore of type JKS.

  ```
  keytool -list -v  -keystore ${keystore.file} -storepass ${keystore.pass}
  ```

- Display certificate information from a keystore of type JKS.

  ```
  keytool  -list -v  -alias ${cert.alias} -keystore ${keystore.file}
  -storepass ${keystore.pass}
  ```

- Import an RFC/text-formatted certificate into a JKS store. Certificates are often stored using the printable encoding format defined by the Internet RFC (Request for Comments) 1421 standard instead of their binary encoding. This certificate format, also known as *Base 64 encoding*, facilitates exporting certificates to other applications by email or through some other mechanism.

  ```
  keytool -import -noprompt -trustcacerts -alias ${cert.alias} -file
  ${cert.file} -keystore ${keystore.file} -storepass ${keystore.pass}
  ```

- Export a certificate from a keystore of type JKS in PKCS7 format. The reply format defined by the Public Key Cryptography Standards #7, Cryptographic Message Syntax Standard, includes the supporting certificate chain in addition to the issued certificate.

  ```
  keytool  -export -noprompt  -alias ${cert.alias} -file ${cert.file}
  -keystore ${keystore.file} -storepass ${keystore.pass}
  ```

- Export a certificate from a keystore of type JKS in RFC/text format.

  ```
  keytool  -export -noprompt -rfc  -alias ${cert.alias} -file
  ${cert.file} -keystore ${keystore.file} -storepass ${keystore.pass}
  ```

- Delete a certificate from a keystore of type JKS.

  ```
  keytool  -delete -noprompt -alias ${cert.alias}  -keystore ${keystore.file}
  -storepass ${keystore.pass}
  ```

  Another example of deleting a certificate from a keystore is shown in "Deleting a Certificate Using the `keytool` Utility" on page 107

Generating a Certificate Using the `keytool` Utility

Use `keytool` to generate, import, and export certificates. By default, `keytool` creates a keystore file in the directory where it is run.

1. Change to the directory where the certificate is to be run.

   Always generate the certificate in the directory containing the keystore and truststore files, by default *domain-dir*`/config`. For information on changing the location of these files, see "Changing the Location of Certificate Files" on page 103.

2. Enter the following `keytool` command to generate the certificate in the keystore file, `keystore.jks`:

   ```
   keytool -genkey -alias keyAlias-keyalg RSA
    -keypass changeit
    -storepass changeit
   -keystore keystore.jks
   ```

   Use any unique name as your *keyAlias*. If you have changed the keystore or private key password from their default, then substitute the new password for `changeit` in the above command.

   A prompt appears that asks for your name, organization, and other information that `keytool` uses to generate the certificate.

3. Enter the following `keytool` command to export the generated certificate to the file `server.cer` (or `client.cer` if you prefer):

   ```
   keytool -export -alias keyAlias-storepass changeit
    -file server.cer
    -keystore keystore.jks
   ```

4. If a certificate signed by a certificate authority is required, see "Signing a Digital Certificate Using the `keytool` Utility" on page 106.
5. To create the truststore file `cacerts.jks` and add the certificate to the truststore, enter the following `keytool` command:

```
keytool -import -v -trustcacerts
-alias keyAlias
 -file server.cer
-keystore cacerts.jks
 -keypass changeit
```

6. If you have changed the keystore or private key password from their default, then substitute the new password for `changeit` in the above command.

   The tool displays information about the certificate and prompts whether you want to trust the certificate.
7. Type `yes`, then press Enter.

   Then `keytool` displays something like this:

```
Certificate was added to keystore
[Saving cacerts.jks]
```

8. Restart the Application Server.

Signing a Digital Certificate Using the `keytool` Utility

After creating a digital certificate, the owner must sign it to prevent forgery. E-commerce sites, or those for which authentication of identity is important can purchase a certificate from a well-known Certificate Authority (CA). If authentication is not a concern, for example if private secure communications is all that is required, save the time and expense involved in obtaining a CA certificate and use a self-signed certificate.

1. Follow the instructions on the CA's Web site for generating certificate key pairs.
2. Download the generated certificate key pair.

   Save the certificate in the directory containing the keystore and truststore files, by default *domain-dir*`/config` directory. See "Changing the Location of Certificate Files" on page 103.
3. In your shell, change to the directory containing the certificate.
4. Use `keytool` to import the certificate into the local keystore and, if necessary, the local truststore.

```
keytool -import -v -trustcacerts
-alias keyAlias
 -file server.cer
-keystore cacerts.jks
 -keypass changeit
-storepass changeit
```

If the keystore or private key password is not the default password, then substitute the new password for `changeit` in the above command.

5. Restart the Application Server.

Deleting a Certificate Using the `keytool` Utility

To delete an existing certificate, use the `keytool -delete` command, for example:

```
keytool -delete
 -alias keyAlias
 -keystore keystore-name
 -storepass password
```

Using Network Security Services (NSS) Tools

In the Enterprise Edition, use Network Security Services (NSS) digital certificates on the server-side to manage the database that stores private keys and certificates. For the client side (appclient or stand-alone), use the JSSE format as discussed in "Using Java Secure Socket Extension (JSSE) Tools" on page 103.

The tools for managing security with Network Security Services (NSS) include the following:

- `certutil`, a command-line utility for managing certificates and key databases. Some examples using the `certutil` utility are shown in "Using the `certutil` Utility" on page 108.
- `pk12util`, a command-line utility used to import and export keys and certificates between the certificate/key databases and files in PKCS12 format. Some examples using the `pk12util` utility are shown in "Importing and Exporting Certificates Using the `pk12util` Utility" on page 109.
- `modutil`, a command-line utility for managing PKCS #11 module information within `secmod.db` files or within hardware tokens. Some examples using the `modutil` utility are shown in "Adding and Deleting PKCS11 Modules using `modutil`" on page 110.

The tools are located in the *install-dir*`/lib/` directory. The following environment variables are used to point to the location of the NSS security tools:

- `LD_LIBRARY_PATH =${`*install-dir*`}/lib`
- `${`*os.nss.path*`}`

In the examples, the certificate common name (CN) is the name of the client or server. The CN is also used during SSL handshake for comparing the certificate name and the host name from which it originates. If the certificate name and the host name do not match, warnings or exceptions are generated during SSL handshake. In some examples, the certificate common name `CN=localhost` is used for convenience so that all users can use that certificate instead of creating a new one with their real host name.

The examples in the following sections demonstrate usage related to certificate handling using NSS tools:

Using the `certutil` Utility

Before running `certutil`, make sure that `LD_LIBRARY_PATH` points to the location of the libraries required for this utility to run. This location can be identified from the value of `AS_NSS_LIB` in `asenv.conf` (product wide configuration file).

The certificate database tool, `certutil`, is an NSS command-line utility that can create and modify the Netscape Communicator `cert8.db` and `key3.db` database files. It can also list, generate, modify, or delete certificates within the `cert8.db` file and create or change the password, generate new public and private key pairs, display the contents of the key database, or delete key pairs within the `key3.db` file.

The key and certificate management process generally begins with creating keys in the key database, then generating and managing certificates in the certificate database. The following document discusses certificate and key database management with NSS, including the syntax for the `certutil` utility: `http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html`.

Each of the items in the list below gives an example using NSS and JSSE security tools to create and/or manage certificates.

- Generate a self-signed server and client certificate. In this example, the CN must be of the form `hostname.domain.[com|org|net|...]`.

  In this example, *domain-dir*`/config`. The `serverseed.txt` and `clientseed.txt` files can contain any random text. This random text will be used for generating the key pair.

```
certutil -S -n $SERVER_CERT_NAME -x -t "u,u,u"
 -s "CN=$HOSTNAME.$HOSTDOMAIN, OU=Java Software, O=Sun Microsystems Inc.,
    L=Santa Clara, ST=CA, C=US"
 -m 25001 -o  $CERT_DB_DIR/Server.crt
 -d $CERT_DB_DIR -f passfile &lt;$CERT_UTIL_DIR/serverseed.txt
```

  Generate the client certificate. This certificate is also a self-signed certificate.

```
certutil -S -n $CLIENT_CERT_NAME  -x -t "u,u,u"
 -s "CN=MyClient, OU=Java Software, O=Sun Microsystems Inc.,
    L=Santa Clara, ST=CA, C=US"
 -m 25002 -o  $CERT_DB_DIR/Client.crt
 -d $CERT_DB_DIR -f passfile &lt;$CERT_UTIL_DIR/clientseed.txt
```

- Verify the certificates generated in the previous bullet.

```
certutil -V -u V -n $SERVER_CERT_NAME -d  $CERT_DB_DIR
certutil -V -u C -n $CLIENT_CERT_NAME -d  $CERT_DB_DIR
```

- Display available certificates.

```
certutil -L -d  $CERT_DB_DIR
```

- Import an RFC text-formatted certificate into an NSS certificate database.

```
certutil -A -a -n ${cert.nickname} -t ${cert.trust.options}
-f ${pass.file} -i ${cert.rfc.file}
-d ${admin.domain.dir}/${admin.domain}/config
```

- Export a certificate from an NSS certificate database in RFC format.

```
certutil -L -a -n ${cert.nickname} -f ${pass.file}
-d ${admin.domain.dir}/${admin.domain}/config > cert.rfc
```

- Delete a certificate from an NSS certificate database.

```
certutil -D -n ${cert.nickname} -f ${pass.file}
-d ${admin.domain.dir}/${admin.domain}/config
```

- Move a certificate from an NSS database to JKS format

```
certutil -L -a -n ${cert.nickname}
-d ${admin.domain.dir}/${admin.domain}/config > cert.rfc
keytool  -import -noprompt -trustcacerts -keystore ${keystore.file}
-storepass ${keystore.pass} -alias ${cert.alias} -file cert.rfc
```

Importing and Exporting Certificates Using the `pk12util` Utility

The command-line utility used to import and export keys and certificates between the certificate/key databases and files in PKCS12 format is `pk12util`. PKCS12 is Public-Key Cryptography Standards (PKCS) #12, Personal Information Exchange Syntax Standard. More description of the `pk12util` utility can be read at `http://www.mozilla.org/projects/security/pki/nss/tools/pk12util.html`.

- Import a PKCS12-formatted certificate into an NSS certificate database.

```
pk12util -i ${cert.pkcs12.file} -k ${certdb.pass.file}
-w ${cert.pass.file} -d ${admin.domain.dir}/${admin.domain}/config
```

- Import a PKCS12-formatted certificate into an NSS certificate database token module.

```
pk12util -i ${cert.pkcs12.file} -h ${token.name} -k ${certdb.pass.file}
-w ${cert.pass.file} -d ${admin.domain.dir}/${admin.domain}/config
```

- Export a certificate from an NSS certificate database in PKCS12 format.

```
pk12util -o -n ${cert.nickname} -k ${pass.file} -w${cert.pass.file}
-d ${admin.domain.dir}/${admin.domain}/config
```

- Export a certificate from an NSS certificate database token module in PKCS12 format (useful for hardware accelerator configuration).

```
pk12util -o -n ${cert.nickname} -h ${token.name} -k ${pass.file}
-w ${cert.pass.file} -d ${admin.domain.dir}/${admin.domain}/config
```

- Convert a PKCS12 certificate into JKS format (requires a Java source):

```
&lt;target name="convert-pkcs12-to-jks" depends="init-common">
   &lt;delete file="${jks.file}" failonerror="false"/>
   &lt;java classname="com.sun.enterprise.security.KeyTool">
      &lt;arg line="-pkcs12"/>
      &lt;arg line="-pkcsFile ${pkcs12.file}"/>
      &lt;arg line="-pkcsKeyStorePass ${pkcs12.pass}"/>
      &lt;arg line="-pkcsKeyPass ${pkcs12.pass}"/>
      &lt;arg line="-jksFile ${jks.file}"/>
      &lt;arg line="-jksKeyStorePass ${jks.pass}"/>
      &lt;classpath>
         &lt;pathelement path="${s1as.classpath}"/>
         &lt;pathelement path="${env.JAVA_HOME}/jre/lib/jsse.jar"/>
      &lt;/classpath>
   &lt;/java>
&lt;/target>
```

Adding and Deleting PKCS11 Modules using `modutil`

The *Security Module Database Tool*, `modutil`, is a command-line utility for managing PKCS #11 (Cryptographic Token Interface Standard) module information within `secmod.db` files or within hardware tokens. You can use the tool to add and delete PKCS #11 modules, change passwords, set defaults, list module contents, enable or disable slots, enable or disable FIPS-140-1 compliance, and assign default providers for cryptographic operations. This tool can also create `key3.db`, `cert7.db`, and `secmod.db` security database files. For more information on this tool, see `http://www.mozilla.org/projects/security/pki/nss/tools/modutil.html`.

- Add a new PKCS11 module or token.

```
modutil -add ${token.module.name} -nocertdb -force -mechanisms RSA:DSA:RC4:DES
-libfile ${SCA.lib.path} -dbdir ${admin.domain.dir}/${admin.domain}/config
```

- Delete a PKCS11 module from an NSS store.

```
modutil -delete ${token.module.name} -nocertdb -force -mechanisms RSA:DSA:RC4:DES
-libfile ${SCA.lib.path} -dbdir ${admin.domain.dir}/${admin.domain}/config
```

- List available token modules in an NSS store.

```
modutil -list  -dbdir ${admin.domain.dir}/${admin.domain}/config
```

Using Hardware Crypto Accelerator With Application Server

You can use hardware accelerator tokens to improve the cryptographic performance and to furnish a secure key storage facility. Additionally, you can provide end users with mobile secure key storage through smart cards.

Sun Java System Application Server 8.1 and 8.2 Standard Edition or Enterprise Edition when run on the Java 2 Platform, Standard Edition (J2SE platform) 5.0, supports the use of PKCS#11 tokens for SSL or TLS communications and Network Security Services (NSS) tools for managing keys and PKCS#11 tokens. This section describes how Application Server provides that support and walks you through the procedures for the related configurations.

J2SE 5.0 PKCS#11 providers can be easily integrated with the Application Server runtime. Through these providers, you can use hardware accelerators and other PKCS#11 tokens in Application Server to achieve fast performance and to protect the private key inherent in SSL or TLS communications.

This section contains the following topics:

About Configuring Hardware Crypto Accelerators

Sun Java System Application Server 8.1 and 8.2 Standard Edition or Enterprise Edition have been tested with Sun Crypto Accelerator 1000 (SCA-1000) and SCA-4000.

Application Server, when used in conjunction with J2SE 5.0, can communicate with PKCS#11 tokens. Packaged with Application Server are an NSS PKCS#11 token library (for the NSS Internal PKCS#11 Module, commonly known as the NSS soft token) and NSS command-line management tools. For more details, see "Using Network Security Services (NSS) Tools" on page 107.

Use the NSS tools to create keys and certificates on PKCS#11 tokens and J2SE PKCS#11 providers to access token keys and certificates at runtime. A PKCS#11 provider is a cryptographic service provider that acts as a wrapper around a native PKCS#11 library. A PKCS#11 token generally refers to all the hardware and software tokens with a native PKCS#11 interface. A hardware token is a PKCS#11 token implemented in physical devices, such as hardware accelerators and smart cards. A software token is a PKCS#11 token implemented entirely in software.

Note – If you run Application Server on the J2SE 1.4.x platform, only one PKCS#11 token, the NSS soft token, is supported.

For the Microsoft Windows environment, add the location of NSS libraries *AS_NSS* and the NSS tools directory, *AS_NSS_BIN* to the PATH environment variable. For simplicity, the procedures described in this section use UNIX commands only. You should replace the UNIX variables with the Windows variables, where appropriate.

Configuring the hardware crypto accelerators is divided into two main procedures:

- "Configuring PKCS#11 Tokens" on page 112
- "Configuring J2SE 5.0 PKCS#11 Providers" on page 115

Configuring PKCS#11 Tokens

This section describes how to configure PKCS#11 tokens with the NSS security tool `modutil`. Use the following procedure to configure a PKCS#11 token.

Enter the following command (all on one line):

```
modutil -dbdir AS_NSS_DB -nocertdb -force -add moduleName -libfile
 absolute_path_of_pkcs11_library -mechanisms list_of_security_mechanisms
```

where, *AS_NSS_DB* is the NSS database directory (same as `AS_DOMAIN_CONFIG` when you use the Domain Administration Server (DAS))

For example, to configure a hardware accelerator token, enter the following (all on one line):

```
modutil -dbdir AS_NSS_DB -nocertdb -force -add "Sun Crypto Accelerator" -libfile
 /opt/SUNWconn/crypto/lib/libpkcs11.so -mechanisms RSA:DSA:RC4:DES
```

The hardware accelerator in this example is a SCA–1000 cryptographic accelerator. The corresponding PKCS#11 library, by default, is located in `/opt/SUNWconn/crypto/lib/libpkcs11.so`.

The `mechanisms` must be a complete list of the cryptographic mechanisms that are available in the token. To use just a few of the available cryptographic mechanisms, see "Configuring J2SE 5.0 PKCS#11 Providers" on page 115. For a list of all supported mechanisms, see the `modutil` documentation on the NSS Security Tools site at `http://www.mozilla.org/projects/security/pki/nss/tools`.

The examples that follow assume that the token name specified at token installation time is `mytoken`.

To verify that the hardware accelerator is configured properly, enter the following command:

```
modutil -list -dbdir AS_NSS_DB
```

The standard output will look similar to the following:

```
Using database directory /var/opt/SUNWappserver/domains/domain1/config ...

Listing of PKCS#11 Modules
-----------------------------------------------------------
  1. NSS Internal PKCS#11 Module
         slots: 2 slots attached
        status: loaded

         slot: NSS Internal Cryptographic Services
        token: NSS Generic Crypto Services

         slot: NSS User Private Key and Certificate Services
        token: NSS Certificate DB

  2. Sun Crypto Accelerator
        library name: /opt/SUNWconn/crypto/lib/libpkcs11.so
         slots: 1 slot attached
        status: loaded

         slot: Sun Crypto Accelerator:mytoken
        token: mytoken
-----------------------------------------------------------
```

Managing Keys And Certificates

This section describes a few common procedures for creating and managing keys and certificates using `certutil` and `pk12util`. For details on `certutil` and `pk12util`, see "Using Network Security Services (NSS) Tools" on page 107 and documentation on the NSS Security Tools site at `http://www.mozilla.org/projects/security/pki/nss/tools`.

Note – By configuring a PKCS#11 provider in the `java.security` properties file (located in the *JAVA_HOME*`/jre/lib/security` directory of the Java runtime), you can also use the J2SE `keytool` utility to manage keys and certificates.

This section describes the following topics:

- "Listing Keys and Certificates " on page 114
- "Working With Private Keys and Certificates" on page 114

Listing Keys and Certificates

- To list the keys and certificates in the configured PKCS#11 tokens, run the following command:

  ```
  certutil -L -d AS_NSS_DB [-h tokenname]
  ```

  For example, to list the contents of the default NSS soft token, type:

  ```
  certutil -L -d AS_NSS_DB
  ```

  The standard output will be similar to the following:

  ```
  verisignc1g1                 T,c,c
  verisignc1g2                 T,c,c
  verisignc1g3                 T,c,c
  verisignc2g3                 T,c,c
  verisignsecureserver         T,c,c
  verisignc2g1                 T,c,c
  verisignc2g2                 T,c,c
  verisignc3g1                 T,c,c
  verisignc3g2                 T,c,c
  verisignc3g3                 T,c,c
  s1as                         u,u,u
  ```

  The output displays the name of the token in the left column and a set of three trust attributes in the right column. For Application Server certificates, it is usually `T,c,c`. Unlike the J2SE `java.security.KeyStore` API, which contains only one level of trust, the NSS technology contains several levels of trust. Application Server is primarily interested in the first trust attribute, which describes how this token uses SSL. For this attribute:

  `T` indicates that the Certificate Authority (CA) is trusted for issuing client certificates.
  `u` indicates that you can use the certificates (and keys) for authentication or signing.
  The attribute combination of `u,u,u` indicates that a private key exists in the database.

- To list the contents of the hardware token, `mytoken`, run the following command:

  ```
  certutil -L -d AS_NSS_DB -h mytoken
  ```

  You will be prompted for the password for the hardware token. The standard output is similar to the following:

  ```
  Enter Password or Pin for "mytoken":
  mytoken:Server-Cert                                          &#9;u,u,u
  ```

Working With Private Keys and Certificates

Use `certutil` to create self-signed certificates and to import or export certificates. To import or export private keys, use the `pk12util` utility. For more details, see "Using Network Security Services (NSS) Tools" on page 107

Caution – In Application Server, do not modify the NSS password directly with the NSS tools `certutil` and `modutil`. If you do so, security data in Application Server might be corrupted.

Configuring J2SE 5.0 PKCS#11 Providers

Application Server relies on J2SE PKCS#11 providers to access keys and certificates that are located in PKCS#11 tokens at runtime. By default, Application Server configures a J2SE PKCS#11 provider for the NSS soft token. This section describes how to override the default configuration for the J2SE PKCS#11 provider.

In Application Server, the following default PKCS#11 configuration parameters are generated for each PKCS#11 token.

- Configuration for the default NSS soft token:

```
name=internal
library=${com.sun.enterprise.nss.softokenLib}
nssArgs="configdir='${com.sun.appserv.nss.db}'
 certPrefix='' keyPrefix='' secmod='secmod.db'"
slot=2
omitInitialize = true
```

- Configuration for the SCA 1000 hardware accelerator:

```
name=HW1000
library=/opt/SUNWconn/crypto/lib/libpkcs11.so
slotListIndex=0
omitInitialize=true
```

These configurations conform to the syntax described in the Java PKCS#11 Reference Guide.

Note – The name parameter has no requirements other than that it must be unique. Certain older versions of J2SE 5.0 support alphanumeric characters only.

You can override the default configuration parameters by creating a custom configuration file. For example, you can explicitly disable the RSA Cipher and RSA Key Pair Generator in SCA–1000. For details on disabling the RSA Cipher and RSA Key Pair Generator, see `http://www.mozilla.org/projects/security/pki/nss/tools`.

To create a custom configuration file:

1. Create a configuration file called *install-dir*`/mypkcs11.cfg` with the following code and save the file.

```
name=HW1000
library=/opt/SUNWconn/crypto/lib/libpkcs11.so
slotListIndex=0
```

```
disabledMechanisms = {
&#9;CKM_RSA_PKCS
&#9;CKM_RSA_PKCS_KEY_PAIR_GEN
}
omitInitialize=true
```

2. Update the NSS database, if necessary. In this case, update the NSS database so that it will disable RSA.

   Run the following command :

   ```
   modutil -undefault "Sun Crypto Accelerator" -dbdir AS_NSS_DB -mechanisms RSA
   ```

   The name of the algorithm on the `mechanisms` list differs from the one in the default configuration. For a list of valid `mechanisms` in NSS, see the `modutil` documentation on the NSS Security Tools site at `http://www.mozilla.org/projects/security/pki/nss/tools`.

3. Update the server with this change by adding a property in the appropriate location, as follows:

   ```
   &lt;property name="mytoken" value="&InstallDir;/mypkcs11.cfg"/>
   ```

   The location for the property could be one of the following:

   - If the provider is for a DAS or server instance, add the property under the associated `<security-service>`.
   - If the provider is for a node agent, add the property under the associated `<node-agent>` element in the `domain.xml` file.

4. Restart the Application Server.

   The customized configurations will be in effect after the restart.

Further Information

- The Java 2 Standard Edition discussion of security can be viewed from `http://java.sun.com/j2se/1.5.0/docs/guide/security/index.html`.
- The *J2EE 1.4 Tutorial* chapter titled *Security* can be viewed from `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`.
- The *Administration Guide* chapter titled Chapter 10, “Configuring Message Security.”
- The *Developer's Guide* chapter titled *Securing Applications*.

◆ ◆ ◆ CHAPTER 10

Configuring Message Security

Some of the material in this chapter assumes a basic understanding of security and web services concepts. To learn more about these concepts, explore the resources listed in "Further Information" on page 116 before beginning this chapter.

This chapter describes the configuration of message layer security for web services in the Application Server. This chapter contains the following topics:

Overview of Message Security

In *message security*, security information is inserted into messages so that it travels through the networking layers and arrives with the message at the message destination(s). Message security differs from transport layer security (which is discussed in the *Security* chapter of the *J2EE 1.4 Tutorial*) in that message security can be used to decouple message protection from message transport so that messages remain protected after transmission.

Web Services Security: SOAP Message Security (WS-Security) is an international standard for interoperable Web Services Security that was developed in OASIS by a collaboration of all the major providers of web services technology (including Sun Microsystems). WS-Security is a message security mechanism that uses XML Encryption and XML Digital Signature to secure web services messages sent over SOAP. The WS-Security specification defines the use of various security tokens including X.509 certificates, SAML assertions, and username/password tokens to authenticate and encrypt SOAP web services messages.

The WS-Security specification can be viewed at http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.

Understanding Message Security in the Application Server

The Application Server offers integrated support for the WS-Security standard in its web services client and server-side containers. This functionality is integrated such that web services security is enforced by the containers of the Application Server on behalf of applications, and such that it can be applied to protect any web service application without requiring changes to the implementation of the application. The Application Server achieves this effect by providing facilities to bind SOAP layer message security providers and message protection policies to containers and to applications deployed in containers.

Assigning Message Security Responsibilities

In the Application Server, the "System Administrator" on page 118 and "Application Deployer" on page 119 roles are expected to take primary responsibility for configuring message security. In some situations, the "Application Developer" on page 119 may also contribute, although in the typical case either of the other roles may secure an existing application without changing its implementation without involving the developer. The responsibilities of the various roles are defined in the following sections:

System Administrator

The system administrator is responsible for:

- Configuring message security providers on the Application Server.
- Managing user databases.
- Managing keystore and truststore files.
- Configuring a Java Cryptography Extension (JCE) provider if using encryption and running a version of the Java SDK prior to version 1.5.0.
- Installing the samples server. This is only done if the `xms` sample application will be used to demonstrate the use of message layer web services security.

A system administrator uses the Administration Console to manage server security settings and uses a command line tool to manage certificate databases. In Platform Edition, certificates and private keys are stored in key stores and are managed with `keytool`. Standard Edition and Enterprise Edition store certificates and private keys in an NSS database, where they are managed using `certutil`. This document is intended primarily for system administrators. For an overview of message security tasks, see "Configuring the Application Server for Message Security" on page 123.

Application Deployer

The application deployer is responsible for:

- Specifying (at application assembly) any required application-specific message protection policies if such policies have not already been specified by upstream roles (the developer or assembler).
- Modifying Sun-specific deployment descriptors to specify application-specific message protection policies information (message-security-binding elements) to web service endpoint and service references.

These security tasks are discussed in the *Securing Applications* chapter of the *Developers' Guide*. For a link to this chapter, see "Further Information" on page 116.

Application Developer

The application developer can turn on message security, but is not responsible for doing so. Message security can be set up by the System Administrator so that all web services are secured, or by the Application Deployer when the provider or protection policy bound to the application must be different from that bound to the container.

The application developer or assembler is responsible for the following:

- Determining if an application-specific message protection policy is required by the application. If so, ensuring that the required policy is specified at application assembly which may be accomplished by communicating with the Application Deployer.

About Security Tokens and Security Mechanisms

The WS-Security specification provides an extensible mechanism for using security tokens to authenticate and encrypt SOAP web services messages. The SOAP layer message security providers installed with the Application Server may be used to employ username/password and X.509 certificate security tokens to authenticate and encrypt SOAP web services messages. Additional providers that employ other security tokens including SAML assertions will be installed with subsequent releases of the Application Server.

About Username Tokens

The Application Server uses *Username tokens* in SOAP messages to establish the authentication identity of the message *sender*. The recipient of a message containing a Username token (within embedded password) validates that the message sender is authorized to act as the user (identified in the token) by confirming that the sender knows the secret (the password) of the user.

When using a Username token, a valid user database must be configured on the Application Server

About Digital Signatures

The Application Server uses XML Digital signatures to bind an authentication identity to message *content*. Clients use digital signatures to establish their caller identity, analogous to the way basic authentication or SSL client certificate authentication have been used to do the same thing when transport layer security is being used. Digital signatures are verified by the message receiver to authenticate the source of the message content (which may be different from the sender of the message.)

When using digital signatures, valid keystore and truststore files must be configured on the Application Server. For more information on this topic, read "About Certificate Files" on page 103.

About Encryption

The purpose of encryption is to modify the data such that it can only be understood by its intended audience. This is accomplished by substituting an encrypted element for the original content. When predicated on public key cryptography, encryption can be used to establish the identity of the parties that can read a message.

When using Encryption, you must have an installed JCE provider that supports encryption. For more information on this topic, read "Configuring a JCE Provider" on page 125.

About Message Protection Policies

Message protection policies are defined for request message processing and response message processing and are expressed in terms of requirements for source and/or recipient authentication. A source authentication policy represents a requirement that the identity of the entity that sent a message or that defined the content of a message be established in the message such that it can be authenticated by the message receiver. A recipient authentication policy represents a requirement that the message be sent such that the identity of the entities that can receive the message can be established by the message sender. The providers apply specific message security mechanisms to cause the message protection policies to be realized in the context of SOAP web services messages.

Request and response message protection policies are defined when a provider is configured into a container. Application-specific message protection policies (at the granularity of the web service port or operation) may also be configured within the Sun-specific deployment descriptors of the application or application client. In any case, where message protection policies are defined, the request and response message protection policies of the client must match (be equivalent to) the request and response message protection policies of the server. For more information on defining application-specific message protection policies, refer to the *Securing Applications* chapter of the *Developers' Guide*.

Glossary of Message Security Terminology

The terminology used in this document is described below. The concepts are also discussed in "Configuring the Application Server for Message Security" on page 123.

- Authentication Layer

  The *authentication layer* is the message layer on which authentication processing must be performed. The Application Server enforces web services message security at the SOAP layer.

- Authentication Provider

  In this release of the Application Server, the Application Server invokes *authentication providers* to process SOAP message layer security.

  - A *client-side provider* establishes (by signature or username/password) the source identity of request messages and/or protects (by encryption) request messages such that they can only be viewed by their intended recipients. A client-side provider also establishes its container as an authorized recipient of a received response (by successfully decrypting it) and validates passwords or signatures in the response to authenticate the source identity associated with the response. Client-side providers configured in the Application Server can be used to protect the request messages sent and the response messages received by server-side components (servlets and EJB components) acting as clients of other services.
  - A *server-side provider* establishes its container as an authorized recipient of a received request (by successfully decrypting it) and validates passwords or signatures in the request to authenticate the source identity associated with the request. A server-side provider also establishes (by signature or username/password) the source identity of response messages and/or protects (by encryption) response messages such that they can only be viewed by their intended recipients. *Server-side providers* are only invoked by server-side containers.

- Default Server Provider

  The *default server provider* is used to identify the server provider to be invoked for any application for which a specific server provider has not been bound. The *default server provider* is sometimes referred to as the *default provider*.

- Default Client Provider

  The *default client provider* is used to identify the client provider to be invoked for any application for which a specific client provider has not been bound.

- Request Policy

  The *request policy* defines the authentication policy requirements associated with request processing performed by the authentication provider. Policies are expressed in message sender order such that a requirement that encryption occur after content would mean that the message receiver would expect to decrypt the message before validating the signature.

- Response Policy

The *response policy* defines the authentication policy requirements associated with response processing performed by the authentication provider. Policies are expressed in message sender order such that a requirement that encryption occur after content would mean that the message receiver would expect to decrypt the message before validating the signature.

Securing a Web Service

Web services deployed on the Application Server are secured by binding SOAP layer message security providers and message protection policies to the containers in which the applications are deployed or to web service endpoints served by the applications. SOAP layer message security functionality is configured in the client-side containers of the Application Server by binding SOAP layer message security providers and message protection policies to the client containers or to the portable service references declared by client applications.

When the Application Server is installed, SOAP layer message security providers are configured in the client and server-side containers of the Application Server, where they are available for binding for use by the containers, or by individual applications or clients deployed in the containers. During installation, the providers are configured with a simple message protection policy that, if bound to a container, or to an application or client in a container, would cause the source of the content in all request and response messages to be authenticated by XML digital signature.

The administrative interfaces of the Application Server can be employed to bind the existing providers for use by the server-side containers of the Application Server, to modify the message protection policies enforced by the providers, or to create new provider configurations with alternative message protection policies. Analogous administrative operations can be performed on the SOAP message layer security configuration of the application client container as defined in "Enabling Message Security for Application Clients" on page 129.

By default, message layer security is disabled on the Application Server. To configure message layer security for the Application Server follow the steps outlined in "Configuring the Application Server for Message Security" on page 123. If you want to cause web services security to be used to protect all web services applications deployed on the Application Server, follow the steps in "Enabling Providers for Message Security" on page 127.

Once you have completed the above steps (which may include restarting the Application Server), web services security will be applied to all web services applications deployed on the Application Server.

Configuring Application-Specific Web Services Security

Application-specific web services security functionality is configured (at application assembly) by defining message-security-binding elements in the Sun-specific deployment descriptors of the application. These message-security-binding elements are used to associate a specific provider or message protection policy with a web services endpoint or service reference, and may be qualified so that they apply to a specific port or method of the corresponding endpoint or referenced service.

For more information on defining application specific message protection policies, refer to the *Securing Applications* chapter of the *Developers' Guide*. There is a link to this chapter in "Further Information" on page 116.

Securing the Sample Application

The Application Server ships with a sample application named `xms`. The `xms` application features a simple web service that is implemented by both a J2EE EJB endpoint and a Java Servlet endpoint. Both endpoints share the same service endpoint interface. The service endpoint interface defines a single operation, `sayHello`, which takes a string argument, and returns a `String` composed by pre-pending `Hello` to the invocation argument.

The `xms` sample application is provided to demonstrate the use of the Application Server's WS-Security functionality to secure an existing web services application. The instructions which accompany the sample describe how to enable the WS-Security functionality of the Application Server such that it is used to secure the `xms` application. The sample also demonstrates the binding of WS-Security functionality directly to the application (as described in "Configuring Application-Specific Web Services Security" on page 123 application.

The `xms` sample application is installed in the directory:
install-dir`/samples/webservices/security/ejb/apps/xms/`.

For information on compiling, packaging, and running the `xms` sample application, refer to the *Securing Applications* chapter of the *Developers' Guide*.

Configuring the Application Server for Message Security

Actions of Request and Response Policy Configurations

The following table shows message protection policy configurations and the resulting message security operations performed by the WS-Security SOAP message security providers for that configuration.

TABLE 10–1 Message protection policy to WS-Security SOAP message security operation mapping

Message Protection Policy	Resulting WS-Security SOAP message protection operations
auth-source="sender"	The message contains a `wsse:Security` header that contains a `wsse:UsernameToken` (with password).
auth-source="content"	The content of the SOAP message Body is signed. The message contains a `wsse:Security` header that contains the message Body signature represented as a `ds:Signature`.
auth-source="sender" auth-recipient="before-content" OR auth-recipient="after-content"	The content of the SOAP message Body is encrypted and replaced with the resulting `xend:EncryptedData`. The message contains a `wsse:Security` header that contains a `wsse:UsernameToken (with password)` and an `xenc:EncryptedKey`. The `xenc:EncryptedKey` contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.
auth-source="content" auth-recipient="before-content"	The content of the SOAP message Body is encrypted and replaced with the resulting `xend:EncryptedData`. The `xenc:EncryptedData` is signed. The message contains a `wsse:Security` header that contains an `xenc:EncryptedKey` and a `ds:Signature`. The `xenc:EncryptedKey` contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.
auth-source="content" auth-recipient="after-content"	The content of the SOAP message Body is signed, then encrypted, and then replaced with the resulting `xend:EncryptedData`. The message contains a `wsse:Security` header that contains an `xenc:EncryptedKey` and a `ds:Signature`. The `xenc:EncryptedKey` contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.

TABLE 10–1 Message protection policy to WS-Security SOAP message security operation mapping *(Continued)*

Message Protection Policy	Resulting WS-Security SOAP message protection operations
auth-recipient="before-content" OR auth-recipient="after-content"	The content of the SOAP message Body is encrypted and replaced with the resulting `xend:EncryptedData`. The message contains a `wsse:Security` header that contains an `xenc:EncryptedKey`. The `xenc:EncryptedKey` contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.
No policy specified.	No security operations are performed by the modules.

Configuring Other Security Facilities

The Application Server implements message security using message security providers integrated in its SOAP processing layer. The message security providers depend on other security facilities of Application Server.

1. If using a version of the Java SDK prior to version 1.5.0, and using encryption technology, configure a JCE provider.
2. Configuring a JCE provider is discussed in "Configuring a JCE Provider" on page 125.
3. If using a username token, configure a user database, if necessary. When using a username/password token, an appropriate realm must be configured and an appropriate user database must be configured for the realm.
4. Manage certificates and private keys, if necessary.

After You Finish

Once the facilities of the Application Server are configured for use by message security providers, then the providers installed with the Application Server may be enabled as described in "Enabling Providers for Message Security" on page 127.

Configuring a JCE Provider

The Java Cryptography Extension (JCE) provider included with J2SE 1.4.x does not support RSA encryption. Because the XML Encryption defined by WS-Security is typically based on RSA encryption, in order to use WS-Security to encrypt SOAP messages you must download and install a JCE provider that supports RSA encryption.

Note – RSA is public-key encryption technology developed by RSA Data Security, Inc. The acronym stands for Rivest, Shamir, and Adelman, the inventors of the technology.

If you are running the Application Server on version 1.5 of the Java SDK, the JCE provider is already configured properly. If you are running the Application Server on version 1.4.x of the Java SDK, you can add a JCE provider statically as part of your JDK environment, as follows.

1. Download and install a JCE provider JAR (Java ARchive) file.

   The following URL provides a list of JCE providers that support RSA encryption: `http://java.sun.com/products/jce/jce14_providers.html`.

2. Copy the JCE provider JAR file to *java-home*`/jre/lib/ext/`.

3. Stop the Application Server.

   If the Application Server is not stopped and then restarted later in this process, the JCE provider will not be recognized by the Application Server.

4. Edit the *java-home*`/jre/lib/security/java.security` properties file in any text editor. Add the JCE provider you've just downloaded to this file.

   The `java.security` file contains detailed instructions for adding this provider. Basically, you need to add a line of the following format in a location with similar properties:

   `security.provider.`*n*`=`*provider-class-name*

   In this example, *n* is the order of preference to be used by the Application Server when evaluating security providers. Set *n* to 2 for the JCE provider you've just added.

   For example, if you've downloaded The Legion of the Bouncy Castle JCE provider, you would add this line.

   ```
   security.provider.2=org.bouncycastle.jce.provider.
      BouncyCastleProvider
   ```

   Make sure that the Sun security provider remains at the highest preference, with a value of 1.

   ```
   security.provider.1=sun.security.provider.Sun
   ```

   Adjust the levels of the other security providers downward so that there is only one security provider at each level.

   The following is an example of a `java.security` file that provides the necessary JCE provider and keeps the existing providers in the correct locations.

   ```
   security.provider.1=sun.security.provider.Sun
   security.provider.2=org.bouncycastle.jce.provider.
      BouncyCastleProvider
   security.provider.3=com.sun.net.ssl.internal.ssl.Provider
   security.provider.4=com.sun.rsajca.Provider
   security.provider.5=com.sun.crypto.provider.SunJCE
   security.provider.6=sun.security.jgss.SunProvider
   ```

5. Save and close the file.

6. Restart the Application Server.

Message Security Setup

Most of the steps for setting up the Application Server for using message security can be accomplished using the Administration Console, the `asadmin` command-line tool, or by manually editing system files. In general, editing system files is discouraged due to the possibility of making unintended changes that prevent the Application Server from running properly, therefore, where possible, steps for configuring the Application Server using the Administration Console are shown first, with the `asadmin` tool command shown after. Steps for manually editing system files are shown only when there is no Administration Console or `asadmin` equivalent.

Support for message layer security is integrated into the Application Server and its client containers in the form of (pluggable) authentication modules. By default, message layer security is disabled on the Application Server. The following sections provide the details for enabling, creating, editing, and deleting message security configurations and providers.

In most cases, it will be necessary to restart the Application Server after performing the administrative operations listed above. This is especially the case if you want the effects of the administrative change to be applied to applications that were already deployed on the Application Server at the time the operation was performed.

Enabling Providers for Message Security

To enable message security for web services endpoints deployed in the Application Server, you must specify a provider to be used by default on the server side. If you enable a default provider for message security, you also need to enable providers to be used by clients of the web services deployed in the Application Server. Information for enabling the providers used by clients is discussed in "Enabling Message Security for Application Clients" on page 129.

To enable message security for web service invocations originating from deployed endpoints, you must specify a default client provider. If you enabled a default client provider for the Application Server, you must ensure that any services invoked from endpoints deployed in the Application Server are compatibly configured for message layer security.

Use the command-line utility:

- To specify the default server provider:

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
default_provider=ServerProvider
```

- To specify the default client provider:

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
default_client_provider=ClientProvider
```

Configuring the Message Security Provider

Typically, a provider would be re-configured to modify its message protection policies, although the provider type, implementation class, and provider-specific configuration properties may also be modified.

Use the command-line utility to set the response policy, replace the word `request` in the following commands with `response`.

- Add a request policy to the client and set the authentication source:

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ClientProvider.request-policy.auth_source=
sender | content
```

- Add a request policy to the server and set the authentication source:

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ServerProvider.request-policy.auth_source=
sender | content
```

- Add a request policy to the client and set the authentication recipient:

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ClientProvider.request-policy.auth_recipient=
before-content | after-content
```

- Add a request policy to the server and set the authentication recipient:

```
asadmin set --user admin-user --port admin-port
server-config.security-service.message-security-config.SOAP.
provider-config.ServerProvider.request-policy.auth_recipient=
before-content | after-content
```

Creating a Message Security Provider

To configure an existing provider using the Admin Console, select Configuration node > the instance to Configure> Security node > Message Security node > SOAP node > Providers tab.

For more detailed instructions on creating a message security provider, see the Admin Console online help.

Enabling Message Security for Application Clients

The message protection policies of client providers must be configured such that they are equivalent to the message protection policies of the server-side providers they will be interacting with. This is already the case for the providers configured (but not enabled) when the Application Server is installed.

To enable message security for client applications, modify the Application Server specific configuration for the application client container.

Setting the Request and Response Policy for the Application Client Configuration

The *request and response policies* define the authentication policy requirements associated with request and response processing performed by the authentication provider. Policies are expressed in message sender order such that a requirement that encryption occur after content would mean that the message receiver would expect to decrypt the message before validating the signature.

To achieve message security, the request and response policies must be enabled on both the server and client. When configuring the policies on the client and server, make sure that the client policy matches the server policy for request/response protection at application-level message binding.

To set the request policy for the application client configuration, modify the Application Server specific configuration for the application client container as described in "Enabling Message Security for Application Clients" on page 129. In the application client configuration file, add the `request-policy` and `response-policy` elements as shown to set the request policy.

The other code is provided for reference. The other code may differ slightly in your installation. Do not change it.

```
<client-container>
  <target-server name="your-host" address="your-host"
      port="your-port"/>
  <log-service file="" level="WARNING"/>
  <message-security-config auth-layer="SOAP"
      default-client-provider="ClientProvider">
    <provider-config
        class-name="com.sun.enterprise.security.jauth.ClientAuthModule"
```

```
        provider-id="ClientProvider" provider-type="client">
      <request-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
      <response-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
       <property name="security.config"
           value="install-dir/lib/appclient/wss-client-config.xml"/>
    </provider-config>
  </message-security-config>
</client-container>
```

Valid values for `auth-source` include `sender` and `content`. Valid values for `auth-recipient` include `before-content` and `after-content`. A table describing the results of various combinations of these values can be found in "Actions of Request and Response Policy Configurations" on page 124.

To not specify a request or response policy, leave the element blank, for example:

```
<response-policy/>
```

Further Information

- The Java 2 Standard Edition discussion of security can be viewed from `http://java.sun.com/j2se/1.4.2/docs/guide/security/index.html`.
- The *J2EE 1.4 Tutorial* chapter titled *Security* can be viewed from `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`.
- The *Administration Guide* chapter titled Chapter 9, "Configuring Security."
- The *Developer's Guide* chapter titled *Securing Applications*.
- The Oasis Web Services Security: SOAP Message Security (WS-Security) specification, can be viewed from http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.
- The OASIS Web Services Security Username Token Profile 1.0, can be found at http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf.
- The OASIS Web Services Security X.509 Certificate Token Profile 1.0, can be found at http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf.
- The *XML-Signature Syntax and Processing* document can be viewed at `http://www.w3.org/TR/xmldsig-core/`.
- The *XML Encryption Syntax and Processing* document can be viewed at `http://www.w3.org/TR/xmlenc-core/`.

◆ ◆ ◆ CHAPTER 11

Transactions

By enclosing one or more steps in an indivisible unit of work, a transaction ensures data integrity and consistency. This chapter contains the following sections:

What is a Transaction?

A transaction is a series of discreet actions in an application that must all complete successfully or else all the changes in each action are backed out. For example, to transfer funds from a checking account to a savings account is a transaction with the following steps:

1. Check to see if the checking account has enough money to cover the transfer.
2. If there's enough money in the checking account debit the amount from the checking account.
3. Credit the money to the savings account.
4. Record the transfer to the checking account log.
5. Record the transfer to the savings account log.

If any of these steps fails, all changes from the preceding steps must be backed out, and the checking account and savings account must be in the same state as they were before the transaction started. This event is called a *rollback*. If all the steps complete successfully, the transaction is in a *committed* state. Transactions end in either a commit or a rollback.

Transactions in J2EE Technology

Transaction processing in J2EE technology involves the following five participants:

- Transaction Manager
- Application Server
- Resource Manager(s)
- Resource Adapter(s)
- User Application.

Each of these entities contribute to reliable transaction processing by implementing the different APIs and functionalities, discussed below:

- The Transaction Manager provides the services and management functions required to support transaction demarcation, transactional resource management, synchronization, and transaction context propagation.
- The Application Server provides the infrastructure required to support the application runtime environment that includes transaction state management.
- The Resource Manager (through a resource adapter) provides the application access to resources. The resource manager participates in distributed transactions by implementing a transaction resource interface used by the transaction manager to communicate transaction association, transaction completion and recovery work. An example of such a resource manager is a relational database server.
- A Resource Adapter is a system level software library that is used by the application server or client to connect to a Resource Manager. A Resource Adapter is typically specific to a Resource Manager. It is available as a library and is used within the address space of the client using it. An example of such a resource adapter is a JDBC driver.
- A Transactional User Application developed to operate in an application server environment looks up transactional data sources and, optionally, the transaction manager, using JNDI. The application may use declarative transaction attribute settings for enterprise beans or explicit programmatic transaction demarcation.

Recovering Transactions

Transactions might be incomplete either because the server crashed or a resource manager crashed. It is essential to complete these stranded transactions and recover from the failures. Application Server is designed to recover from these failures and complete the transactions upon server startup.

While performing the recovery, if some of the resources are unreachable the server restart may be delayed as it tries to recover the transactions.

When the transaction spans across servers, the server that started the transaction can contact the other servers to get the outcome of the transactions. If the other servers are unreachable, the transaction uses the Heuristic Decision field to determine the outcome.

You can configure Application Server to recover transactions using the Admin Console. Detailed procedures for doing this are provided in the Admin Console Online Help.

Transaction Timeout Value

By default, the server does not timeout a transaction. That is, the server waits indefinitely for a transaction to complete. If you set a timeout value for transactions, if a transaction isn't completed within the configured time, the Application Server rolls back the transaction. Detailed steps to do this are provided in the Admin Console Online Help.

Transaction Logs

The transaction log records the information about each transaction in order to maintain the data integrity of the resources involved and to recover from failures. Transaction logs are kept in the `tx` subdirectory of the directory specified by the Transaction Log Location field. These logs are not human readable.

Keypoint Interval

Keypoint operations compress the transaction log file. The keypoint interval is the number of transactions between keypoint operations on the log. Keypoint operations can reduce the size of the transaction log files. A larger number of keypoint intervals (for example, 2048) results in larger transaction log files, but fewer keypoint operations, and potentially better performance. A smaller keypoint interval (for example, 256) results in smaller log files but slightly reduced performance due to the greater frequency of keypoint operations.

◆ ◆ ◆ CHAPTER 12

Configuring the HTTP Service

The HTTP service is the component of the Application Server that provides facilities for deploying web applications and for making deployed web applications accessible by HTTP clients. These facilities are provided by means of two kinds of related objects, virtual servers and HTTP listeners.

This chapter discusses the following topics:

Virtual Servers

A virtual server, sometimes called a virtual host, is an object that allows the same physical server to host multiple Internet domain names. All virtual servers hosted on the same physical server share the Internet Protocol (IP) address of that physical server. A virtual server associates a domain name for a server (such as `www.aaa.com`) with the particular server on which the Application Server is running.

Note – Do not confuse an Internet domain with the administrative domain of the Application Server.

For instance, assume you want to host these domains on your physical server:

```
www.aaa.com
www.bbb.com
www.ccc.com
```

Assume also that `www.aaa.com`, `www.bbb.com`, and `www.ccc.com` have web modules `web1`, `web2`, and `web3`, respectively, associated with them.

This means that all of these URLs are handled by your physical server:

```
http://www.aaa.com:8080/web1
http://www.bbb.com:8080/web2
http://www.ccc.com:8080/web3
```

The first URL is mapped to virtual host `www.aaa.com`, the second URL is mapped to virtual host `www.bbb.com`, and the third is mapped to virtual host `www.ccc.com`.

On the other hand, the following URL results in a 404 return code, because `web3` isn't registered with `www.bbb.com`:

```
http://www.bbb.com:8080/web3
```

For this mapping to work, make sure that `www.aaa.com`, `www.bbb.com`, and `www.ccc.com` all resolve to your physical server's IP address. They need to be registered with the DNS server for your network. In addition, on a UNIX system, add these domains to your `/etc/hosts` file (if the setting for `hosts` in your `/etc/nsswitch.conf` file includes `files`).

When the Application Server is started, it starts the following virtual servers automatically:

- A virtual server named `server`, which hosts all user-defined web modules
- A virtual server named `__asadmin`, which hosts all administration-related web modules (specifically, the Administration Console). This server is restricted; you cannot deploy web modules to this virtual server.

For development, testing, and deployment of web services in a non-production environment, `server` is often the only virtual server required. In a production environment, additional virtual servers provide hosting facilities for users and customers so that each appears to have its own web server, even though there is only one physical server.

HTTP Listeners

Each virtual server provides connections between the server and clients through one or more HTTP listeners. Each HTTP listener is a listen socket that has an IP address, a port number, a server name, and a default virtual server.

HTTP listeners must have a unique combination of port number and IP address. For example, an HTTP listener can listen on all configured IP addresses on a given port for a machine by specifying the IP address 0.0.0.0. Alternatively, the HTTP listener can specify a unique IP address for each listener, but use the same port.

Since an HTTP listener is a combination of IP address and port number, you can have multiple HTTP listeners with the same IP address and different port numbers (for example, 1.1.1.1:8081 and 1.1.1.1:8082), or with different IP addresses and the same port number (for example, 1.1.1.1:8081 and 1.2.3.4:8081, if your machine was configured to respond to both these addresses).

However, if an HTTP listener uses the 0.0.0.0 IP address, which listens on all IP addresses on a port, you cannot create HTTP listeners for additional IP addresses that listen on the same port for a specific IP address. For example, if an HTTP listener uses 0.0.0.0:8080 (all IP addresses on port 8080), another HTTP listener cannot use 1.2.3.4:8080.

Because the system running the Application Server typically has access to only one IP address, HTTP listeners typically use the 0.0.0.0 IP address and different port numbers, with each port number serving a different purpose. If the system does have access to more than one IP address, each address can serve a different purpose.

By default, when the Application Server starts, it has the following HTTP listeners:

- Two HTTP listeners named `http-listener-1` and `http-listener-2`, associated with the virtual server named `server`. The listener named `http-listener-1` does not have security enabled; `http-listener-2` has security enabled.
- An HTTP listener named `admin-listener`, associated with the virtual server named `__asadmin`. This listener has security enabled.

All these listeners use the IP address 0.0.0.0 and the port numbers specified as the HTTP server port numbers during installation of the Application Server. If the Application Server uses the default port number values, `http-listener-1` uses port 8080, `http-listener-2` uses port 8181, and `admin-listener` uses port 4849.

Each HTTP listener has a default virtual server. The default virtual server is the server to which the HTTP listener routes all request URLs whose host component does not match any of the virtual servers that are associated with the HTTP listener (a virtual server is associated with an HTTP listener by listing the HTTP listener in its `http-listeners` attribute).

In addition, specify the number of acceptor threads in the HTTP listener. Acceptor threads are threads that wait for connections. The threads accept connections and put them in a queue, called the connection queue, where they are then picked up by worker threads. Configure enough acceptor threads so that there is always one available when a new request comes in, but few enough so that they do not provide too much of a burden on the system.The connection queue includes both the new connections just accepted by acceptor threads and persistent connections managed by the Keep-Alive connection management subsystem.

A set of request processing threads retrieves incoming HTTP requests from the connection queue and processes the requests. These threads parse the HTTP headers, select the appropriate virtual server, and run through the request processing engine to service the request. When there are no more requests to process, but the connection can be kept persistent (either by using `HTTP/1.1` or sending a `Connection: keep-alive` header), the request processing thread assumes the connection to be idle and passes the connection to the Keep-Alive connection management subsystem.

The Keep-Alive subsystem periodically polls such idle connections and queues those connections with activity into the connection queue for future processing. From there, a request processing thread again retrieves the connection and processes its request. The Keep-Alive

subsystem is multi-threaded, as it manages potentially tens of thousands of connections. Efficient polling techniques are used, by dividing the number of connections into smaller subsets, to determine which connections are ready with requests and which of those connections have idled for sufficient time to deem them closed (beyond a maximum permissible Keep-Alive timeout).

The HTTP listener's server name is the host name that appears in the URLs the server sends to the client as part of a redirect. This attribute affects URLs the server automatically generates; it does not affect the URLs for directories and files stored in the server. This name is normally the alias name if the server uses an alias. If a client sends a `Host:` header, that host name supersedes the HTTP listener's server name value in redirects.

Specify a redirect port to use a different port number from that specified in the original request. A *redirect* occurs in one of these situations:

- If a client tries to access a resource that no longer exists at the specified URL (that is, the resource has moved to another location), the server redirects the client to the new location (instead of returning a 404), by returning a designated response code and including the new location in the response's Location header.
- If a client tries to access a resource that is protected (for example, SSL) on the regular HTTP port, the server redirects the request to the SSL-enabled port. In this case, the server returns a new URL in the Location response header, in which the original insecure port has been replaced with the SSL-enabled port. The client then connects to this new URL.

Specify also whether security is enabled for an HTTP listener and what kind of security is used (for example, which SSL protocol and which ciphers).

To access a web application deployed on the Application Server, use the URL `http://localhost:8080/` (or `https://localhost:8181/` if it is a secure application), along with the context root specified for the web application. To access the Administration Console, use the URL `https://localhost:4849/` or `https://localhost:4849/asadmin/` (its default context root).

Because a virtual server must specify an existing HTTP listener, and because it cannot specify an HTTP listener that is already being used by another virtual server, create at least one HTTP listener before creating a new virtual server.

◆ ◆ ◆ CHAPTER 13

Configuring the Object Request Broker

This chapter describes how to configure the Object Request Broker (ORB) and IIOP listeners. It has the following sections:

CORBA

The Application Server supports a standard set of protocols and formats that ensure interoperability. Among these protocols are those defined by CORBA.

The CORBA (Common Object Request Broker Architecture) model is based on clients requesting services from distributed objects or servers through a well-defined interface by issuing requests to the objects in the form of remote method requests. A remote method request carries information about the operation that needs to be performed, including the object name (called an object reference) of the service provider and parameters, if any, for the invoked method. CORBA automatically handles network programming tasks such as object registration, object location, object activation, request de-multiplexing, error-handling, marshalling, and operation dispatching.

What is the ORB?

The Object Request Broker (ORB) is the central component of CORBA. The ORB provides the required infrastructure to identify and locate objects, handle connection management, deliver data, and request communication.

A CORBA object never talks directly with another. Instead, the object makes requests through a remote stub to the ORB running on the local machine. The local ORB then passes the request to

an ORB on the other machine using the Internet Inter-Orb Protocol (IIOP for short). The remote ORB then locates the appropriate object, processes the request, and returns the results.

IIOP can be used as a Remote Method Invocation (RMI) protocol by applications or objects using RMI-IIOP. Remote clients of enterprise beans (EJB modules) communicate with the Application Server via RMI-IIOP.

IIOP Listeners

An IIOP listener is a listen socket that accepts incoming connections from the remote clients of enterprise beans and from other CORBA-based clients. Multiple IIOP listeners can be configured for the Application Server. For each listener, specify a port number, a network address, and optionally, security attributes.

Working with the ORB

To create an IIOP listener, select Configuration > ORB > IIOP Listeners and click New in the Admin Console. Alternatively, you can use the following asadmin command to create IIOP listeners: `create-iiop-listener(1)` and `create-ssl(1)`.

To edit an IIOP listener, select Configuration > ORB > IIOP Listeners and select the listener to be modified in the Admin Console. Modify the settings. If you have changed the port number, restart the server. The ORB uses thread pools to respond to requests from remote clients of enterprise beans and other clients that communicate via RMI-IIOP.

To delete an IIOP listener, select Configuration > ORB > IIOP Listeners and select the listener to be deleted in the Admin Console. Alternatively, you can use the `delete-iiop-listener(1)` command.

The `ORBCommunicationsRetryTimeout` property specifies the number of seconds the ORB client will try to establish a connection to an unreachable ORB back-end. The default value is `60` seconds. With this default setting, you may see a large number of CORBA exceptions in the logs, as well as high network usage if the ORB back-end is not reachable.

In such cases, set the `ORBCommunicationsRetryTimeout` to a lower value.

Third-party ORBs

The Sun Java System Application Server can be used in conjunction with third-party ORB software. To support such third-party ORBs, you need to revise the server-side settings.

To implement the support for a third-party ORB in Application Server, you need to edit the files, `domain.xml` and `server.policy`. For detailed instructions on how to configure a sample

third-party ORB, see Configuring Sun Java System Application Server for Third-Party ORBs (`http://developers.sun.com/prodtech/appserver/reference/techart/orb.html`)

◆ ◆ ◆ CHAPTER 14

Thread Pools

The Java Virtual Machine (JVM) can support many threads of execution at once. To help performance, the Application Server maintains one or more thread pools. It is possible to assign specific thread pools to connector modules and to the ORB.

One thread pool can serve multiple connector modules and enterprise beans. Request threads handle user requests for application components. When the server receives a request, it assigns the request to a free thread from the thread pool. The thread executes the client's requests and returns results. For example, if the request needs to use a system resource that is currently busy, the thread waits until that resource is free before allowing the request to use that resource.

Specify the minimum and maximum number of threads that are reserved for requests from applications. The thread pool is dynamically adjusted between these two values. The minimum thread pool size that is specified signals the server to allocate at least that many threads in reserve for application requests. That number is increased up to the maximum thread pool size that is specified.

Increasing the number of threads available to a process allows the process to respond to more application requests simultaneously.

Avoid thread starvation, where one resource adapter or application occupies all threads in the Application Server, by dividing the Application Server threads into different thread-pools.

This chapter contains the following topics:

Configuring Thread Pools

To create a thread pool using the Administration Console, go to Configuration > Thread Pools > Current Pools > New.

- Enter the name of the thread pool in the Thread Pool ID field.
- Enter the minimum number of threads in the thread pool servicing requests in this queue in the Minimum Thread Pool Size field.

  These threads are created up front when this thread pool is instantiated.
- Enter the maximum number of threads in the thread pool servicing requests in this queue in the Maximum Thread Pool Size field.

  This is the upper limit on the number of threads that exist in the thread pool.
- Enter the number, in seconds, after which idle threads are removed from pool in the Idle Timeout field.
- Enter the total number of work queues that are serviced by this thread pool in the Number of Work Queues field.
- Restart the Application Server.

For more details on creating thread pools, click Help in the Administration Console.

You can also create a thread pool from the command line by using the `asadmin` command, `create-threadpool(1)`.

To edit a settings for a thread pool using the Administration Console, go to Configuration > Thread Pools > Current Pools and select the pool you want to configure. Modify the values for the selected thread pool, save and restart the Application Server.

For more details on editing thread pools, click Help in the Administration Console.

To delete a thread pool using the Administration Console, go to Configuration > Thread Pools > Current Pools. Check the thread pool name to be deleted and click Delete.

Restart the Application Server.

You can also create a thread pool from the command line by using the `asadmin` command, `delete-threadpool(1)`.

◆ ◆ ◆ CHAPTER 15

Configuring Logging

This chapter briefly describes how to configure logging and view the server log. It contains the following sections:

Log Records

The Application Server uses the Java 2 platform Logging API specified in JSR 047. Application Server logging messages are recorded in the server log, normally found at *domain-dir*`/logs/server.log`.

The *domain-dir*`/logs` directory contains two other kinds of logs in addition to the server log. In the `access` subdirectory are the HTTP Service access logs, and in the `tx` subdirectory are the Transaction Service logs. For information about these logs, consult the Administration Console online help.

The components of the Application Server generate logging output. Application components can also generate logging output.

Application components may use the Apache Commons Logging Library to log messages. The platform standard JSR 047 API, however, is recommended for better log configuration.

Log records follow a uniform format:

```
[#|yyyy-mm-ddThh:mm:ss.SSS-Z|Log Level|ProductName-Version|LoggerName|Key Value Pairs|Message|#]
```

For example:

```
[#|2004-10-21T13:25:53.852-0400|INFO|sun-appserver-e8.1|javax.enterprise.
system.core|_ThreadID=13;|CORE5004: Resource Deployed:
[cr:jms/DurableConnectionFactory].|#]
```

In this example,

- `[#` and `#]` mark the beginning and end of the record.
- The vertical bar (`|`) separates the record fields.
- `2004-10-21T13:25:53.852-0400` specifies the date and time.
- The *Log Level* is `INFO`. This level may have any of the following values: `SEVERE`, `WARNING`, `INFO`, `CONFIG`, `FINE`, `FINER`, and `FINEST`.
- The *ProductName-Version* is `sun-appserver-ee8.1`.
- The *LoggerName* is a hierarchical logger namespace that identifies the source of the log module, in this case `javax.enterprise.system.core`.
- The *Key Value Pairs* are key names and values, typically a thread ID such as `_ThreadID=14;`.
- The *Message* is the text of the log message. For all Application Server `SEVERE` and `WARNING` messages and many `INFO` messages, it begins with a message ID that consists of a module code and a numerical value (in this case, `CORE5004`).

You can use the `com.sun.enterprise.server.logging.max_history_files` system property to limit the number of rotated log files for both access logging and the server log. The property specifies the maximum number of access log files to keep, starting with the most recent ones.

- If the property is not set, then no history files cleanup is done.
- If the property is set and it's either an invalid number or null, then a default of 10 history files is kept.
- If the property is set to 0, then no history files are kept.

In the Admin Console, access the config/server that this system property needs to be set for. On the command line, use the `create-jvm-option` command.

Setting Custom Log Levels

This section explains how to configure custom logging levels for applications that make use of the `java.util.logging` package and access the Application Server's logging sub system.

The `java.util.logging` package provides a hierarchical name space in which logger instances can be created. Whether a particular logging record is output to an Application Server instance's log file depends on the log level of the Log Record and the log level specified.

The Application Serverlogger settings configuration provides over twenty logging modules that allow fine grained control over the Application Server's own internal logging. There is also an option to create additional custom Log Modules by specifying a module name and the logging level that the module should use.

The important point here is that the logger is a static name and no inheritance is provided. Therefore, if a custom logger is configured with the name `com.someorg.app` and an application attempts to look up the logger `com.someorg.app.submodule`, then it will not be provided with a logger that inherits the settings from `com.someorg.app`. Instead, `com.someorg.app.submodule` will have a default logger that is set to log at the INFO level or higher.

If an application needs to use logger inheritance, this can be configured by editing the `logging.properties` file of the Java runtime that is being used to run the Application Server. For example, adding the following entry to the `logging.properties` file, would result in `com.someorg.app.submodule` inheriting the same FINE level when it is created:

```
com.someorg.app.level = FINE
```

For more details about the Java logging API, refer to the Java documentation at `http://java.sun.com/j2se/1.5.0/docs/api/java/util/logging/package-summary.html`, as well as the other `java.util.logging` classes.

The Logger Namespace Hierarchy

The Application Server provides a logger for each of its modules. The following table lists the names of the modules and the namespace for each logger in alphabetical order, as they appear on the Log Levels page of the Administration Console. The last three modules in the table do not appear on the Log Levels page.

TABLE 15–1 Application Server Logger Namespaces

Module Name	Namespace
Admin	`javax.enterprise.system.tools.admin`
Classloader	`javax.enterprise.system.core.classloading`
CMP	`javax.enterprise.system.container.cmp`
Configuration	`javax.enterprise.system.core.config`
Connector	`javax.enterprise.resource.resourceadapter`
CORBA	`javax.enterprise.resource.corba`
Deployment	`javax.enterprise.system.tools.deployment`
EJB Container	`javax.enterprise.system.container.ejb`
JavaMail	`javax.enterprise.resource.javamail`
JAXR	`javax.enterprise.resource.webservices.registry`

TABLE 15–1 Application Server Logger Namespaces *(Continued)*

Module Name	Namespace
JAX-RPC	`javax.enterprise.resource.webservices.rpc`
JDO	`javax.enterprise.resource.jdo`
JMS	`javax.enterprise.resource.jms`
JTA	`javax.enterprise.resource.jta`
JTS	`javax.enterprise.system.core.transaction`
MDB Container	`javax.enterprise.system.container.ejb.mdb`
Naming	`javax.enterprise.system.core.naming`
Node Agent (Enterprise Edition only)	`javax.ee.enterprise.system.nodeagent`
Root	`javax.enterprise`
SAAJ	`javax.enterprise.resource.webservices.saaj`
Security	`javax.enterprise.system.core.security`
Server	`javax.enterprise.system`
Synchronization (Enterprise Edition only)	`javax.ee.enterprise.system.tools.synchronization`
Util	`javax.enterprise.system.util`
Verifier	`javax.enterprise.system.tools.verifier`
Web Container	`javax.enterprise.system.container.web`
Core	`javax.enterprise.system.core`
System Output (`System.out.println`)	`javax.enterprise.system.stream.out`
System Error (`System.err.println`)	`javax.enterprise.system.stream.err`

◆ ◆ ◆ CHAPTER 16

Monitoring Components and Services

Use monitoring to observe the runtime state of various components and services deployed in a server instance of theApplication Server. With the information on the state of runtime components and processes, it is possible to identify performance bottlenecks for tuning purposes, aid capacity planning, predict failures, do root cause analysis in case of failures, and ensure that everything is functioning as expected.

Turning monitoring on reduces performance by increasing overhead.

This chapter contains the following sections:

Overview of Monitoring

To monitor the Application Server, perform these steps:

1. Enable the monitoring of specific services and components using either the Administration Console or the `asadmin` tool.

   For more information on this step, refer to "Enabling and Disabling Monitoring" on page 174.

2. View monitoring data for the specified services or components using either the Administration Console or the `asadmin` tool.

   For more information on this step, refer to "Viewing Monitoring Data" on page 175.

About the Tree Structure of Monitorable Objects

The Application Server uses a tree structure to track monitorable objects. Because the tree of monitoring objects is dynamic, it changes as components are added, updated, or removed in the instance. The root object in the tree is the server instance name, for example, `server`. (In the Platform Edition, just one server instance is permitted.)

The following command displays the top level of the tree:

```
asadmin> list --user adminuser --monitor server
server.applications
server.http-service
server.connector-service
server.jms-service
server.jvm
server.orb
server.resources
server.thread-pools
```

The following sections describe these sub-trees:

- "The Applications Tree" on page 150
- "The HTTP Service Tree" on page 151
- "The Resources Tree" on page 152
- "The Connector Service Tree" on page 152
- "The JMS Service Tree" on page 152
- "The ORB Tree" on page 153
- "The Thread Pool Tree" on page 153

The Applications Tree

The following schematic shows the top and child nodes for the various components of enterprise applications. The nodes at which monitoring statistics are available are marked with an asterisk (*). For more information, refer to "EJB Container Statistics" on page 154.

EXAMPLE 16–1 Applications Node Tree Structure

```
applications
   |--- application1
   |    |--- ejb-module-1
   |    |        |--- ejb1 *
   |    |                 |--- cache (for entity/sfsb) *
   |    |                 |--- pool (for slsb/mdb/entity) *
   |    |                 |--- methods
   |    |                      |---method1 *
   |    |                      |---method2 *
   |    |                 |--- stateful-session-store (for sfsb)*
```

EXAMPLE 16–1 Applications Node Tree Structure *(Continued)*

```
|    |                  |--- timers (for s1sb/entity/mdb) *
|    |--- web-module-1
|    |        |--- virtual-server-1 *
|    |                       |---servlet1  *
|    |                       |---servlet2  *
|--- standalone-web-module-1
|    |        |----- virtual-server-2 *
|    |                       |---servlet3 *
|    |                       |---servlet4 *
|    |        |----- virtual-server-3 *
|    |                       |---servlet3 *(same servlet on different vs)
|    |                       |---servlet5 *
|--- standalone-ejb-module-1
|    |        |--- ejb2 *
|    |                  |--- cache (for entity/sfsb) *
|    |                  |--- pool (for slsb/mdb/entity) *
|    |                  |--- methods
|    |                       |--- method1 *
|    |                       |--- method2 *
|--- application2
```

The HTTP Service Tree

The nodes of the HTTP service are shown in the following schematic. The nodes at which monitoring information is available are marked with an asterisk (*). See "HTTP Service Statistics" on page 159.

EXAMPLE 16–2 HTTP Service Schematic (Platform Edition version)

```
http-service
         |--- virtual-server-1
              |--- http-listener-1 *
              |--- http-listener-2 *
         |--- virtual-server-2
              |--- http-listener-1 *
              |--- http-listener-2 *
```

EXAMPLE 16–3 HTTP Service Schematic (Enterprise Edition version)

```
http-service *
             |---connection-queue *
             |---dns *
             |---file-cache *
             |---keep-alive *
             |---pwc-thread-pool *
```

EXAMPLE 16–3 HTTP Service Schematic (Enterprise Edition version) *(Continued)*

```
|---virtual-server-1*
|        |--- request *
|---virtual-server-2*
|        |--- request *
```

The Resources Tree

The resources node holds monitorable attributes for pools such as the JDBC connection pool and connector connection pool. The following schematic shows the top and child nodes for the various resource components. The nodes at which monitoring statistics are available are marked with an asterisk (*). See "JDBC Connection Pools Statistics" on page 160.

EXAMPLE 16–4 Resources Schematic

```
resources
       |---connection-pool1(either connector-connection-pool or jdbc)*
       |---connection-pool2(either connector-connection-pool or jdbc)*
```

The Connector Service Tree

The connector services node holds monitorable attributes for pools such as the connector connection pool. The following schematic shows the top and child nodes for the various connector service components. The nodes at which monitoring statistics are available are marked with an asterisk (*). See "JMS/Connector Service Statistics" on page 161.

EXAMPLE 16–5 Connector Service Schematic

```
connector-service
       |--- resource-adapter-1
       |        |-- connection-pools
       |        |        |-- pool-1 (All pool stats for this pool)
       |        |-- work-management (All work mgmt stats for this RA)
```

The JMS Service Tree

The JMS services node holds monitorable attributes for pools such as the connector connection pool. The following schematic shows the top and child nodes for the various JMS service components. The nodes at which monitoring statistics are available are marked with an asterisk (*).

EXAMPLE 16–6 JMS Service Schematic

```
jms-service
       |-- connection-factories [AKA conn. pools in the RA world]
       |        |-- connection-factory-1 (All CF stats for this CF)
```

EXAMPLE 16–6 JMS Service Schematic *(Continued)*

```
        |-- work-management (All work mgmt stats for the MQ-RA)
```

The ORB Tree

The ORB node holds monitorable attributes for connection managers. The following schematic shows the top and child nodes for the ORB components. The nodes at which monitoring statistics are available are marked with an asterisk (*). See "Statistics for Connection Managers in an ORB" on page 163.

EXAMPLE 16–7 ORB Schematic

```
orb
        |--- connection-managers
        |        |--- connection-manager-1 *
        |        |--- connection-manager-1 *
```

The Thread Pool Tree

The thread pool node holds monitorable attributes for connection managers. The following schematic shows the top and child nodes for the ORB components. The nodes at which monitoring statistics are available are marked with an asterisk (*). See "Thread Pools Statistics" on page 163.

EXAMPLE 16–8 Thread Pool Schematic

```
thread-pools
        |    |--- thread-pool-1 *
        |    |--- thread-pool-2 *
```

Statistics for Monitored Components and Services

This section describes the monitoring statistics that are available:

■ “Production Web Container (PWC) Statistics” on page 168

EJB Container Statistics

EJB statistics are described in the following table.

TABLE 16–1 EJB Statistics

Attribute Name	Data Type	Description
`createcount`	CountStatistic	Number of times an EJB's `create` method is called.
`removecount`	CountStatistic	Number of times an EJB's `remove` method is called.
`pooledcount`	RangeStatistic	Number of entity beans in pooled state.
`readycount`	RangeStatistic	Number of entity beans in ready state.
`messagecount`	CountStatistic	Number of messages received for a message-driven bean.
`methodreadycount`	RangeStatistic	Number of stateful or stateless session beans that are in the `MethodReady` state.
`passivecount`	RangeStatistic	Number of stateful session beans that are in `Passive` state.

The statistics available for EJB method invocations are listed in the following table.

TABLE 16–2 EJB Method Statistics

Attribute Name	Data Type	Description
`methodstatistic`	TimeStatistic	Number of times an operation is called; the total time that is spent during the invocation, and so on.
`totalnumerrors`	CountStatistic	Number of times the method execution resulted in an exception. This is collected for stateless and stateful session beans and entity beans if monitoring is enabled for the EJB container.

TABLE 16–2 EJB Method Statistics *(Continued)*

Attribute Name	Data Type	Description
`totalnumsuccess`	CountStatistic	Number of times the method successfully executed. This is collected for stateless and stateful session beans and entity beans if monitoring enabled is true for EJB container.
`executiontime`	CountStatistic	Time (ms) spent executing the method for the last successful/unsuccessful attempt to execute the operation. This is collected for stateless and stateful session beans and entity beans if monitoring is enabled on the EJB container.

The statistics for EJB Session Stores are listed in the following table.

TABLE 16–3 EJB Session Store Statistics

Attribute Name	Data Type	Description
`currentSize`	RangeStatistic	Number of passivated or checkpointed sessions currently in the store.
`activationCount`	CountStatistic	Number of sessions activated from the store.
`activationSuccessCount`	CountStatistic	Number of sessions successfully activated from the store
`activationErrorCount`	CountStatistic	Time (ms) spent executing the method for the last successful/unsuccessful attempt to execute the operation. This is collected for stateless and stateful session beans and entity beans, if monitoring is enabled on the EJB container.
`passivationCount`	CountStatistic	Number of sessions passivated (inactivated) using this store.
`passivationSuccessCount`	CountStatistic	Number of sessions successfully passivated using this store.
`passivationErrorCount`	CountStatistic	Number of sessions that could not be passivated using this store.
`expiredSessionCount`	CountStatistic	Number of expired sessions that were removed by this store.

TABLE 16–3 EJB Session Store Statistics *(Continued)*

Attribute Name	Data Type	Description
`passivatedBeanSize`	CountStatistic	Total number of bytes passivated by this store, including total, minimum, and maximum.
`passivationTime`	CountStatistic	Time spent on passivating beans to the store, including the total, minimum, and maximum.
`checkpointCount (EE only)`	CountStatistic	Number of sessions checkpointed using this store.
`checkpointSuccessCount (EE only)`	CountStatistic	Number of sessions checkpointed successfully.
`checkpointErrorCount (EE only)`	CountStatistic	Number of sessions that couldn't be checkpointed.
`checkpointedBeanSize (EE only)`	ValueStatistic	Total number of beans checkpointed by the store.
`checkpointTime (EE only)`	TimeStatistic	Time spent on checkpointing beans to the store.

The statistics available for EJB pools are listed in the following table.

TABLE 16–4 EJB Pool Statistics

Attribute Name	Data Type	Description
`numbeansinpool`	BoundedRangeStatistic	Number of EJB's in the associated pool, providing an idea about how the pool is changing.
`numthreadswaiting`	BoundedRangeStatistic	Number of threads waiting for free beans, giving an indication of possible congestion of requests.
`totalbeanscreated`	CountStatistic	Number of beans created in associated pool since the gathering of data started.
`totalbeansdestroyed`	CountStatistic	Number of beans destroyed from associated pool since the gathering of data started.
`jmsmaxmessagesload`	CountStatistic	The maximum number of messages to load into a JMS session at one time for a message-driven bean to serve. Default is 1. Applies only to pools for message driven beans.

The statistics available for EJB caches are listed in the following table.

TABLE 16–5 EJB Cache Statistics

Attribute Name	Data Type	Description
cachemisses	BoundedRangeStatistic	The number of times a user request does not find a bean in the cache.
cachehits	BoundedRangeStatistic	The number of times a user request found an entry in the cache.
numbeansincache	BoundedRangeStatistic	The number of beans in the cache. This is the current size of the cache.
numpassivations	CountStatistic	Number of passivated beans. Applies only to stateful session beans.
numpassivationerrors	CountStatistic	Number of errors during passivation. Applies only to stateful session beans.
numexpiredsessionsremoved	CountStatistic	Number of expired sessions removed by the cleanup thread. Applies only to stateful session beans.
numpassivationsuccess	CountStatistic	Number of times passivation completed successfully. Applies only to stateful session beans.

The statistics available for Timers are listed in the following table.

TABLE 16–6 Timer Statistics

Statistic	Data Type	Description
numtimerscreated	CountStatistic	Number of timers created in the system.
numtimersdelivered	CountStatistic	Number of timers delivered by the system.
numtimersremoved	CountStatistic	Number of timers removed from the system.

Web Container Statistics

The web container fits into the tree of objects as shown in "The Applications Tree" on page 150. Web container statistics are displayed for each individual web application. Statistics available for the web container for Servlets are shown in "Web Container Statistics" on page 157 and statistics available for web modules are shown in "Web Container Statistics" on page 157.

TABLE 16–7 Web Container (Servlet) Statistics

Statistic	Units	Data Type	Comments
`errorcount`	Number	CountStatistic	Cumulative number of cases where the response code is greater than or equal to 400.
`maxtime`	Milliseconds	CountStatistic	The maximum amount of time the web container waits for requests.
`processingtime`	Milliseconds	CountStatistic	Cumulative value of the amount of time required to process eac request. The processing time is the average of request processin times divided by the request count.
`requestcount`	Number	CountStatistic	The total number of requests processed so far.

Statistics available for web modules are shown in "Web Container Statistics" on page 157.

TABLE 16–8 Web Container (Web Module) Statistics

Statistic	Data Type	Comments
`jspcount`	CountStatistic	Number of JSP pages that have been loaded in the web module.
`jspreloadcount`	CountStatistic	Number of JSP pages that have been reloaded in the web module.
`sessionstotal`	CountStatistic	Total number of sessions that have been created for the web module.
`activesessionscurrent`	CountStatistic	Number of currently active sessions for the web module.
`activesessionshigh`	CountStatistic	Maximum number of concurrently active sessions for the web module.
`rejectedsessionstotal`	CountStatistic	Total number of rejected sessions for the web module. This is the number of sessions that were not created because the maximum allowed number of sessions were active.
`expiredsessionstotal`	CountStatistic	Total number of expired sessions for the web module.
`sessionsize (EE only)`	AverageRangeStatistic	Size of the session for the web module. Value is either high, low, or average, or is in bytes for serialized sessions.

TABLE 16–8 Web Container (Web Module) Statistics *(Continued)*

Statistic	Data Type	Comments
`containerlatency (EE only)`	AverageRangeStatistic	Latency for the web container's part of the overall latency request. Value is either high, low, or average.
`sessionpersisttime (EE only)`	AverageRangeStatistic	Time (in ms, low, high, or average) taken to persist HTTP session state to back-end store for the web module.
`cachedsessionscurrent (EE only)`	CountStatistic	Current number of sessions cached in memory for the web module.
`passivatedsessionscurrent (EE only)`	CountStatistic	Current number of sessions passivated for the web module.

HTTP Service Statistics

The statistics available for the HTTP service are shown in "HTTP Service Statistics" on page 159. These statistics are applicable to the Platform Edition only. For statistics for the HTTP Service on the Enterprise Edition, see "Production Web Container (PWC) Statistics" on page 168.

TABLE 16–9 HTTP Service Statistics (applicable to Platform Edition only)

Statistic	Units	Data Type	Comments
`bytesreceived`	Bytes	CountStatistic	The cumulative value of the bytes received by each of the request processors.
`bytessent`	Bytes	CountStatistic	The cumulative value of the bytes sent by each of the request processors.
`currentthreadcount`	Number	CountStatistic	The number of processing threads currently in the listener thread pool.
`currentthreadsbusy`	Number	CountStatistic	The number of request processing threads currently in use in the listener thread pool serving requests.
`errorcount`	Number	CountStatistic	The cumulative value of the error count, which represents the number of cases where the response code is greater than or equal to 400.
`maxsparethreads`	Number	CountStatistic	The maximum number of unused response processing threads that can exist.

TABLE 16–9 HTTP Service Statistics (applicable to Platform Edition only) *(Continued)*

Statistic	Units	Data Type	Comments
`minsparethreads`	Number	CountStatistic	The minimum number of unused response processing threads that can exist.
`maxthreads`	Number	CountStatistic	The maximum number of request processing threads created by the listener.
`maxtime`	Milliseconds	CountStatistic	The maximum amount of time for processing threads.
`processing-time`	Milliseconds	CountStatistic	The cumulative value of the times taken to process each request. The processing time is the average of request processing times divided by the request count.
`request-count`	Number	CountStatistic	The total number of requests processed so far.

JDBC Connection Pools Statistics

Monitor JDBC resources to measure performance and capture resource usage at runtime. As the creation of JDBC connections are expensive and frequently cause performance bottlenecks in applications, it is crucial to monitor how a JDBC connection pool is releasing and creating new connections and how many threads are waiting to retrieve a connection from a particular pool.

The statistics available for the JDBC connection pool are shown in the following table.

TABLE 16–10 JDBC Connection Pool Statistics

Statistic	Units	Data Type	Description
`numconnfailedvalidation`	Number	CountStatistic	The total number of connections in the connection pool that failed validation from the start time until the last sample time.
`numconnused`	Number	RangeStatistic	Provides connection usage statistics. The total number of connections that are currently being used, as well as information about the maximum number of connections that were used (the high water mark).
`numconnfree`	Number	RangeStatistic	The total number of free connections in the pool as of the last sampling.

TABLE 16–10 JDBC Connection Pool Statistics *(Continued)*

Statistic	Units	Data Type	Description
numconntimedout	Number	BoundedRangeStatistic	The total number of connections in the pool that timed out between the start time and the last sample time.
averageconnwaittime	Number	CountStatistic	Indicates the average wait time of connections for successful connection request attempts to the connector connection pool.
waitqueuelength	Number	CountStatistic	Number of connection requests in the queue waiting to be serviced.
connectionrequestwaittime		RangeStatistic	The longest and shortest wait times of connection requests. The current value indicates the wait time of the last request that was serviced by the pool.
numconncreated	Milliseconds	CountStatistic	The number of physical connections that were created since the last reset.
numconndestroyed	Number	CountStatistic	Number of physical connections that were destroyed since the last reset.
numconnacquired	Number	CountStatistic	Number of logical connections acquired from the pool.
numconnreleased	Number	CountStatistic	Number of logical connections released to the pool.

JMS/Connector Service Statistics

The statistics available for the connector connection pools are shown in "JMS/Connector Service Statistics" on page 161. Statistics for Connector Work Management are shown in "JMS/Connector Service Statistics" on page 161.

TABLE 16–11 Connector Connection Pool Statistics

Statistic	Units	Data Type	Description
numconnfailedvalidation	Number	CountStatistic	The total number of connections in the connection pool that failed validation from the start time until the last sample time.

TABLE 16–11 Connector Connection Pool Statistics *(Continued)*

Statistic	Units	Data Type	Description
numconnused	Number	RangeStatistic	Provides connection usage statistics. The total number of connections that are currently being used, as well as information about the maximum number of connections that were used (the high water mark).
numconnfree	Number	RangeStatistic	The total number of free connections in the pool as of the last sampling.
numconntimedout	Number	CountStatistic	The total number of connections in the pool that timed out between the start time and the last sample time.
averageconnwaittime	Number	CountStatistic	Average wait time of connections before they are serviced by the connection pool.
waitqueuelenght	Number	CountStatistic	Number of connection requests in the queue waiting to be serviced.
connectionrequestwaittime		RangeStatistic	The longest and shortest wait times of connection requests. The current value indicates the wait time of the last request that was serviced by the pool.
numconncreated	Milliseconds	CountStatistic	The number of physical connections that were created since the last reset.
numconndestroyed	Number	CountStatistic	Number of physical connections that were destroyed since the last reset.
numconnacquired	Number	CountStatistic	Number of logical connections acquired from the pool.
numconnreleased	Number	CountStatistic	Number of logical connections released to the pool.

Statistics available for Connector Work Management are listed in "JMS/Connector Service Statistics" on page 161.

TABLE 16–12 Connector Work Management Statistics

Statistic	Data Type	Description
activeworkcount	RangeStatistic	Number of work objects executed by the connector.
waitqueuelength	RangeStatistic	Number of work objects waiting in the queue before executing.
workrequestwaittime	RangeStatistic	Longest and shortest wait of a work object before it gets executed.

TABLE 16–12 Connector Work Management Statistics *(Continued)*

Statistic	Data Type	Description
`submittedworkcount`	CountStatistic	Number of work objects submitted by a connector module.
`rejectedworkcount`	CountStatistic	Number of work objects rejected by the Application Server.
`completedworkcount`	CountStatistic	Number of work objects that were completed.

Statistics for Connection Managers in an ORB

The statistics available for the connection manager in an ORB are listed in "Statistics for Connection Managers in an ORB" on page 163.

TABLE 16–13 Connection Manager (in an ORB) Statistics

Statistic	Units	Data Type	Description
`connectionsidle`	Number	CountStatistic	Provides total number of connections that are idle to the ORB.
`connectionsinuse`	Number	CountStatistic	Provides total number of connections in use to the ORB.
`totalconnections`	Number	BoundedRangeStatistic	Total number of connections to the ORB.

Thread Pools Statistics

The statistics available for the thread pool are shown in the following table.

TABLE 16–14 Thread Pool Statistics

Statistic	Units	Data Type	Description
`averagetimeinqueue`	Milliseconds	RangeStatistic	The average amount of time in milliseconds a request waited in the queue before getting processed.
`averageworkcompletion-time`	Milliseconds	RangeStatistic	The average amount of time taken to complete an assignment, in milliseconds.
`currentnumberofthreads`	Number	BoundedRangeStatistic	Current number of request processing threads.
`numberofavailablethreads`	Number	CountStatistic	The number of threads that are available.
`numberofbusythreads`	Number	CountStatistic	The number of threads that are busy.

TABLE 16–14 Thread Pool Statistics *(Continued)*

Statistic	Units	Data Type	Description
`totalworkitemsadded`	Number	CountStatistic	The total number of work items added so far to the work queue.

Transaction Service Statistics

The transaction service allows the client to freeze the transaction subsystem in order to roll back transactions and determine the transactions that are in process at the time of the freeze. The statistics available for the transaction service are shown in the following table.

TABLE 16–15 Transaction Service Statistics

Statistic	Data Type	Description
`activecount`	CountStatistic	Number of transactions currently active.
`activeids`	StringStatistic	The ID's of the transactions that are currently active. Every such transaction can be rolled back after freezing the transaction service.
`committedcount`	CountStatistic	Number of transactions that have been committed.
`rolledbackcount`	CountStatistic	Number of transactions that have been rolled back.
`state`	StringStatistic	Indicates whether or not the transaction has been frozen.

Java Virtual Machine (JVM) Statistics

The JVM has monitorable attributes that are always enabled. The statistics available for the JVM are shown in the following table.

TABLE 16–16 JVM Statistics

Statistic	Data Type	Description
`heapsize`	BoundedRangeStatistic	The resident memory footprint with the higher and lower bounds of the JVM's memory heap size.
`uptime`	CountStatistic	The amount of time the JVM has been running.

JVM Statistics in J2SE 5.0

If the Application Server is configured to run on J2SE version 5.0 or higher, additional monitoring information can be obtained from the JVM. Set the monitoring level to LOW to enable the display of this additional information. Set the monitoring level to HIGH to also view information pertaining to each live thread in the system. More information on the additional

monitoring features available in J2SE 5.0 is available in a document titled *Monitoring and Management for the Java Platform*, which is available from `http://java.sun.com/j2se/1.5.0/docs/guide/management/`.

The J2SE 5.0 monitoring tools are discussed at `http://java.sun.com/j2se/1.5.0/docs/tooldocs/#manage`.

The statistics available for class loading in the JVM in J2SE 5.0 are shown in "JVM Statistics in J2SE 5.0" on page 164.

TABLE 16–17 JVM Statistics for J2SE 5.0 - Class Loading

Statistic	Data Type	Description
`loadedclasscount`	CountStatistic	Number of classes that are currently loaded in the JVM.
`totalloadedclasscount`	CountStatistic	Total number of classes that have been loaded since the JVM bega execution.
`unloadedclasscount`	CountStatistic	Number of classes that have been unloaded from the JVM since th began execution.

The statistics available for compilation in the JVM in J2SE 5.0 are shown in "JVM Statistics in J2SE 5.0" on page 164.

TABLE 16–18 JVM Statistics for J2SE 5.0 - Compilation

Statistic	Data Type	Description
`totalcompilationtime`	CountStatistic	Accumulated time (in milliseconds) spent in compilation.

The statistics available for garbage collection in the JVM in J2SE 5.0 are shown in "JVM Statistics in J2SE 5.0" on page 164.

TABLE 16–19 JVM Statistics for J2SE 5.0 - Garbage Collection

Statistic	Data Type	Description
`collectioncount`	CountStatistic	Total number of collections that have occurred.
`collectiontime`	CountStatistic	Accumulated collection time (in milliseconds).

The statistics available for memory in the JVM in J2SE 5.0 are shown in "JVM Statistics in J2SE 5.0" on page 164.

TABLE 16–20 JVM Statistics for J2SE 5.0 - Memory

Statistic	Data Type	Description
`objectpendingfinalizationcount`	CountStatistic	Approximate number of objects that are pending finalization.
`initheapsize`	CountStatistic	Size of the heap initially requested by the JVM.
`usedheapsize`	CountStatistic	Size of the heap currently in use.
`maxheapsize`	CountStatistic	Maximum amount of memory (in bytes) that can be used for memory management.
`committedheapsize`	CountStatistic	Amount of memory (in bytes) that is committed for the JVM to use.
`initnonheapsize`	CountStatistic	Size of the non-heap area initially requested by the JVM.
`usednonheapsize`	CountStatistic	Size of the non-heap area currently in use.
`maxnonheapsize`	CountStatistic	Maximum amount of memory (in bytes) that can be used for memory management.
`committednonheapsize`	CountStatistic	Amount of memory (in bytes) that is committed for the JVM to use.

The statistics available for the operating system in the JVM in J2SE 5.0 are shown in "JVM Statistics in J2SE 5.0" on page 164.

TABLE 16–21 JVM Statistics for J2SE 5.0 - Operating System

Statistic	Data Type	Description
`arch`	StringStatistic	Operating system architecture.
`availableprocessors`	CountStatistic	Number of processors available to the JVM.
`name`	StringStatistic	Operating system name.
`version`	StringStatistic	Operating system version.

The statistics available for the runtime in the JVM in J2SE 5.0 are shown in "JVM Statistics in J2SE 5.0" on page 164.

TABLE 16–22 JVM Statistics for J2SE 5.0 - Runtime

Statistic	Data Type	Description
`name`	StringStatistic	Name representing the running JVM
`vmname`	StringStatistic	JVM implementation name.
`vmvendor`	StringStatistic	JVM implementation vendor.

TABLE 16–22 JVM Statistics for J2SE 5.0 - Runtime *(Continued)*

Statistic	Data Type	Description
`vmversion`	StringStatistic	JVM implementation version.
`specname`	StringStatistic	JVM specification name.
`specvendor`	StringStatistic	JVM specification vendor.
`specversion`	StringStatistic	JVM specification version.
`managementspecversion`	StringStatistic	Management spec. version implemented by the JVM.
`classpath`	StringStatistic	Classpath that is used by the system class loader to search for class fi
`librarypath`	StringStatistic	Java library path.
`bootclasspath`	StringStatistic	Classpath that is used by the bootstrap class loader to search for clas
`inputarguments`	StringStatistic	Input arguments passed to the JVM. Does not include the argumen `main` method.
`uptime`	CountStatistic	Uptime of the JVM (in milliseconds).

The statistics available for `ThreadInfo` in the JVM in J2SE 5.0 are shown in "JVM Statistics in J2SE 5.0" on page 164.

TABLE 16–23 JVM Statistics for J2SE 5.0 - Thread Info

Statistic	Data Type	Description
`threadid`	CountStatistic	Id of the thread.
`threadname`	StringStatistic	Name of the thread.
`threadstate`	StringStatistic	State of the thread.
`blockedtime`	CountStatistic	Time elapsed (in milliseconds) since the thread entered the `BLOCKED` Returns -1 if thread contention monitoring is disabled.
`blockedcount`	CountStatistic	Total number of times that the thread entered the `BLOCKED` state.
`waitedtime`	CountStatistic	Elapsed time (in milliseconds) that the thread has been in a `WAITING` Returns -1 if thread contention monitoring is disabled.
`waitedcount`	CountStatistic	Total number of times the thread was in `WAITING` or `TIMED_WAITING`
`lockname`	StringStatistic	String representation of the monitor lock that the thread is blocked or waiting to be notified through the `Object.wait` method.
`lockownerid`	CountStatistic	Id of the thread that holds the monitor lock of an object on which th is blocking.

TABLE 16–23 JVM Statistics for J2SE 5.0 - Thread Info *(Continued)*

Statistic	Data Type	Description
`lockownername`	StringStatistic	Name of the thread that holds the monitor lock of the object this thread is blocking on.
`stacktrace`	StringStatistic	Stack trace associated with this thread.

The statistics available for threads in the JVM in J2SE 5.0 are shown in "JVM Statistics in J2SE 5.0" on page 164.

TABLE 16–24 JVM Statistics for J2SE 5.0 - Threads

Statistic	Data Type	Description
`threadcount`	CountStatistic	Current number of live daemon and non-daemon threads.
`peakthreadcount`	CountStatistic	Peak live thread count since the JVM started or the peak was reset.
`totalstartedthreadcount`	CountStatistic	Total number of threads created and/or started since the JVM started.
`daemonthreadcount`	CountStatistic	Current number of live daemon threads.
`allthreadids`	StringStatistic	List of all live thread ids.
`currentthreadcputime`	CountStatistic	CPU time for the current thread (in nanoseconds) if CPU time measurement is enabled. If CPU time measurement is disabled, returns -1.
`monitordeadlockedthreads`	StringStatistic	List of thread ids that are monitor deadlocked.

Production Web Container (PWC) Statistics

Statistics are available for the following PWC components and services on the Enterprise Edition (EE) of the Application Server:

- "Production Web Container (PWC) Statistics" on page 168, PWC Virtual Server
- "Production Web Container (PWC) Statistics" on page 168, PWC Request
- "Production Web Container (PWC) Statistics" on page 168, PWC File Cache
- "Production Web Container (PWC) Statistics" on page 168, PWC Keep Alive
- "Production Web Container (PWC) Statistics" on page 168, PWC DNS
- "Production Web Container (PWC) Statistics" on page 168, PWC Thread Pool
- "Production Web Container (PWC) Statistics" on page 168, PWC Connection Queue
- "Production Web Container (PWC) Statistics" on page 168, PWC HTTP Service

Statistics for PWC virtual servers are listed in "Production Web Container (PWC) Statistics" on page 168.

TABLE 16–25 PWC Virtual Server Statistics (EE only)

Attribute Name	Data Type	Description
`id`	StringStatistic	The ID of the virtual server.
`mode`	StringStatistic	The mode the virtual server is in. Options include `unknown` or `active`.
`hosts`	StringStatistic	Name of the hosts serviced by this virtual server.
`interfaces`	StringStatistic	Type of interfaces (listeners) for which the virtual server is configured.

The statistics available for PWC requests are listed in the following table.

TABLE 16–26 PWC Request Statistics (EE only)

Attribute Name	Data Type	Description
`method`	StringStatistic	Method used for request.
`uri`	StringStatistic	Last URI served.
`countrequests`	CountStatistic	Number of requests served.
`countbytestransmitted`	CountStatistic	Number of bytes transmitted, or 0 if this information is not available
`countbytesreceived`	CountStatistic	Number of bytes received, or 0 if this information is not available.
`ratebytesreceived`	CountStatistic	Rate at which data was received over some server-defined interval, or 0 if this information is not available
`maxbytestransmissionrate`	CountStatistic	Maximum rate at which data was transmitted over some server-defined interval, or 0 if this information is not available.
`countopenconnections`	CountStatistic	Number of connections that are currently open, or 0 if this information is not available.
`maxopenconnections`	CountStatistic	Maximum number of concurrently open connections, or 0 if this information is not available.
`count2xx`	CountStatistic	Total number of responses of code 2XX.
`count3xx`	CountStatistic	Total number of responses of code 3XX.
`count4xx`	CountStatistic	Total number of responses of code 4XX.

TABLE 16–26 PWC Request Statistics (EE only) *(Continued)*

Attribute Name	Data Type	Description
count5xx	CountStatistic	Total number of responses of code 5XX.
countother	CountStatistic	Total number of responses with other response codes.
count200	CountStatistic	Total number of responses of code 200.
count302	CountStatistic	Total number of responses of code 302.
count304	CountStatistic	Total number of responses of code 304.
count400	CountStatistic	Total number of responses of code 400.
count401	CountStatistic	Total number of responses of code 401.
count403	CountStatistic	Total number of responses of code 403.
count404	CountStatistic	Total number of responses of code 404.
count503	CountStatistic	Total number of responses of code 503.

The cache information section provides information on how the file cache is being used. Statistics for PWC file caches are listed in the following table.

TABLE 16–27 PWC File Cache Statistics (EE only)

Attribute Name	Data Type	Description
flagenabled	CountStatistic	Indicates whether the file cache is enabled. Valid values are 0 for no or 1 for yes.
secondsmaxage	CountStatistic	Maximum age of a valid cache entry, in seconds.
countentries	CountStatistic	Number of current cache entries. A single cache entry represents a single URI.
maxentries	CountStatistic	Maximum number of simultaneous cache entries.
countopenentries	CountStatistic	Number of entries associated with an open file.
maxopenentries	CountStatistic	Maximum number of simultaneous cache entries associated with an open file.
sizeheapcache	CountStatistic	Heap space used for cache content.
maxheapcachesize	CountStatistic	Maximum heap space used for cache file content.
sizemmapcache	CountStatistic	Amount of address space used by memory mapped file content.
maxmmapcachesize	CountStatistic	Maximum amount of address space used by the file cache for memory mapped fi content.
counthits	CountStatistic	Number of successful cache lookups.
countmisses	CountStatistic	Number of failed cache lookups.

TABLE 16–27 PWC File Cache Statistics (EE only) *(Continued)*

Attribute Name	Data Type	Description
countinfohits	CountStatistic	Number of times a file information lookup succeeded.
countinfomisses	CountStatistic	Number of misses on cached file information.
countcontenthits	CountStatistic	Number of hits on cached file content.
countcontentmisses	CountStatistic	Number of times a file information lookup failed.

This section provides information about the server's HTTP-level keep-alive system. The statistics available for PWC Keep Alive are listed in the following table.

TABLE 16–28 PWC Keep Alive Statistics (EE only)

Attribute Name	Data Type	Description
countconnections	CountStatistic	Number of connections in keep-alive mode.
maxconnections	CountStatistic	Maximum number of connections simultaneously allowed in keep-alive r
counthits	CountStatistic	The total number of times connections in keep-alive mode have subseque made a valid request.
countflushes	CountStatistic	The number of times keep-alive connections have been closed by the serv
countrefusals	CountStatistic	The number of times the server could not hand off the connection to a kee thread, possibly due to too many persistent connections.
counttimeouts	CountStatistic	The number of times the server terminated keep-alive connections as the connections timed out without any activity.
secondstimeout	CountStatistic	The time (in seconds) before idle keep-alive connections are closed.

The DNS Cache caches IP addresses and DNS names. The server's DNS cache is disabled by default. A single cache entry represents a singular IP address or DNS name lookup. The statistics available for PWC DNS are listed in the following table.

TABLE 16–29 PWC DNS Statistics (EE only)

Attribute Name	Data Type	Description
flagcacheenabled	CountStatistic	Indicates whether the DNS cache is enabled (on). Either 0 for off or 1 for on.
countcacheentries	CountStatistic	Number of DNS entries presently in the cache.
maxcacheentries	CountStatistic	Maximum number of DNS entries that can be accommodated by the cache.

TABLE 16–29 PWC DNS Statistics (EE only) *(Continued)*

Attribute Name	Data Type	Description
countcachehits	CountStatistic	Number of times a DNS cache lookup has succeeded.
countcachemisses	CountStatistic	Number of times a DNS cache lookup has failed.
flagasyncenabled	CountStatistic	Indicates whether asynchronous DNS lookups are enabled (on). Either 0 for off or 1 for on.
countasyncnamelookups	CountStatistic	Total number of asynchronous DNS name lookups.
countasyncaddrlookups	CountStatistic	Total number of asynchronous DNS address lookups.
countasynclookupsinprogress	CountStatistic	Number of asynchronous lookups in progress.

Statistics for PWC thread pools are listed in the following table.

TABLE 16–30 PWC Thread Pool Statistics (EE only)

Attribute Name	Data Type	Description
id	StringStatistic	ID of the thread pool.
countthreadsidle	CountStatistic	Number of request-processing threads currently idle.
countthreads	CountStatistic	Current number of request-processing threads.
maxthreads	CountStatistic	Maximum number of request processing threads that can exist concurrently.
countqueued	CountStatistic	Number of requests queued for processing by this thread pool.
peakqueued	CountStatistic	The largest number of requests in the queue simultaneously.
maxqueued	CountStatistic	Maximum number of requests that can be in the queue at one time.

The Connection Queue is the queue in which requests are held prior to being serviced. Statistics for the connection queue show the number of sessions in the queue and the average delay before the connection is accepted. Statistics for PWC connection queues are listed in the following table.

TABLE 16–31 PWC Connection Queue Statistics (EE only)

Attribute Name	Data Type	Description
`id`	StringStatistic	ID of the connection queue.
`counttotalconnections`	CountStatistic	Total number of connections that have been accepted.
`countqueued`	CountStatistic	Number of connections currently in the queue.
`peakqueued`	CountStatistic	Largest number of connections that were in the queue simultaneously.
`maxqueued`	CountStatistic	Maximum size of the connection queue.
`countoverflows`	CountStatistic	The number of times the queue has been too full to accommodate a connection.
`counttotalqueued`	CountStatistic	The total number of connections that have been queued. A given connection may be queued multiple times, so `counttotalqueued` may be greater than or equal to `counttotalconnections`.
`tickstotalqueued`	CountStatistic	The total number of ticks that connections have spent in the queue. A tick is a system-dependent unit of time.
`countqueued1minuteaverage`	CountStatistic	Average number of connections queued in the last 1 minute.
`countqueued5minuteaverage`	CountStatistic	Average number of connections queued in the last 5 minutes.
`countqueued15minuteaverage`	CountStatistic	Average number of connections queued in the last 15 minutes.

Statistics for PWC HTTP service are listed in the following table.

TABLE 16–32 PWC HTTP Service Statistics (EE only)

Attribute Name	Data Type	Description
`id`	StringStatistic	Instance name of the HTTP service.
`versionserver`	StringStatistic	Version number of the HTTP service.
`timestarted`	StringStatistic	Time the HTTP service was started (GMT).
`secondsrunning`	CountStatistic	Time (in seconds) since the HTTP service started.

TABLE 16–32 PWC HTTP Service Statistics (EE only) *(Continued)*

Attribute Name	Data Type	Description
`maxthreads`	CountStatistic	Maximum number of worker threads in each instance.
`maxvirtualservers`	CountStatistic	Maximum number of virtual servers that can be configured in each instance.
`flagprofilingenabled`	CountStatistic	Whether or not HTTP service performance profiling is enabled. Valid values are 0 or 1.
`flagvirtualserveroverflow`	CountStatistic	Indicates whether more than `maxvirtualservers` are configured. If this is set to 1 statistics are not being tracked for all virtual servers.
`load1minuteaverage`	CountStatistic	Average load for requests in the last 1 minute.
`load5minuteaverage`	CountStatistic	Average load for requests in the last 5 minutes.
`load15minuteaverage`	CountStatistic	Average load for requests in the last 15 minutes.
`ratebytestransmitted`	CountStatistic	The rate at which data is transmitted over some server-defined interval. The result is 0 when this information is not available.
`ratebytesreceived`	CountStatistic	The rate at which data is received over some server-defined interval. The result is 0 when this information is not available.

Enabling and Disabling Monitoring

In the Admin Console, you can configure the monitoring levels by selecting the Configurations node > Server instance node >Monitoring Page and choose the value for the services whose monitoring level is changing. By default, monitoring is turned off for all components and services.

For detailed steps on how to configure the monitoring levels, consult the Admin Console online help.

From the command line, use `asadmin set` to turn ON monitoring for various Application Server components. For example, run the following command to turn on monitoring for the HTTP service:

```
asadmin> set --user admin-user
server.monitoring-service.module-monitoring-levels.http-service=HIGH
```

Use the `get` command to find out what services and components currently have monitoring enabled.

```
asadmin> get --user admin-user server.monitoring-service.module-monitoring-levels.*
```

Returns:

```
server.monitoring-service.module-monitoring-levels.
connector-connection-pool = OFF
server.monitoring-service.module-monitoring-levels.
connector-service = OFF
server.monitoring-service.module-monitoring-levels.ejb-container = OFF
server.monitoring-service.module-monitoring-levels.http-service = HIGH
server.monitoring-service.module-monitoring-levels.jdbc-connection-pool = OFF
server.monitoring-service.module-monitoring-levels.jms-service = OFF
server.monitoring-service.module-monitoring-levels.jvm = OFF
server.monitoring-service.module-monitoring-levels.orb = OFF
server.monitoring-service.module-monitoring-levels.thread-pool = OFF
server.monitoring-service.module-monitoring-levels.transaction-service = OFF
server.monitoring-service.module-monitoring-levels.web-container = OFF
```

Use the `set` command to turn OFF monitoring.

For example, run the following command to disable monitoring for the HTTP service:

```
asadmin> set --user admin-user
server.monitoring-service.module-monitoring-levels.http-service=OFF
```

Similarly, to disable monitoring for other components, use the `set` command and specify `OFF` for the monitoring level.

Viewing Monitoring Data

In the Admin Console, you can view monitoring data by navigating to the Monitoring page for the standalone server instance and selecting the component or service that has been deployed onto the server instance for which monitoring is enabled. The monitoring data for the selected component or service displays below the View field. For a description of the monitorable properties.

Use the Administration Console to monitor remote applications and instances. For this to occur, the remote instance must be running and the configuration set. For detailed steps on viewing the monitoring data, consult the Admin Console online help.

To view monitoring data using command line utility, use the `asadmin list` and `asadmin get` commands followed by the dotted name of a monitorable object, as follows:

1. To view the names of the objects that can be monitored, use the `asadmin list` command.

   For example, to view a list of application components and subsystems that have monitoring enabled for the server instance, type the following command in a terminal window:

   ```
   asadmin> list --user adminuser --monitor server
   ```

   The preceding command returns a list of application components and subsystems that have monitoring enabled, for example:

```
server.resources
server.connector-service
server.orb
server.jms-service
server.jvm
server.applications
server.http-service
server.thread-pools
```

2. To display monitoring statistics for an application component or subsystem for which monitoring has been enabled, use the `asadmin get` command.

   To get the statistics, type the `asadmin get` command in a terminal window, specifying a name displayed by the `list` command in the preceding step. The following example attempts to get all attributes from a subsystem for a specific object:

   ```
   asadmin> get --user adminuser --monitor server.jvm.*
   ```

   The command returns the following attributes and data:

   ```
   server.jvm.dotted-name = server.jvm
   server.jvm.heapsize-current = 21241856
   server.jvm.heapsize-description = Provides statistical information about
      the JVM's memory heap size.
   server.jvm.heapsize-highwatermark = 21241856
   server.jvm.heapsize-lastsampletime = 1080232913938
   server.jvm.heapsize-lowerbound = 0
   server.jvm.heapsize-lowwatermark = 0
   server.jvm.heapsize-name = JvmHeapSize
   server.jvm.heapsize-starttime = 1080234457308
   server.jvm.heapsize-unit = bytes
   server.jvm.heapsize-upperbound = 518979584
   server.jvm.uptime-count = 1080234457308
   server.jvm.uptime-description = Provides the amount of time the JVM has
      been running.
   server.jvm.uptime-lastsampletime = 1080234457308
   server.jvm.uptime-name = JvmUpTime
   server.jvm.uptime-starttime = 1080232913928
   server.jvm.uptime-unit = milliseconds
   ```

Understanding and Specifying Dotted Names

In the `asadmin list` and `get` commands, specify the dotted name of monitorable objects. All child objects are addressed using the dot (.) character as separator, thus these are referred to as *dotted names*. If a child node is of singleton type, then only the monitoring object type is needed to address the object, otherwise a name of the form `type.name` is needed to address the object.

For example, `http-service` is one of the valid monitorable object types and is a singleton. To address a singleton child node representing the `http-service` of instance `server`, the dotted name is:

```
server.http-service
```

Another example, `application`, is a valid monitorable object type and is not a singleton. To address a non-singleton child node representing, for example, the application `PetStore`, the dotted name is:

```
server.applications.petstore
```

The dotted names can also address specific attributes in monitorable objects. For example, `http-service` has a monitorable attribute called `bytesreceived-lastsampletime`. The following name addresses the `bytesreceived` attribute:

```
server.http-service.server.http-listener-1.
  bytesreceived-lastsampletime
```

The administrator is not expected to know the valid dotted names for `asadmin list` and `get` commands. The `list` command displays available monitorable objects, while the `get` command used with a wildcard parameter allows the inspection of all available attributes on any monitorable object.

The underlying assumptions for using the `list` and `get` commands with dotted names are:

- Any `list` command that has a dotted name that is **not** followed by a wildcard (`*`) gets as its result the current node's immediate children. For example, `list --user` *adminuser* `--monitor server` lists all immediate children belonging to the `server` node.
- Any `list` command that has a dotted name followed by a wildcard of the form `.*` gets as its result a hierarchical tree of children nodes from the current node. For example, `list --user` *adminuser* `--monitor server.applications.*` lists all children of `applications` and their subsequent child nodes and so on.
- Any `list` command that has a dotted name preceded or followed by a wildcard of the form **dottedname* or *dotted* * *name* or **`dotted name *`** gets as its result all nodes and their children matching the regular expression created by the provided matching pattern.
- A `get` command followed by a `.*` or a `*` gets as its result the set of attributes and their values belonging to the current node to be matched.

For more information, see "Expected Output for `list` and `get` Commands at All Levels" on page 183.

Examples of the `list` Command

The `list` command provides information about the application components and subsystems currently being monitored for the specified server instance name. Using this command, you can see the monitorable components and subcomponents for a server instance. For a more complete listing of `list` examples, see "Expected Output for `list` and `get` Commands at All Levels" on page 183.

Example 1

```
asadmin> list --user admin-user --monitor server
```

The preceding command returns a list of application components and subsystems that have monitoring enabled, for example:

```
server.resources
server.orb
server.jvm
server.jms-service
server.connector-service
server.applications
server.http-service
server.thread-pools
```

It is also possible to list applications that are currently monitored in the specified server instance. This is useful when particular monitoring statistics are sought from an application using the `get` command.

Example 2

```
asadmin> list --user admin-user --monitor server.applications
```

Returns:

```
server.applications.adminapp
 server.applications.admingui
server.applications.myApp
```

Examples of the `get` Command

The `get` command retrieves the following monitored information:

- All attribute(s) monitored within a component or subsystem
- Specific attribute monitored within a component or subsystem

  When an attribute is requested that does not exist for a particular component or subsystem, an error is returned. Similarly, when a specific attribute is requested that is not active for a component or subsystem, an error is returned.

Refer to "Expected Output for `list` and `get` Commands at All Levels" on page 183 for more information on the use of the `get` command.

Example 1

Attempt to get all attributes from a subsystem for a specific object:

```
asadmin> get --user admin-user --monitor server.jvm.*
```

Returns:

```
server.jvm.dotted-name= server.jvm
server.jvm.heapsize-current = 21241856
server.jvm.heapsize-description = Provides statistical information about
   the JVM's memory heap size.
server.jvm.heapsize-highwatermark = 21241856
server.jvm.heapsize-lastsampletime = 1080232913938
server.jvm.heapsize-lowerbound = 0
server.jvm.heapsize-lowwatermark = 0
server.jvm.heapsize-name = JvmHeapSize
server.jvm.heapsize-starttime = 1080234457308
server.jvm.heapsize-unit = bytes
server.jvm.heapsize-upperbound = 518979584
server.jvm.uptime-count = 1080234457308
server.jvm.uptime-description = Provides the amount of time the JVM has
   been running.
server.jvm.uptime-lastsampletime = 1080234457308
server.jvm.uptime-name = JvmUpTime
server.jvm.uptime-starttime = 1080232913928
server.jvm.uptime-unit = milliseconds
```

Example 2

Attempt to get all attributes from a J2EE application:

```
asadmin> get --user admin-user --monitor server.applications.myJ2eeApp.*
```

Returns:

```
No matches resulted from the wildcard expression.
CLI137 Command get failed.
```

There are no monitorable attributes exposed at the J2EE-application level, therefore this reply displays.

Example 3

Attempt to get a specific attribute from a subsystem:

```
asadmin> get --user admin-user --monitor server.jvm.uptime-lastsampletime
```

Returns:

```
server.jvm.uptime-lastsampletime = 1093215374813
```

Example 4

Attempt to get an unknown attribute from within a subsystem attribute:

```
asadmin> get --user admin-user --monitor server.jvm.badname
```

Returns:

```
No such attribute found from reflecting the corresponding Stats
interface: [badname]
CLI137 Command get failed.
```

Using the `PetStore` Example

The following example illustrates how the `asadmin` tool might be used for monitoring purposes.

A user wants to inspect the number of calls made to a method in the sample `PetStore` application after it has been deployed onto the Application Server. The instance onto which it has been deployed is named `server`. A combination of the `list` and `get` commands are used to access desired statistics on a method.

1. Start the Application Server and the `asadmin` tool.
2. Set some useful environment variables to avoid entering them for every command:

   ```
   asadmin> export AS_ADMIN_USER=admin AS_ADMIN_PASSWORD=admin123
   asadmin>export AS_ADMIN_HOST=localhost AS_ADMIN_PORT=4849
   ```

3. List monitorable components for instance `server`:

   ```
   asadmin> list --user adminuser --monitor server*
   ```

   Returns (output will be similar to):

   ```
   server
   server.applications
   server.applications.CometEJB
   server.applications.ConverterApp
   server.applications.petstore
   ```

```
server.http-service
server.resources
server.thread-pools
```

The list of monitorable components includes `thread-pools`, `http-service`, `resources`, and all deployed (and enabled) `applications`.

4. List the monitorable subcomponents in the `PetStore` application (`-m` can be used instead of `--monitor`):

   ```
   asadmin> list -m server.applications.petstore
   ```

   Returns:

   ```
   server.applications.petstore.signon-ejb_jar
   server.applications.petstore.catalog-ejb_jar
   server.applications.petstore.uidgen-ejb_jar
   server.applications.petstore.customer-ejb_jar
   server.applications.petstore.petstore-ejb_jar
   server.applications.petstore.petstore\.war
   server.applications.petstore.AsyncSenderJAR_jar
   server.applications.petstore.cart-ejb_jar
   ```

5. List the monitorable subcomponents in the EJB module `signon-ejb_jar` of the `PetStore` application:

   ```
   asadmin> list -m server.applications.petstore.signon-ejb_jar
   ```

   Returns:

   ```
   server.applications.petstore.signon-ejb_jar.SignOnEJB
   server.applications.petstore.signon-ejb_jar.UserEJB
   ```

6. List the monitorable subcomponents in the entity bean `UserEJB` for the EJB module `signon-ejb_jar` of the `PetStore` application:

   ```
   asadmin> list -m server.applications.petstore.signon-ejb_jar.UserEJB
   ```

   Returns (with dotted name removed for space considerations):

   ```
   server.applications.petstore.signon-ejb_jar.UserEJB.bean-cache
   server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods
   server.applications.petstore.signon-ejb_jar.UserEJB.bean-pool
   ```

7. List the monitorable subcomponents in the method `getUserName` for the entity bean `UserEJB` in the EJB module `signon-ejb_jar` of the `PetStore` application:

   ```
   asadmin> list -m
   server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.getUserName
   ```

   Returns:

   ```
   Nothing to list at server.applications.petstore.signon-ejb_jar.
   UserEJB.bean-methods.getUserName. To get the valid names beginning with a
   string, use the wildcard "*" character. For example, to list all names
   ```

```
that begin with "server", use "list server*".
```

8. There are no monitorable subcomponents for methods. Get all monitorable statistics for the method `getUserName`.

```
asadmin> get -m
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.getUserName.*
```

Returns:

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-description = Provides the time in milliseconds
   spent during the last successful/unsuccessful attempt to execute the
   operation.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-lastsampletime = 1079981809259
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-name = ExecutionTime
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-unit = count
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-description = Provides the number of times an
   operation was called, the total time that was spent during the
   invocation and so on.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-lastsampletime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-maxtime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-mintime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-name = ExecutionTime
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-totaltime = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.methodstatistic-unit =
 server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-description = Provides the total number of errors
   that occured during invocation or execution of an operation.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
```

```
getUserName.totalnumerrors-lastsampletime = 1079981809273
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-name = TotalNumErrors
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumerrors-unit = count
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-count = 0
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-description = Provides the total number of
   successful invocations of the method.
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-lastsampletime = 1079981809255
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-name = TotalNumSuccess
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-starttime = 1079980593137
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.totalnumsuccess-unit = count
```

9. To also get a specific statistic, such as execution time, use a command such as the following:

```
asadmin> get -m server.applications.petstore.signon-ejb_jar.
UserEJB.bean-methods.getUserName.executiontime-count
```

Returns:

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-count = 1
```

Expected Output for `list` and `get` Commands at All Levels

The following tables show the command, dotted name, and corresponding output at each level of the tree.

TABLE 16–33 Top Level

Command	Dotted Name	Output
`list -m`	`server`	`server.applicationsserver.thread-poolsserver.resourcesserver.http-serviceserver.transaction-serviceserver.orb.connection-managersserver.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnectionsserver.jvm`

TABLE 16–33 Top Level *(Continued)*

Command	Dotted Name	Output
`list -m`	`server.*`	Hierarchy of child nodes below this node.
`get -m`	`server.*`	No output except a message saying there are no attributes at this node.

The following table shows the command, dotted name, and corresponding output for the applications level.

TABLE 16–34 Applications Level

Command	Dotted Name	Output

`list -m`	`server.applications`
	or
	`*applications`
		`appl1app2web-module1_warejb-module2_jar...`

`list -m`	`server.applications.*`
	or
	`*applications.*`
		Hierarchy of child nodes below this node.

`get -m`	`server.applications.*`
	or
	`*applications.*`
		No output except message saying there are no attributes at this node.

The following table shows the command, dotted name, and corresponding output for stand-alone modules and enterprise applications at the applications level.

TABLE 16–35 Applications - Enterprise Applications and Standalone Modules

Command	Dotted Name	Output

`list -m`	`server.applications.app1`
	or
	`*app1`
	Note: this level is only applicable if an enterprise application has been deployed. It is not applicable if a standalone module is deployed.
		`ejb-module1_jarweb-module2_warejb-module3_jarweb-module3_war...`

TABLE 16–35 Applications - Enterprise Applications and Standalone Modules *(Continued)*

Command	Dotted Name	Output

`list -m`	`server.applications.app1.*`
	or
	`*app1.*`
		Hierarchy of child nodes below this node.

`get -m`	`server.applications.app1.*`
	or
	`*app1.*`
		No output except message saying there are no attributes at this node.

`list -m`	`server.applications.app1.ejb-module1_jar`
	or
	`*ejb-module1_jar`
	or
	`server.applications.ejb-module1_jar`
		`bean1bean2bean3...`

`list -m`	`server.applications.app1.ejb-module1_jar`
	or
	`*ejb-module1_jar`
	or
	`server.applications.ejb-module1_jar`
		Hierarchy of child nodes below this node.

`get -m`	`server.applications.app1.ejb-module1_jar.*`
	or
	`*ejb-module1_jar.*`
	or
	`server.applications.ejb-module1_jar.*`
		No output except message saying there are no attributes at this node.

`list -m`	`server.applications.app1.`
	`ejb-module1_jar.bean1`
	Note: In standalone modules, the node containing the application name (`app1` in this example) will not appear.
		List of child nodes:
		`bean-poolbean-cachebean-method`

`list -m`	`server.applications.app1.`
	`ejb-module1_jar.bean1`
	Note: In standalone modules, the node containing the application name (`app1` in this example) will not appear.
		Hierarchy of child nodes and a list of all attributes for this node and for any subsequent child nodes.

TABLE 16–35 Applications - Enterprise Applications and Standalone Modules *(Continued)*

Command	Dotted Name	Output
`get -m`	`server.applications.app1.ejb-module1_jar.bean1.*` Note: In standalone modules, the node containing the application name (`app1` in this example) does not appear.	The following attributes and their associated values: `CreateCount_CountCreateCount_DescriptionCreateCount_LastSampleTimeCreateCount_NameCreateCount_StartTimeCreateCount_UnitMethodReadyCount_CurrentMethodReadyCount_DescriptionMethodReadyCount_HighWaterMarkMethodReadyCount_LastSampleTimeMethodReadyCount_LowWaterMarkMethodReadyCount_NameMethodReadyCount_StartTimeMethodReadyCount_UnitRemoveCount_CountRemoveCount_DescriptionRemoveCount_LastSampleTimeRemoveCount_NameRemoveCount_StartTimeAttributeRemoveCount_Unit`
`list -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-pool` Note: In standalone modules, the node containing the application name (`app1` in this example) will not appear.	No attributes, but a message saying: Nothing to list at `server.applications.appl.ejb-module1_jar.bean1-cache`. To get the valid names beginning with a string, use the wildcard (*) character. For example, to list all names that begin with `server`, use `list server*`.
`get -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-pool.*` Note: In standalone modules, the node containing the application name (`app1` in this example) will not appear.	List of attributes and values corresponding to EJB Pool attributes as described in Table 1-4.
`list -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-cache` Note: In standalone modules, the node containing the application name (`app1` in this example) will not appear.	No attributes, but a message saying "Use get command with the --monitor option to view this node's attributes and values."
`get -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-cache.*` Note: In standalone modules, the node containing the application name (`app1` in this example) does not appear.	List of attributes and values corresponding to EJB Cache attributes as described in Table 1-5.

TABLE 16–35 Applications - Enterprise Applications and Standalone Modules *(Continued)*

Command	Dotted Name	Output
`list -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-method.method1` Note: In standalone modules, the node containing the application name (`app1` in this example) will not appear.	No attributes, but a message saying "Use get command with the --monitor option to view this node's attributes and values."
`get -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-method.method1.*` Note: In standalone modules, the node containing the application name (`app1` in this example) will not appear.	List of attributes and values corresponding to EJB Methods attributes as described in Table 1-2.
`list -m`	`server.applications.app1.web-module1_war`	Displays the virtual server(s) assigned to the module.
`get -m`	`server.applications.app1.web-module1_war.*`	No output except a message saying there are no attributes at this node.
`list -m`	`server.applications.app1.web-module1_war.virtual_server`	Displays list of servlets registered.
`get -m`	`server.applications.app1.web-module1_war.virtual_server.*`	No output except a message saying there are no attributes at this node.
`list -m`	`server.applications.app1.web-module1_war.virtual_server.servlet1`	No attributes, but a message saying "Use get command with the --monitor option to view this node's attributes and values."
`get -m`	`server.applications.app1.web-module1_war.virtual_server.servlet1.*`	List of attributes and values corresponding to web container (Servlet) attributes as described in Table 1-7.

The following table shows the command, dotted name, and corresponding output for the HTTP Service level.

TABLE 16–36 HTTP-Service Level

Command	Dotted Name	Output
`list -m`	`server.http-service`	List of virtual servers.
`get -m`	`server.http-service.*`	No output except message saying there are no attributes at this node.
`list -m`	`server.http-service.server`	List of HTTP Listeners.
`get -m`	`server.http-service.server.*`	No output except message saying there are no attributes at this node.

TABLE 16–36 HTTP-Service Level *(Continued)*

Command	Dotted Name	Output
`list -m`	`server.http-service.server.http-listener1`	No attributes, but a message saying "Use get command with the --monitor option to view this node's attributes and values."
`get -m`	`server.http-service.server.*`	List of attributes and values corresponding to HTTP Service attributes.

The following table shows the command, dotted name, and corresponding output for the thread pools level.

TABLE 16–37 Thread-Pools Level

Command	Dotted Name	Output
`list -m`	`server.thread-pools`	List of `thread-pool` names.
`get -m`	`server.thread-pools.*`	No output except message saying there are no attributes at this node.
`list -m`	`server.thread-pools.orb\ .threadpool\.thread-pool-1`	No attributes, but a message saying "Use get command with the `--monitor` option to view this node's attributes and values."
`get -m`	`server.thread-pools..orb\ .threadpool\.thread-pool-1.*`	List of attributes and values corresponding to Thread Pool attributes.

The following table shows the command, dotted name, and corresponding output for the resources level.

TABLE 16–38 Resources Level

Command	Dotted Name	Output
`list -m`	`server.resources`	List of pool names.
`get -m`	`server.resources.*`	No output except message saying there are no attributes at this node.

TABLE 16–38 Resources Level *(Continued)*

Command	Dotted Name	Output
`list -m`	`server.resources.jdbc-connection-pool-pool.connection-pool1`	No attributes, but a message saying "Use get command with the --monitor option to view this node's attributes and values."
`get -m`	`server.resources.jdbc-connection-pool-pool.connection-pool1.*`	List of attributes and values corresponding to Connection Pool attributes.

The following table shows the command, dotted name, and corresponding output for the transaction service level.

TABLE 16–39 Transaction-Service Level

Command	Dotted Name	Output
`list -m`	`server.transaction-service`	No attributes, but a message saying "Use get command with the `--monitor` option to view this node's attributes and values."
`get -m`	`server.transaction-service.*`	List of attributes and values corresponding to Transaction Service attributes.

The following table shows the command, dotted name, and corresponding output for the ORB level.

TABLE 16–40 ORB Level

Command	Dotted Name	Output
`list -m`	`server.orb`	`server-orb.connection-managers`
`get -m`	`server.orb.*`	No output except message saying there are no attributes at this node.
`list -m`	`server.orb.connection-managers`	Name(s) of ORB connection managers.
`get -m`	`server.orb.connection-managers.*`	No output except message saying there are no attributes at this node.
`list -m`	`server.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnections`	No attributes, but a message saying "Use get command with the --monitor option to view this node's attributes and values."

TABLE 16–40 ORB Level *(Continued)*

Command	Dotted Name	Output
`get -m`	`server.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnections.*`	List of attributes and values corresponding to ORB Connection Manager attributes.

The following table shows the command, dotted name, and corresponding output for the JVM level.

TABLE 16–41 JVM Level

Command	Dotted Name	Output
`list -m`	`server.jvm`	No attributes, but a message similar to the following displays: Use get command with the `--monitor` option to view this node's attributes and values.
`get -m`	`server.jvm.*`	List of attributes and values corresponding to JVM attributes.

Using JConsole

This section contains the following topics:

Administration (management and monitoring) of the Application Server is based on JMX. This means that the managed components are represented as MBeans. With Java 2 Standard Edition (J2SE) 5.0, you can monitor the JVM and view the JVM MBeans to understand what is going on there. To expose this instrumentation, Application Server provides a configuration of Standard JMX Connector Server called `System JMX Connector Server`. When the Application Server starts up, an instance of this JMX Connector Server starts up, exposing the instrumentation to trusted clients.

The Java Monitoring and Management Console (JConsole) is a popular JMX Connector that can manage a JMX backend. JConsole (`http://java.sun.com/j2se/1.5.0/docs/tooldocs/share/jconsole.html`) is available as part of the standard JDK distribution beginning with J2SE 5.0. For more information on JConsole, see `http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html`

When you configure JConsole with Application Server, the Application Server becomes the JMX Connector's server-end and JConsole becomes the JMX Connector's preferred client-end.

Securing JConsole to Application Server Connection

There are subtle differences in how to connect to Application Server, or any JMX Connector Server end based on the transport layer security of the connection. If the server end is secure (guarantees transport layer security), there is a little more configuration to be performed on the client end.

- By default, Platform Edition of Application Server has System JMX Connector Server end as insecure.
- By default, Enterprise Edition of Application Server has System JMX Connector Server end as secure.
- The protocol used for communication is RMI/JRMP. If security is enabled for the JMX Connector, the protocol used is RMI/JRMP over SSL.

Note – RMI over SSL does not provide additional checks to ensure that the client is talking to the intended server. Thus, there is always a possibility, while using JConsole, that you are sending the user name and password to a malicious host. It is completely up to the administrator to make sure that security is not compromised.

When you install a Platform Edition domain on a machine such as `appserver.sun.com`, you will see the following in the DAS's (Domain Administration Server, the admin server or simply the domain) `domain.xml`:

```
<!- – The JSR 160 "system-jmx-connector" – –>
<jmx-connector accept-all="false" address="0.0.0.0"
auth-realm-name="admin-realm" enabled="true" name="system" port="8686"
protocol="rmi_jrmp" security-enabled="false"/>
<!- – The JSR 160 "system-jmx-connector" – –>
```

The `security-enabled` flag for the JMX Connector is *false*. If you are running the Enterprise Edition, or if you have turned on security for the JMX Connector in the Platform Edition, this flag is set to *true*.

```
<!- – The JSR 160 "system-jmx-connector" – –>
<jmx-connector accept-all="false" address="0.0.0.0"
auth-realm-name="admin-realm" enabled="true" name="system" port="8686"
protocol="rmi_jrmp" security-enabled="true"/>
...
</jmx-connector>
<!- – The JSR 160 "system-jmx-connector" – –>
```

Prerequisites for Connecting JConsole to Application Server

The JConsole setup has two parts: a server end and a client end. The Application Server domain is installed on a machine called `appserver.sun.com`, which is a powerful Solaris server. This is the server end.

The client end also has an installation of Application Server. Let us assume that the client end is a Windows machine with Java SE 5.0 and Application Server installed.

Note – The Application Server installation is needed on the client end only when your Application Server domain has security enabled on the remote machine (the default for Enterprise Edition). If you just want to administer an Application Server Platform Edition domain on the Solaris machine above, you do not need the Application Server installation on this client machine.

If the server and client ends are on the same machine, you can use `localhost` to specify the host name.

Connecting JConsole to the Application Server

This section describes connecting JConsole to Application Server without security enabled on the JMX Connector. By default, security is not enabled on Application Server Platform Edition.

1. Start the domain on `appserver.sun.com`.
2. Start JConsole by running *JDK_HOME*`/bin/jconsole`
3. In the Connect to Agent tab of JConsole, enter user name, password, host name and port (8686, by default).

   The user name refers to the admin user name and password refers to the admin password of the domain.
4. Click Connect.

   In the JConsole window you will see all your MBeans, VM information etc., in various tabs.

Connecting JConsole Securely to Application Server

This section describes how to connect JConsole to Application Server with security enabled on the JMX Connector. By default, security is enabled on Application Server Enterprise Edition. Use this procedure if you have security enabled on the Platform Edition's JMX Connector.

1. Install Application Server on the client machine (where JConsole is installed).

The only reason you need this is to let JConsole know where the server certificate of the Domain Administration Server that you trust is located. To obtain that certificate, invoke at least one *remote* `asadmin` command and to do that, you need the local installation of Application Server.

2. Start the Application Server Enterprise Edition on `appserver.sun.com`.

   Since this is an Enterprise Edition domain, the system JMX Connector server is secure.

3. From the local Application Server installation, run *install-dir*`/bin/asadmin list --user admin --secure=true --host appserver.sun.com --port 4849` (where 4849 is the server's admin port).

   Though `asadmin list` command is chosen for this example, you can run any remote `asadmin` command. You will now be prompted to accept the certificate sent by the DAS of `appserver.sun.com`.

4. Press `y` to accept the certificate sent by the Domain Administration Server on `appserver.sun.com`.

   The server's certificate is stored in a file called `.asadmintruststore` in your home directory on the client machine.

   Note – This step is not required if your server machine and client machine is the same. That is, if you are running JConsole also on `appserver.sun.com`.

   ---

5. Let JConsole know the DAS's trust store location by using the following JConsole command:

   JDK-dir`/bin/jconsole.exe -J-Djavax.net.ssl.trustStore="C:/Documents and Settings/`*user*`/.asadmintruststore"`

   This certificate is now automatically trusted by JConsole.

6. Start JConsole by running *JDK_HOME*`/bin/jconsole`

7. In the Connect to Agent tab of JConsole, enter user name, password, host name and port (8686, by default).

   The user name refers to the admin user name and password refers to the admin password of the domain.

8. Click Connect.

   In the JConsole window you will see all your MBeans, VM information etc., in various tabs.

◆ ◆ ◆ CHAPTER 17

Java Virtual Machine and Advanced Settings

The Java virtual machine (JVM) is an interpretive computing engine responsible for running the byte codes in a compiled Java program. The JVM translates the Java byte codes into the native instructions of the host machine. The application server, being a Java process, requires a JVM in order to run, and to support the Java applications running on it. JVM settings are part of an application server configuration.

This chapter explains how to configure the Java Virtual Machine (JVM) and other advanced settings. It contains the following sections:

Tuning the JVM Settings

As part of configuring the application server, you define settings that enhance the use of the Java Virtual machine. To change the JVM configuration using the Admin Console, select Application Server > JVM Settings tab and define the general JVM settings as follows:

- Java Home: Enter the name of the installation directory of the Java software. The Application Server relies on the Java SE software.

  Note – If you enter a nonexistent directory name or the installation directory name of an unsupported version of the Java EE software, then the Application Server will not start.

- Javac Options: Enter the command-line options for the Java programming language compiler. The Application Server runs the compiler when EJB components are deployed.
- Debug: To set up debugging with the JPDA (Java Platform Debugger Architecture), select this Enabled checkbox.

  JPDA is used by application developers.

- Debug Options: Specify the JPDA options passed to the JVM when the debugging is enabled.
- RMI Compile Options: Enter the command-line options for the `rmic` compiler. The Application Server runs the `rmic` compiler when EJB components are deployed.
- Bytecode Preprocessor: Enter a comma separated list of class names. Each class must implement the `com.sun.appserv.BytecodePreprocessor` interface. The classes are called in the order specified.

  Tools such as profilers may require entries in the Bytecode Preprocessor field. Profilers generate information used to analyze server performance.

Configuring Advanced Settings

To set the advanced application configuration using the Admin Console, select Application Server > Advanced tab > Application Configuration tab and set the application configuration as follows:

- Reload: Select this checkbox to enable dynamic reloading of applications.

  If dynamic reloading is enabled (it is by default), you do not have to redeploy an application or module when you change its code or deployment descriptors. All you have to do is copy the changed JSP or class files into the deployment directory for the application or module. The server checks for changes periodically and redeploys the application, automatically and dynamically, with the changes. This is useful in a development environment, because it allows code changes to be tested quickly. In a production environment, however, dynamic reloading might degrade performance. In addition, whenever a reload is done, the sessions at that transit time become invalid. The client must restart the session.
- Reload Poll Interval: Define the interval at which applications and modules are checked for code changes and dynamically reloaded. The default is 2.
- Admin Session Timeout: Specify the number of minutes of inactivity after which the admin session times out.

In addition, define the deploy settings as follows:

- Auto Deploy: Select this checkbox to enable automatic deployment of applications.

  Autodeployment involves copying an application or module file (JAR, WAR, RAR, or EAR) into a special directory, where it is automatically deployed by the Application Server.
- Auto Deploy Poll Interval: Define the interval at which applications and modules are checked for code changes and dynamically reloaded. The default is 2.
- Verifier: Check the Verifier Enabled box to verify your deployment descriptor files. This is optional.
- Precompile: Check the Precompile Enabled box to precompile any JSP files.

◆ ◆ ◆ CHAPTER 18

Dotted Name Attributes for domain.xml

This appendix describes the dotted name attributes that can be used to address the MBean and its attributes. Every element in the `domain.xml` file has a corresponding MBean. Because the syntax for using these names involves separating names between periods, these names are called *dotted names*.

A * (asterisk) can be used anywhere in the dotted name and it acts like the wildcard character in regular expressions. The benefit of using a wildcard character is that it can collapse all the parts of the dotted name. For example, a long dotted name like `this.is.really.long.hierarchy` can be abbreviated to `th*.hierarchy`. However, the `.` always delimits parts of the name. An * will get you the entire list of dotted-names.

This appendix contains the following topics:

Top Level Elements

The following conditions must be adhered to for all top level elements in the `domain.xml` file:

- Each server, configuration, cluster, or node agent must have a unique name.
- Servers, configurations, clusters, or node agents cannot be named `domain`.
- Server instances cannot be named `agent`.

The following table identifies the top level elements and the corresponding dotted name prefix.

Element Name	Dotted Name Prefix
applications	`domain.applications`

Element Name	Dotted Name Prefix
resources	`domain.resources`
configurations	`domain.configs`
servers	`domain.servers` Every server contained in this element is accessible as *server-name*. Where *server-name* is the value of the name attribute for the server subelement.
clusters	`domain.clusters` Every cluster contained in this element is accessible as *cluster-name*. Where *cluster-name* is the value of the name attribute for the cluster subelement.
node-agents	`domain.node-agents`
lb-configs	`domain.lb-configs`
system-property	`domain.system-property`

Two levels of aliasing are available:

1. The first level of alias allows access to attributes of server instances or clusters without going through the `domain.servers` or `domain.clusters` prefix. So, for example, a dotted name of the form `server1` maps to the dotted name `domain.servers.server1` (where `server1` is a server instance).
2. The second level of alias is used to refer to configurations, applications, and resources of a cluster or a standalone server instance (target).

The following table identifies dotted names beginning with the server name, or cluster name, that are aliased to top level names under the domain:

Dotted Name	Aliased to	Comments
target`.applications.*`	`domain.applications.*`	The alias resolves to applications referenced by the *target* only.
target`.resources.*`	`domain.resources.*`	The alias resolves to all `jdbc-connection-pool`, `connector-connection-pool`, `resource-adapter-config`, and all other resources referenced by the *target*.

The following table identifies dotted names beginning with the server name, or cluster name, that are aliased to top level names within the configuration referenced by the server or cluster.

Dotted Name	Aliased to
target.http-service	*config-name*.http-service
target.iiop-service	*config-name*.iiop-service
target.admin-service	*config-name*.admin-service
target.web-container	*config-name*.web-container
target.ejb-container	*config-name*.ejb-container
target.mdb-container	*config-name*.mdb-container
target.jms-service	*config-name*.jms-service
target.log-service	*config-name*.log-service
target.security-service	*config-name*.security-service
target.transaction-service	*config-name*.transaction-service
target.monitoring-service	*config-name*.monitoring-service
target.java-config	*config-name*.java-config
target.availability-service	*config-name*.availability-service
target.thread-pools	*config-name*.thread-pools

Elements Not Aliased

A clustered instance should not be aliased. To get a system property for a clustered instance, the dotted name attribute you should use is as follows:
`domain.servers.`*clustered-instance-name*`.`*system-property* , not
clustered-instance-name.*system-property*.

asadmin Commands

The `asadmin` `get`, `set`, and `list` commands work in tandem to provide a navigation mechanism for the Application Server's abstract hierarchy. There are two hierarchies: configuration and monitoring and these commands operate on both. The `list` command provides the fully qualified dotted names of the management components that have read-only or modifiable attributes.

The configuration hierarchy provides attributes that can be modified; whereas the attributes of management components from monitoring hierarchy are purely read only. The configuration hierarchy is loosely based on the domain's schema document. Use the `list` command to reach a particular management component in the desired hierarchy. Then, invoke the `get` and `set`

commands to get the names and values or set the values of the attributes of the management component. Use the wildcard (*) option to fetch all matches in a given fully qualified dotted name. For examples of using the `get`, `set`, and `list` commands, see the following man pages:

`get(1)`
`set(1)`
`list(1)`

◆ ◆ ◆ CHAPTER 19

The asadmin Utility

The Application Server includes a command-line administration utility known as `asadmin`. The`asadmin` utility is used to start and stop the Application Server, as well as manage users, resources, and applications.

This chapter contains the following sections:

The asadmin Command Usage

Use the asadmin utility to perform any administrative tasks for the Application Server. You can use this asadmin utility in place of using the Administration Console.

The asadmin utility invokes commands that identify the operation or task you wish to perform. These commands are case-sensitive. Short option arguments have a single dash (-); while long option arguments have two dashes (--). Options control how the utility performs a command. Options are also case-sensitive. Most options require argument values except boolean options which toggle to switch a feature ON or OFF. Operands appear after the argument values, and are set off by a space, a tab, or double dashes (--). The asadmin utility treats anything that comes after the options and their values as an operand.

EXAMPLE 19–1 Syntax Example

`asadmin` *command* `[-short_option] [`*short_option_argument*`]* [--long_option [`*long_option_argument*`]* [`*operand*`]*`

```
asadmin create-profiler -u admin --passwordfile password.txt myprofiler
```

To access the man pages for the Application Server asadmin utility commands on the Solaris platform, add $AS_INSTALL/man to your MANPATH environment variable.

You can obtain overall usage information for any of the asadmin utility commands by invoking the --help option. If you specify a command, the usage information for that command is displayed. Using the --help option without a command displays a listing of all the available commands.

EXAMPLE 19–2 help Command Example

`asadmin --help` displays general help

`asadmin` *command* `--help` displays help for the specified command.

This section contains the following topics:

Multi and Interactive Modes

The asadmin utility can be used in command shell invocation or multi command mode (known as the multimode command). In command shell invocation you invoke the asadmin utility from your command shell. asadmin executes the command, then exits. In multiple command mode,

you invoke asadmin once, it then accepts multiple commands until you exit asadmin and return to the normal command shell invocation. Environment variables set while in multiple command mode are used for all subsequent commands until you exit multimode. You may provide commands by passing a previously prepared list of commands from a file or standard input (pipe). Additionally, you can invoke multimode from within a multimode session; once you exit the second multimode environment, you return to your original multimode environment.

You can also run the asadmin utility in interactive or non-interactive mode. By default, the interactive mode option is enabled. It prompts you for the required arguments. You can use the interactive mode option in command shell invocation under all circumstances. You can use the interactive mode option in multimode when you run one command at a time from the command prompt; and when you run in multimode from a file. Commands in multimode, when piped from an input stream, and commands invoked from another program, cannot run in the interactive mode.

Local Commands

Local commands can be executed without the presence of an administration server. However, it is required that the user be logged into the machine hosting the domain in order to execute the command and have access (permissions) for the installation and domain directories.

For commands that can be executed locally or remotely, if any one of the--host, --port, --user, and --passwordfile options are set, either in the environment or in the command line, the command will run in remote mode. Also, if none of the local options are set, either on the command line or in the environment, the command is executed locally by default. For commands that can be executed locally or remotely, if any one of the--host, --port, --user, and --passwordfile options are set, either in the environment or in the command line, the command will run in remote mode.

Remote Commands

Remote commands are always executed by connecting to an administration server and executing the command there. A running administration server is required. All the remote commands require the following common options:

TABLE 19–1 Remote Commands Required Options

Short Option	Option	Definition
-H	--host	The machine name where the domain administration server is running. The default value is localhost.

TABLE 19–1 Remote Commands Required Options *(Continued)*

Short Option	Option	Definition
-p	--port	The HTTP/S port for administration. This is the port to which you should point your browser in order to manage the domain. For example, http://localhost:4848. The default port number for Platform Edition is 4848.
-u	--user	The authorized domain administration server administrative username. If you have authenticated to a domain using the asadmin login command, then you need not specify the --user option on subsequent operations to this particular domain.
	--passwordfile	The --passwordfile option specifies the name of a file containing the password entries in a specific format. The entry for the password must have the AS_ADMIN_ prefix followed by the password name in uppercase letters. For example, to specify the domain administration server password, use an entry with the following format: AS_ADMIN_PASSWORD=*password*, where *password* is the actual administrator password. Other passwords that can be specified include AS_ADMIN_PASSWORD, AS_ADMIN_USERPASSWORD, and AS_ADMIN_ALIASPASSWORD, AS_ADMIN_MAPPEDPASSWORD. All remote commands must specify the admin password to authenticate to the domain administration server, either through --passwordfile or asadmin login, or interactively on the command prompt. The asadmin login command can be used only to specify the admin password. For other passwords, that must be specified for remote commands, use the --passwordfile or enter them at the command prompt. If you have authenticated to a domain using the asadmin login command, then you need not specify the admin password through the --passwordfile option on subsequent operations to this particular domain. However, this is applicable only to AS_ADMIN_PASSWORD option. You will still need to provide the other passwords, for example, AS_ADMIN_USERPASSWORD, as and when required by individual commands, such as update-file-user. For security reasons, passwords specified as an environment variable will not be read by asadmin.
-s	--secure	If set to true, uses SSL/TLS to communicate with the domain administration server.
-I	--interactive	If set to true (default), only the required password and user options are prompted.
-t	--terse	Indicates that any output data must be very concise, typically avoiding human-friendly sentences and favoring well-formatted data for consumption by a script. Default is false.

TABLE 19–1 Remote Commands Required Options *(Continued)*

Short Option	Option	Definition
`-e`	`--echo`	Setting to true will echo the command line statement on the standard output. Default is false.
`-h`	`--help`	Displays the help text for the command.

The Password File

For security purposes, you can set the password for a command from a file instead of entering the password at the command line. The `--passwordfile` option takes the file containing the passwords. The valid contents for the file are:

EXAMPLE 19–3 Passwordfile contents

```
AS_ADMIN_PASSWORD=value
AS_ADMIN_ADMINPASSWORD=value
AS_ADMIN_USERPASSWORD=value
AS_ADMIN_MASTERPASSWORD=value
```

Multimode Command

Use the `multimode` command to process the `asadmin` commands. The command-line interface will prompt you for a command, execute that command, display the results of the command, and then prompt you for the next command. Additionally, all the `asadmin` option names set in this mode are used for all the subsequent commands. You can set your environment and run commands until you exit `multimode` by typing “exit” or “quit.” You can also provide commands by passing a previously prepared list of commands from a file or standard input (pipe). You can invoke `multimode` from within a *multimode* session; once you exit the second *multimode* environment, you return to your original *multimode* environment.

To invoke multimode, enter `asadmin multimode`.

The List, Get and Set Commands

The `asadmin list`, `get` and `set` commands work in tandem to provide a navigation mechanism for the Application Server's dotted naming hierarchy. There are two hierarchies: **configuration** and**monitoring** and these commands operate on both. The `list` command provides the fully qualified dotted names of the management components that have read-only or modifiable attributes.

The **configuration** hierarchy provides attributes that are modifiable; whereas the attributes of management components from the**monitoring** hierarchy are purely read-only. The**configuration** hierarchy is loosely based on the domain's schema document; whereas the **monitoring** hierarchy is a little different.

Use the `list` command to reach a particular management component in the desired hierarchy. Then, invoke the `get`and `set` commands to get the names and values or set the values of the attributes of the management component at hand. Use the wildcard (*) option to fetch all matches in a given fully qualified dotted name.

An Application Server dotted name uses the "`.`" (period) as a delimiter to separate the parts of a complete name. This is similar to how the "`/`" character is used to delimit the levels in the absolute path name of a file in the UNIX file system. The following rules apply while forming the dotted names accepted by the `get`, `set`, and `list` commands. Note that a specific command has some additional semantics applied.

- A `.` (period) always separates two sequential parts of the name.
- A part of the name usually identifies an application server subsystem and/or its specific instance. For example: web-container, log-service, thread-pool-1, etc.
- If any part of the name itself contains a `.` (period), then it must be escaped with a leading \(backslash) so that the "`.`" does not act like a delimiter.
- An * (asterisk) can be used anywhere in the dotted name and it acts like the wildcard character in regular expressions. Additionally, an * can collapse all the parts of the dotted name. Long dotted name like "`<classname>this.is.really.long.hierarchy </classname>`" can be abbreviated to "`<classname>th*.hierarchy</classname>`." But note that the `.` always delimits the parts of the name.
- On Solaris, quotes are needed when executing commands with * as the option value or operand.
- The top level switch for any dotted name is `--monitor` or `-m` that is separately specified on a given command line. The presence or lack of this switch implies the selection of one of the two hierarchies for application server management: monitoring and configuration.
- If you happen to know the exact complete dotted name without any wildcard character, then `list` and `get`/`set` have a little difference in their semantics:
  - The `list` command treats this complete dotted name as the complete name of a parent node in the hierarchy. Upon providing this name to the `list` command, it simply returns the names of the immediate children at that level. For example,`list server.applications.web-module` will list all the web modules deployed to the domain or the default server.
  - The `get` and `set` commands treat this complete dotted name as the fully qualified name of the attribute of a node (whose dotted name itself is the name that you get when you remove the last part of this dotted name) and it gets/sets the value of that attribute. This is true if such an attribute exists. You will never start with this case because in order to find out the names of attributes of a particular node in the hierarchy, you must use the

wildcard character *. For example, `server.applications.web-module.JSPWiki.context-root*` returns the context-root of the web-application deployed to the domain or default server.

The `list` command is the progenitor of navigational capabilities of these three commands. If you want to `set` or `get` attributes of a particular application server subsystem, you must know its dotted name. The `list` command is the one which can guide you to find the dotted name of that subsystem. For example, to find out the modified date (attribute) of a particular file in a large file system that starts with `/`. First you must find out the location of that file in the file system, and then look at its attributes. Therefore, two of the first commands to understand the hierarchies in the Application Server are: `* list "*"` and `<command>* list * --monitor`. Consult the `get`, `set` or `list` commands man pages to identify the sorted output of these commands.

Server Lifecycle Commands

The server lifecycle commands are commands that create, delete, or start, stop a domain, service (DAS), or an instance.

TABLE 19–2 Server Lifecycle Commands

Command	Definition
`create-domain`	Creates the configuration of a domain. A domain is an administrative namespace. Every domain has a configuration, which is stored in a set of files. Any number of domains, each of which has a distinct administrative identity, can be created in a given installation of the Application Server. A domain can exist independent of other domains. Any user who has access to the `asadmin` utility on a given system can create a domain and store its configuration in a folder of choice. By default, the domain configuration is created in the *install_dir*`/domains` directory. You can override this location to store the configuration elsewhere.
`delete-domain`	Deletes the named domain. The domain must already exist and must be stopped.
`start-domain`	Starts a domain. If the domain directory is not specified, the domain in the default *install_dir*`/domains` directory is started. If there are two or more domains, the *domain_name* operand must be specified.
`stop-domain`	Stops the Domain Administration Server of the specified domain.
`restore-domain`	Restores files under the domain from a backup directory.
`list-domains`	Lists the domain. If the domain directory is not specified, the domain in the default *install_dir*`/domains` directory is listed. If there is more than one domain, the *domain_name* operand must be specified.

TABLE 19–2 Server Lifecycle Commands *(Continued)*

Command	Definition
`backup-domain`	Backs up files under the named domain.
`list-backups`	Displays the status information about all backups in the backup repository.
`shutdown`	Gracefully brings down the administration server and all the running instances. You must manually start the administration server to bring it up again.

List and Status Commands

The list and status commands display the status of a deployed component.

TABLE 19–3 List and Status Commands

Command	Definition
`show-component-status`	Gets the status of the deployed component. The status is a string representation returned by the server. The possible status strings include status of *app-name* is `enabled` or status of *app-name* is `disabled`.
`list-components`	Lists all deployed Java EE 5 components. If the `--type` option is not specified, all components are listed.
`list-sub-components`	Lists EJBs or Servlets in a deployed module or in a module of the deployed application. If a module is not identified, all modules are listed.

Deployment Commands

The deployment commands deploy an application or get the client stubs.

TABLE 19–4 Deployment Commands

Command	Definition
`deploy`	Deploys an enterprise application, web application, EJB module, connector module, or application client module. If the component is already deployed or already exists, it is forcefully redeployed if the `--force` option is set to `true`.
`deploydir`	Deploys an application directly from a development directory. The appropriate directory hierarchy and deployment descriptors conforming to the Java EE specification must exist in the deployment directory.

TABLE 19–4 Deployment Commands *(Continued)*

Command	Definition
`get-client-stubs`	Gets the client stubs JAR file for an `AppClient` standalone module or an application containing the `AppClient` module, from the server machine to the local directory. The application or module should be deployed before executing this command. it is also possible to get the client stubs as part of the `deploy` command using `--retrieve` option.
`undeploy`	Removes the specified deployed component.

Message Queue Administration Commands

The Message Queue administration commands allow you to manage the JMS destinations.

TABLE 19–5 Message Queue Commands

Command	Definition
`create-jmsdest`	Creates a JMS physical destination. Along with the physical destination, you use the `create-jms-resource` command to create a JMS destination resource that has a `Name` property that specifies the physical destination.
`delete-jmsdest`	Removes the specified JMS destination.
`flush-jmsdest`	Purges the messages from a physical destination in the specified target's JMS Service configuration.
`list-jmsdest`	Lists the JMS physical destinations.
`jms-ping`	Checks if the JMS service (also known as the JMS provider) is up and running. When you start the Application Server, the JMS service starts by default. Additionally, it pings only the default JMS host within the JMS service. It displays an error message when it is unable to ping a built-in JMS service.

Resource Management Commands

The resource commands allow you to manage the various resources used in your application.

TABLE 19–6 Resource Management Commands

Command	Definition
`create-jdbc-connection-pool`	Registers a new JDBC connection pool with the specified JDBC connection pool name.

TABLE 19–6 Resource Management Commands *(Continued)*

Command	Definition
`delete-jdbc-connection-pool`	Deletes a JDBC connection pool. The operand identifies the JDBC connection pool to be deleted.
`list-jdbc-connection-pools`	Gets the JDBC connection pools that have been created.
`create-jdbc-resource`	Creates a new JDBC resource.
`delete-jdbc-resource`	Removes a JDBC resource with the specified JNDI name.
`list-jdbc-resources`	Displays a list of JDBC resources that have been created.
`create-jms-resource`	Creates a Java Message Service (JMS) connection factory resource or a JMS destination resource.
`delete-jms-resource`	Removes the specified JMS resource.
`list-jms-resources`	Lists the existing JMS resources (destination and connection factory resources).
`create-jndi-resource`	Registers a JNDI resource.
`delete-jndi-resource`	Removes the JNDI resource with the specified JNDI name.
`list-jndi-resources`	Identifies all the existing JNDI resources.
`list-jndi-entries`	Browses and queries the JNDI tree.
`create-javamail-resource`	Creates a JavaMail session resource.
`delete-javamail-resource`	Removes the specified JavaMail session resource.
`list-javamail-resources`	Lists the existing JavaMail session resources.
`create-persistence-resource`	Registers a persistence resource.
`delete-persistence-resource`	Removes a persistence resource. When you delete a persistence resource, the command also removes the JDBC resource if it was created using the `create-persistence-resource` command.
`list-persistence-resources`	Displays all the persistence resources.
`create-custom-resource`	Creates a custom resource. A custom resource specifies a custom server-wide resource object factory that implements the `javax.naming.spi.ObjectFactory` interface.
`delete-custom-resource`	Removes a custom resource.
`list-custom-resources`	Lists the custom resources.
`create-connector-connection-pool`	Adds a new connector connection pool with the specified connection pool name.

TABLE 19–6 Resource Management Commands *(Continued)*

Command	Definition
`delete-connector-connection-pool`	Removes the connector connection pool specified using the operand `connector_connection_pool_name`.
`list-connector-connection-pools`	Lists the connector connection pools that have been created.
`create-connector-resource`	Registers the connector resource with the specified JNDI name.
`delete-connector-resource`	Removes the connector resource with the specified JNDI name.
`list-connector-resources`	Gets all the connector resources.
`create-admin-object`	Adds the administered object that has the specified JNDI name.
`delete-admin-object`	Removes the administered object with the specified JNDI name.
`list-admin-objects`	Lists all the administered objects.
`create-resource-adapter-config`	Creates configuration information for the connector module.
`delete-resource-adapter-config`	Deletes the configuration information created in `domain.xml` for the connector module.
`list-resource-adapter-configs`	Lists the configuration information in the `domain.xml` for the connector module
`add-resources`	Creates the resources named in the specified XML file. The *xml_file_path* is the path to the XML file containing the resources to be created. The `DOCTYPE` should be specified as *install_dir*`/lib/dtds/sun-resources_1_2.dtd` in the `resources.xml` file.
`ping-connection-pool`	Tests if a connection pool is usable for both JDBC connection pools and connector connection pools. For example, if you create a new JDBC connection pool for an application that is expected to be deployed later, the JDBC pool is tested with this command before deploying the application. Before pinging a connection pool, you must create the connection pool with authentication and ensure that the Application Server or database is started.

Application Server Configuration Commands

The configuration commands allow you to configure the operation of the Application Server. This section contains the following topics:

General Configuration Commands

These commands allow you to manage the configuration of the Application Server components.

TABLE 19–7 General Configuration Commands

Command	Definition
`enable`	Enables the specified component. If the component is already enabled, then it is re-enabled. The component must have been deployed in order to be enabled. If it has not been deployed, then an error message is returned.
`disable`	Immediately disables the named component. The component must have been deployed. If the component has not been deployed, an error message is returned.
`export`	Marks a variable name for automatic export to the environment of subsequent commands. All subsequent commands use the variable name value as specified unless you `unset` them or exit `multimode`.
`get`	Gets the names and values of attributes.
`set`	Sets the values of one or more configurable attribute.
`list`	Lists the configurable element. On Solaris, quotes are needed when executing commands with * as the option value or operand.
`unset`	Removes one or more variables you set for the multimode environment. The variables and their associated values will no longer exist in the environment.

HTTP, IIOP and SSL Listener Commands

The HTTP and IIOP listener commands help you manage listeners. These commands are supported in remote mode only.

TABLE 19–8 IIOP Listener Commands

Command	Definition
`create-http-listener`	Adds a new HTTP listener.
`delete-http-listener`	Removes the specified HTTP listener.
`list-http-listeners`	Lists the existing HTTP listener.

TABLE 19–8 IIOP Listener Commands *(Continued)*

Command	Definition
`create-iiop-listener`	Creates an IIOP listener.
`delete-iiop-listener`	Removes the specified IIOP listener.
`list-iiop-listeners`	Lists the existing IIOP listeners.
`create-ssl`	Creates and configures the SSL element in the selected HTTP listener, IIOP listener, or IIOP service to enable secure communication on that listener/service.
`delete-ssl`	Deletes the SSL element in the selected HTTP listener, IIOP listener, or IIOP service.

Lifecycle and Audit Module Commands

The lifecycle and audit module commands help you control lifecycle modules and optional plug-in modules which implement audit capabilities. The commands are supported in remote mode only.

TABLE 19–9 Lifecycle Module Commands

Command	Definition
`create-lifecycle-module`	Creates a lifecycle module. The lifecycle modules provide a means of running short or long duration Java-based tasks within the Application Server environment.
`delete-lifecycle-module`	Removes the specified lifecycle module.
`list-lifecycle-modules`	Lists the existing lifecycle module.
`create-audit-module`	Adds the named audit module for the plug-in module that implements the audit capabilities.
`delete-audit-module`	Removes the named audit module.
`list-audit-modules`	Lists all the audit modules.

Profiler and JVM Options Commands

The Profiler and JVM options commands allow you to administrate profilers and control these elements. These commands are supported in remote mode only.

TABLE 19–10 Profiler and JVM Options Commands

Command	Definition
`create-profiler`	Creates the profiler element. A server instance is tied to a particular profiler, by the profiler element in the Java configuration. Changing a profiler requires you to restart the server.
`delete-profiler`	Deletes the profiler element you specify. A server instance is tied to a particular profiler by the profiler element in the Java configuration. Changing a profiler requires you to restart the server.
`create-jvm-option`	Creates JVM options in the Java configuration or profiler elements of the `domain.xml` file. If JVM options are created for a profiler, they are used to record the settings needed to get a particular profiler going. You must restart the server for newly created JVM options to take effect.
`delete-jvm-option`	Removes JVM options from the Java configuration or profiler elements of the `domain.xml` file.

Virtual Server Commands

The Virtual Server commands allow you to control these elements. These commands are supported in remote mode only.

TABLE 19–11 Virtual Server Commands

Command	Definition
`create-virtual-server`	Creates the named virtual server. Virtualization in the Application Server allows multiple URL domains to be served by a single HTTP server process that is listening on multiple host addresses. If the application is available at two virtual servers, they still share the same physical resource pools.
`delete-virtual-server`	Removes the virtual server with the specified virtual server ID.
`list-virtual-server`	Lists the existing virtual servers.

Threadpool Commands

The threadpool commands allow you to control these elements. These commands are supported in remote mode only.

TABLE 19–12 Threadpool Commands

Command	Definition
`create-threadpool`	Creates a threadpool with the specified name. You can specify maximum and minimum number of threads in the pool, the number of work queues, and the idle timeout of a thread. The created thread pool can be used for servicing IIOP requests and for resource adapters to service work management requests. A created thread pool can be used in multiple resource adapters.
`delete-threadpool`	Removes the threadpool with the named ID.
`list-threadpools`	Lists all the thread pools.

Transaction and Timer Commands

The transaction and timer commands allow you to control the transaction and timer subsystems; allowing you to suspend any inflight transactions. These commands are supported in remote mode only.

TABLE 19–13 Transaction Commands

Command	Definition
`freeze-transaction`	Freezes the transaction subsystem during which time all the inflight transactions are suspended. Invoke this command before rolling back any inflight transactions. Invoking this command on an already frozen transaction subsystem has no effect.
`unfreeze-transaction`	Resumes all the suspended inflight transactions. Invoke this command on an already frozen transaction.
`recover-transactions`	Manually recovers pending transactions.
`rollback-transaction`	Rolls back the named transaction.
`list-timers`	Lists the timers owned by a specific server instance

User Management Commands

These user commands are to administer the users support by the file realm authentication. These commands are supported in remote mode only.

TABLE 19–14 User Management Commands

Command	Definition
`create-file-user`	Creates an entry in the keyfile with the specified username, password, and groups. Multiple groups can be created by separating them with a colon (`:`).
`delete-file-user`	Deletes the entry in the keyfile with the specified username.
`update-file-user`	Updates an existing entry in the keyfile using the specified `user_name`, `user_password` and groups. Multiple groups can be entered by separating them, with a colon (`:`).
`list-file-users`	Creates a list of file users supported by file realm authentication.
`list-file-groups`	Lists the users and groups supported by the file realm authentication. This command lists available groups in the file user.

Monitoring Data Commands

The monitoring data commands allow you to monitor the server. These commands are supported in remote mode only.

TABLE 19–15 Monitoring Data Commands

Command	Definition
`start-callflow-monitoring`	Collects and correlates data from Web container, EJB container and JDBC to provide a complete call flow/path of a request. Data is collected only if `callflow-monitoring` is ON.
`stop-callflow-monitoring`	Disables collection of call flow information of a request.

Database Commands

The database commands allow you to start and stop the Java DB database (based on Apache Derby). These commands are supported in local mode only.

TABLE 19–16 Database Commands

Command	Definition
start-database	Starts the Java DB server that is available with the Application Server. Use this command only for working with applications deployed to the Application Server.

TABLE 19–16 Database Commands *(Continued)*

Command	Definition
stop-database	Stops a process of the Java DB server. Java DB server is available with the Application Server.

Diagnostic and Logging Commands

The diagnostic and logging commands help you troubleshoot problems with the Application Server. These commands are supported in remote mode only.

TABLE 19–17 Diagnostic and Logging Commands

Command	Definition
`generate-diagnostic-report`	Generates an HTML report that contains pointers or navigational links to an application server installation details such as configuration details, logging details, or process specific information for an application server instance.
`generate-jvm-report`	Displays the threads (dump of stack trace), classes and memory for a given target instance, including the Domain Administration Service. This command works only with the application server instance processes. This command replaces the traditional techniques like sending `ctrl+break` or `kill -3` signals to application server processes. The command will not work if the target server instance is not running.
`display-error-statistics`	Displays a summary list of severities and warnings in `server.log` since the last server restart.
`display-error-distribution`	Displays distribution of errors from instance `server.log` at module level.™
`display-log-records`	Displays all the error messages for a given module at a given timestamp.

Web Service Commands

The web service commands allow you to monitor a deployed web service and manage transformation rules.

TABLE 19–18 Web Service Commands

Command	Definition
`configure-webservice-management`	Configure the monitoring or the `maxhistorysize` attributes of a deployed web service endpoint.

TABLE 19–18 Web Service Commands *(Continued)*

Command	Definition
`create-transformation-rule`	Creates an XSLT transformation rule that can be applied to a web service operation. The rule can be applied to a request, a response, or both.
`delete-transformation-rule`	Deletes an XSLT transformation rule of a given web service.
`list-transformation-rules`	Lists all the transformation rules of a given web service in the order they are applied.
`publish-to-registry`	Publishes the web service artifacts to registry servers.
`unpublish-from-registry`	Unpublishes the web service artifacts from the registry servers.
`list-registry-locations`	Displays a list of configured web service entry access points.

Security Service Commands

These security commands are used to control the security mapping for the connector connection pool. These commands are supported in remote mode only.

TABLE 19–19 Security Commands

Command	Definition
`create-connector-security-map`	Creates a security map for the specified connector connection pool. If the security map is not present, a new one is created. Also, use this command to map the caller identity of the application (principal or user group) to a suitable enterprise information system (EIS) principal in container-managed transaction-based scenarios. One or more named security maps may be associated with a connector connection pool. The connector security map configuration supports the use of the wild card asterisk (*) to indicate all users or all user groups. For this command to succeed, you must have first created a connector connection pool. The EIS is any system that holds the data of an organization. It can be a mainframe, a messaging system, a database system, or an application.
`delete-connector-security-map`	Deletes a security map for the specified connector connection pool.
`update-connector-security-map`	Modifies a security map for the specified connector connection pool.
`list-connector-security-maps`	Lists the security maps belonging to the specified connector connection pool.

TABLE 19–19 Security Commands *(Continued)*

Command	Definition
`create-message-security-provider`	Enables administrators to create a `provider-config` sub-element for the given message layer (`message-security-config` element of `domain.xml`, the file that specifies parameters and properties to the Application Server).
`delete-message-security-provide`	Enables administrators to delete a `provider-config` sub-element for the given message layer (`message-security-config` element of `domain.xml`, the file that specifies parameters and properties to the Application Server).
`list-message-security-providers`	Enables administrators to list all security message providers (`provider-config` sub-elements) for the given message layer (`message-security-config` element of `domain.xml`).
`create-auth-realm`	Adds the named authentication realm.
`delete-auth-realm`	Removes the named authentication realm.
`list-auth-realms`	Lists the existing authentication realms.

Password Commands

The password commands allow you to manage passwords and ensure security for the Application Server.

TABLE 19–20 Password Commands

Command	Definition
`create-password-alias`	Creates an alias for a password and stores it in `domain.xml`. An alias is a token of the form `${ALIAS=password-alias-password}`. The password corresponding to the alias name is stored in an encrypted form. This command takes both a secure interactive form (in which the user is prompted for all information) and a more script-friendly form, in which the password is propagated on the command line.
`delete-password-alias`	Deletes a password alias.
`update-password-alias`	Updates the password alias IDs in the named target.
`list-password-aliases`	Lists all password aliases.
`change-admin-password`	This remote command modifies the admin password. This command is interactive in that the user is prompted for the old and new admin password (with confirmation).

TABLE 19–20 Password Commands *(Continued)*

Command	Definition
change-master-password	This local command is used to modify the master password. This command is interactive in that the user is prompted for the old and new master password. This command will not work unless the server is stopped.

Verify domain.xml Command

The XML verify command verifies the content of the domain.xml file.

TABLE 19–21 Verify domain.xml Command

Command	Definition
verify-domain-xml	Verifies the content of the domain.xml file.

Miscellaneous Commands

The miscellaneous commands allow you to manage various aspects of the Application Server.

TABLE 19–22 Miscellaneous Commands

Command	Definition
login	Lets you log into a domain. If various application server domains are created on various machines (locally), asadmin invocation from any of these machines can manage the domains located elsewhere (remotely). This comes in handy especially when a particular machine is chosen as an administration client and it manages multiple domains and servers. asadmin commands that are used to manage domains located elsewhere are called remote commands. The asadmin login command eases the administration of such remote domains. The login command runs only in the interactive mode. It prompts you for the admin user name and password. On successful login, the file .asadminpass will be created in the user's home directory. This is the same file that is modified by the create-domain command while using the --savelogin option. The domain must be running for this command to run.
version	Displays the version information. If the command cannot communicate with the administration server with the given user/password and host/port, then the command will retrieve the version locally and display a warning message.
help	Displays a list of all the asadmin utility commands. Specify the command to display the usage information for that command

TABLE 19–22 Miscellaneous Commands *(Continued)*

Command	Definition
`install-license`	Prevents unauthorized use of the Application Server. Allows you to install the license file.

Index

K

L

M

N

O

P

Q

R

S

T

V

W