Sun Java™ System

Application Server Platform Edition 8 Administration Guide

2004Q2

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

Part No: 817-6088

Contents

About This Guide

This guide describes how to configure and administer the Sun Java™ System Application Server Platform Edition 8. This preface contains information about the following topics:

- Who Should Use This Guide
- Using the Documentation
- How This Guide Is Organized
- Documentation Conventions
- Contacting Sun

Who Should Use This Guide

This guide is intended for information technology administrators in production envrionments. This guide assumes you are familiar with the following topics:

- Basic system administration tasks
- Installing software
- Using Web browsers
- Starting database servers
- Issuing commands in a terminal window

Using the Documentation

The Sun Java™ System Application Server Platform Edition 8 manuals are available as online files in Portable Document Format (PDF) and Hypertext Markup Language (HTML).

The following table lists tasks and concepts described in the Sun Java System Application Server manuals.

Table 1 Sun Java System Application Server Documentation Roadmap

For information about	See the following
Late-breaking information about the software and the documentation. Includes a comprehensive, table-based summary of supported hardware, operating system, JDK, and JDBC/RDBMS.	*Release Notes*
Installing the Sun Java System Application Server software and its components, such as sample applications, the Administration Console, and the high-availability components. Instructions for implementing a basic high-availability configuration are included.	*Installation Guide*
Creating and implementing Java™ 2 Platform, Enterprise Edition (J2EE™ platform) applications intended to run on the Sun Java System Application Server that follow the open Java standards model for J2EE components and APIs. Includes general information about application design, developer tools, security, assembly, deployment, debugging, and creating lifecycle modules. A comprehensive Sun Java System Application Server glossary is included.	*Developer's Guide*
Using J2EE 1.4 platform technologies and APIs to develop J2EE applications and deploying the applications on the Sun Java System Application Server.	*J2EE 1.4 Tutorial*
Information and instructions on the configuration, management, and deployment of the Sun Java System Application Server subsystems and components, from both the Administration Console and the command-line interface. Topics include cluster management, the high-availability database, load balancing, and session persistence. A comprehensive Sun Java System Application Server glossary is included.	*Administration Guide*
Editing the Sun Java System Application Server configuration file, `domain.xml`.	*Reference*
Migrating your applications to the new Sun Java System Application Server programming model, specifically from iPlanet Application Server 6.x and from Netscape Application Server 4.0. Includes a sample migration.	*Migrating and Redeploying Server Applications Guide*
Information on solving Sun Java System Application Server problems.	*Troubleshooting Guide*
Utility commands available with the Sun Java System Application Server; written in manpage style.	*Utility Reference Manual*

Table 1 Sun Java System Application Server Documentation Roadmap *(Continued)*

For information about	See the following
Using the Sun™ Java System Message Queue 3.5 software.	The Sun Java System Message Queue documentation at: `http://docs.sun.com/db?p=prod/s1.s1msgqu`

How This Guide Is Organized

The organization of this guide corresponds to the layout of the Admin Console, the browser-based tool for administering the Application Server. Each chapter begins with conceptual information, followed by procedural sections that explain how to perform specific tasks with the Admin Console.

Documentation Conventions

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

- General Conventions
- Conventions Referring to Directories

General Conventions

The following general conventions are used in this guide:

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

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

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

- **Font conventions** include:
    - The `monospace` font is used for sample code and code listings, API and language elements (such as function names and class names), file names, pathnames, directory names, and HTML tags.
    - *Italic* type is used for code variables.
    - *Italic* type is also used for book titles, emphasis, variables and placeholders, and words used in the literal sense.
    - **Bold** type is used as either a paragraph lead-in or to indicate words used in the literal sense.
- **Installation root directories** for most platforms are indicated by *install_dir* in this document. Exceptions are noted in "Conventions Referring to Directories" on page 13.

  By default, the location of *install_dir* on **most** platforms is:

    - Solaris and Linux file-based installations, non-root user:

      user's home directory`/SUNWappserver`

    - Solaris and Linux file-based installations, root user:

      `/opt/SUNWappserver`

    - Windows, all installations:

      system drive`:\Sun\AppServer`

  For the platforms listed above, *default_config_dir* is identical to *install_dir*. See "Conventions Referring to Directories" on page 13 for exceptions and additional information.

- **Domain root directories** are indicated by *domain_dir* in this document, which by default is an abbreviation for the following:

  install_dir`/domains/`*domain_dir*

  However, for package-based installations, the directory containing all the domains can be changed from *install_dir*`/domains/` to another directory during installation. In configuration files, you may see *domain_dir* represented as follows:

  `${com.sun.aas.instanceRoot}`

- **UNIX-specific descriptions** throughout this manual apply to the Linux operating system as well, except where Linux is specifically mentioned.

Conventions Referring to Directories

By default, when using the Solaris package-based or Linux RPM-based installation, the application server files are spread across several root directories. This guide uses the following document conventions to correspond to the various default installation directories provided:

- *install_dir* refers to `/opt/SUNWappserver`, which is the default location for the static portion of the installation image. All utilities, executables, and libraries that make up the application server reside in this location.
- *default_config_dir* refers to `/var/opt/SUNWappserver/domains`, which is the default location for any domains that are created.

Contacting Sun

You might want to contact Sun Microsystems in order to:

- Give Us Feedback
- Obtain Training
- Contact Product Support

Give Us Feedback

If you have general feedback on the product or documentation, please send this to appserver-feedback@sun.com.

Obtain Training

Application Server training courses are available at:

`http://training.sun.com/US/catalog/enterprise/web_application.html`

Visit this site often for new course availability on the Sun Java System Application Server.

Contact Product Support

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

- The online support web site at:

  `http://www.sun.com/supportraining/`

- The telephone dispatch number associated with your maintenance contract

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

- Description of the problem, including the situation where the problem occurs and its impact on your operation
- Machine type, operating system version, and product version, including any patches and other software that might be affecting the problem. Here are some of the commonly used commands:
  - **Solaris:** `pkginfo, showrev`
  - **Linux:** `rpm`
  - **All:** `asadmin version --verbose`
- Detailed steps on the methods you have used to reproduce the problem
- Any error logs or core dumps
- Configuration files such as:
  - *domain_dir*`/config/domain.xml`
  - a web application’s `web.xml` file,
    when a web application is involved in the problem
- For an application, whether the problem appears when it is running in a cluster or standlone

Chapter 1

Getting Started

This chapter briefly describes the Sun Java™ System Application Server Platform Edition 8 and introduces basic administration tasks. This chapter contains following sections:

- About the Sun Java™ System Application Server Platform Edition 8
- Configuring Domains
- Starting and Stopping a Domain

About the Sun Java™ System Application Server Platform Edition 8

- What is the Sun Java™ System Application Server Platform Edition 8?
- Application Server Architecture
- Administrative Domains
- Tools for Administration
- Configuration Changes and Restarting the Server
- Changing Port Numbers
- Changing the J2SE Software Used By the Application Server
- Further Information

What is the Sun Java™ System Application Server Platform Edition 8?

The Sun Java™ System Application Server Platform Edition 8 provides a robust J2EE platform for the development, deployment, and management of enterprise applications. Key features include transaction management, performance, scalability, security, and integration. The Application Server supports services from Web publishing to enterprise-scale transaction processing, while enabling developers to build applications based on JavaServer Pages (JSP™), Java servlets, and Enterprise JavaBeans™ (EJB™) technology.

The Sun Java™ System Application Server Platform Edition 8 is FREE for development, production deployment, and redistribution. For more information on redistribution, please visit:

```
http://www.sun.com/software/products/appsrvr/appsrvr_oem.html
```

Application Server Architecture

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

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 beans, the components of EJB technology, 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 bean clients communicate with the Object Request Broker (ORB) through the 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 to a specific port number.

Web Services

On the J2EE 1.4 platform, you can 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 can 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 can locate 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 can restrict access to the container's resources and services.
- Transaction management : A transaction is an indivisible unit of work. For example, transfering 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. As an administrator, one of your responsibilities will be resource configuration. The J2EE platform enables access to external systems through the following APIs and componenets:

- 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 can be described as persistent because it is saved on disk and exists after the application ends. The Application Server bundle includes the PointBase DBMS.
- Messaging: Messaging is a method of communication between software components or applications. A messaging client can send messages to, and receive 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 can access an EIS through a portable J2EE component called a connector or resource adapter.
- JavaMail: Through the JavaMail API, applications can connect to an SMTP server in order to send and receive email.

Server Administration

The lower right-hand corner of Figure 1-1 shows some of the tasks performed by the administrator of the Application Server. For example, as an administrator, you will deploy (install) applications and monitor the server's performance. You will perform these tasks with the administration tools provided by the Application Server. See the section Tools for Administration.

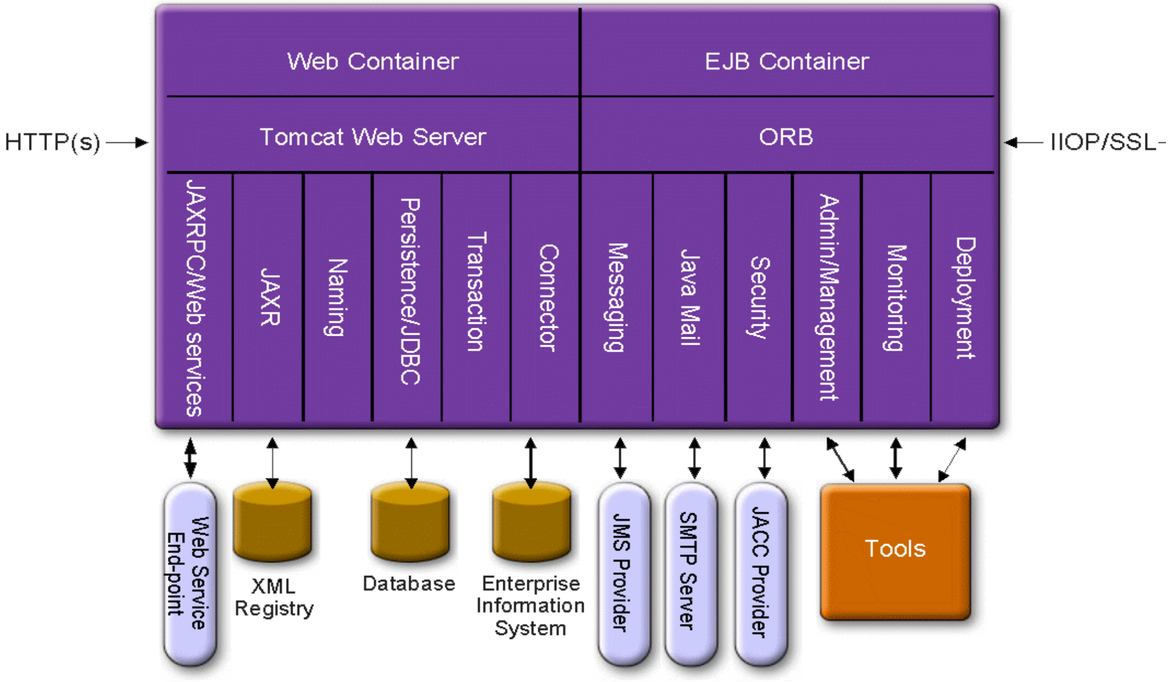


Figure 1-1 Sun Java™ System Application Server Platform Edition 8 Architecture

Administrative Domains

Administrative domains 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 you change the configuration for one domain, the configurations of other domains are not affected.

In an Admin Console session, you can configure and manage a single domain. Each domain has its own administration server, which has a unique port number. You specify the port number in the URL of your browser to run the Admin Console.

In this release, each administrative domain has a single application server instance. An application server instance may belong to just one domain. When you installed the Application Server, an administrative domain named `domain1` was automatically created.

Tools for Administration

The Application Server includes two administrative tools:

- Admin Console
- The asadmin Utility

Admin Console

The Admin Console is a browser-based tool that features an easy-to-navigate interface and online help. This manual provides step-by-step instructions for using the Admin Console. The administration server must be running in order for you to use the Admin Console.

When you installed the Application Server, you chose a port number for the server, or used the default port of 4848. You also specified a user name and password.

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

```
http://hostname:port
```

For example:

```
http://austen.sun.com:4848
```

If you are running the Admin Console on the machine on which the Application Server was installed, you can specify `localhost` for the host name.

On Windows, you can start the Admin Console from the Start menu by choosing Programs -> Sun Microsystems -> J2EE 1.4 SDK -> Admin Console.

The installation program creates the default administrative domain (named `domain1`) with the default port number 4848. After installation, you may choose to create additional administrative domains. Each domain has its own administration server, which has a unique port number. When you specify the URL for the Admin Console, be sure to use the port number for the domain that you want to administer.

The asadmin Utility

The `asadmin` utility is a command-line tool. You can use the `asadmin` utility and the commands associated with it to perform the same set of tasks as you can perform in the Admin Console. For example, you can start and stop domains, configure the server, and deploy applications.

You can use these commands either from a command prompt in the shell, or you can call them from other scripts and programs. You can use these commands to automate repetitive administration tasks.

To start the `asadmin` utility:

```
$ asadmin
```

To list the commands available within `asadmin`:

```
asadmin> help
```

You can also 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 format.

Configuration Changes and Restarting the Server

If you make the following configuration changes, you must restart the server 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 Server.

With dynanamic configuration, most changes will take effect while the server is running. If you make the following configuration changes, you do *NOT* have to restart the server:

- 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.

Changing Port Numbers

This section describes the ports used by the Application Server and explains how to change port numbers with the Admin Console. This section contains the following topics:

- Ports in the Application Server
- Viewing Port Numbers
- Changing the Administrative Server Port
- Changing an HTTP Port

- Changing an IIOP Port

Ports in the Application Server

Table 1-1 describes the the port listeners of the Application Server.

Table 1-1 Application Server Listeners that Use Ports

Listener	Description
Administrative server	A domain’s administrative server is accessed by the Admin Console and the asadmin utility. For the Admin Console, you specify the port number in the URL of the browser. If you execute an asadmin command remotely, you must specify the port number with the --port option.
HTTP	The Web server listens for HTTP requests on a port. To access deployed Web applications and services, clients connect to this port.
HTTPS	Web applications configured for secure communications listen on a separate port.
IIOP	Remote clients of enterprise beans (EJB components) access the beans through the IIOP listener.
IIOP, SSL	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.

Viewing Port Numbers

1. In the tree component, select the Application Server node.
2. Select the General tab.
3. On the General Information page, note the port numbers.

Changing the Administrative Server Port

1. In the tree component, expand the HTTP Service node.
2. Select the HTTP Listeners node.
3. On the HTTP Listeners page, in the Name column select the admin-listener entry.
4. On the Edit HTTP Listener page, change the value of the Listener Port field.
5. Restart the server.

Changing an HTTP Port

1. In the tree component, expand the HTTP Service node.
2. Select the HTTP Listeners node.
3. On the HTTP Listeners page, in the Name column select the the listener whose port number you want to change.
4. On the Edit HTTP Listener page, change the value of the Listener Port field.
5. Click Save.
6. Restart the server.

Changing an IIOP Port

1. In the tree component, expand the ORB node.
2. Select the IIOP Listeners node.
3. On the IIOP Listeners page, in the Name column select the the listener whose port number you want to change.
4. On the Edit IIOP Listener page, change the value of the Listener Port field.
5. Click Save.
6. Restart the server.

Changing the J2SE Software Used By the Application Server

The Application Server relies on the Java 2 Standard Edition (J2SE) software. When you installed the Application Server, you specified the directory of the J2SE software. For instructions on changing the J2SE software, see Configuring the JVM General Settings.

Further Information

- Sun Microsystems Wordlwide Training - Over 250,000 students each year are trained by Sun and its authorized centers through Web-based courses and at over 250 training sites located in more than 60 countries. For more information, see:

  `http://training.sun.com/`

- *The J2EE 1.4 Tutorial* - Written for developers, the tutorial has administrative instructions for configuring JMS, setting up JavaMail resources, and managing security. To access the tutorial, go to this URL:

  `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`

- *Sun Java™ System Application Server Platform Edition 8 Developer's Guide* - This guide contains development information that is specific to the Application Server. The Developer's Guide is available at:

  `http://docs.sun.com/db/doc/817-6087`

- The `asadmin` man pages - Available in HTML format, these pages include syntax and examples for `asadmin` commands. These HTML pages are posted at the following URL:

  `http://docs.sun.com/db/doc/817-6092`

- *Sun Java™ System Application Server Platform Edition 8 Release Notes* - Available online at:

  `http://docs.sun.com/db/doc/817-6082`

- *Getting Started With J2EE Connectors* - This document has instructions for for configuring connectors (resource adapters), connection pools, and connector resources:

  `http://java.sun.com/j2ee/connector/`

- docs.sun.com: Sun Product Documentation - From this site you can search for and access all of our product documentation:

  `http://docs.sun.com/`

- J2EE 1.4 Documentation page - Located on our public Web site, this page has links to the technical documentation for the J2EE 1.4 platform:

  `http://java.sun.com/j2ee/1.4/docs/`

- *The Quick Start Guide* - This document shows you how to deploy and run a simple Web application. The guide is in the *install_dir*`/docs/QuickStart.html` file.

Configuring Domains

- Creating a Domain
- Deleting a Domain

- Listing Domains

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 123 and the admininstrative user name is buzz. The command prompts for the administrative password.

```
$ asadmin create-domain --adminport 123 --adminuser buzz mydomain
```

To start the Admin Console for this domain, in a browser you would enter the following URL:

`http://`*hostname*`:123/asadmin`

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

install_dir`/domains/mydomain`

To create the domain's directory in another location, you specify the --domaindir option. For the full synax 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 *install_dir*/domains directory, type this command:

```
$ asadmin list-domains
```

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

Starting and Stopping a Domain

- Starting a Domain
- Restarting the Server
- Stopping a Domain

Starting a Domain

When you start a domain, you start its administration server and application server instance. 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, use one of the following methods.

Starting With the asadmin start-domain Command

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

```
$ asadmin start-domain domain1
```

If there is only one domain, you may omit the domain name. For the full command syntax, type `asadmin help start-domain`.

Starting With the Windows Start Menu

On Windows, you can start the default domain from the Windows Start menu by choosing Programs -> Sun Microsystems -> J2EE 1.4 SDK -> Start Default Server.

Restarting the Server

In this release, restarting the server is the same as restarting the domain. To restart the domain, you stop and start the domain.

Stopping a Domain

Stopping a domain shuts down its administration server and application server instance. When you stop a domain, the server instance stops accepting new connections and then waits for all outstanding connections to complete. It may take a few seconds for the server instance to complete its shut-down process. While the domain is stopped, you cannot use the Admin Console or most `asadmin` commands.

Stopping With the asadmin stop-domain Command

To stop a domain, you type the `asadmin stop-domain` command and specify the domain name. For example, to stop the default domain (`domain1`), you 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 With the Admin Console

1. In the tree component, select the Application Server node.
2. Select the General tab.
3. On the General Information page, click Stop Server.

Stopping With the Windows Start Menu

On Windows, you can stop the default domain from the Start menu by choosing Programs -> Sun Microsystems -> J2EE 1.4 SDK -> Stop Default Server.

Chapter 2

Application Deployment

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

- About Deployment
- Admin Console Tasks for Deploying Applications
- Admin Console Tasks for Listing, Undeploying, and Enabling Applications
- Deployment Methods for Developers

About Deployment

- The Deployment Life Cycle
- Types of J2EE JAR Files
- Naming Conventions

The Deployment Life Cycle

After you've installed the Application Server and started a domain, you can deploy (install) J2EE applications and modules. During deployment, an application or module can go through the following stages:

1. Initial Deployment

   You deploy (install) an application or module into a specific domain. Before you deploy an application or module, you must first start the domain. Because applications and modules are packaged in JAR files, you specify the JAR file name during deployment.

   Deployment is dynamic: you don't need to restart the domain when you deploy an application. If you do restart the domain, all deployed applications and modules are still deployed.

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. To prevent access, you can disable the application or module. A disabled application or module is not uninstalled from the domain and can be easily enabled.

3. Redeployment

   To replace a deployed application or module, you redeploy it. You don't have to undeploy and then deploy it again. Redeployment is common in a development environment, but less so in a production environment.

4. Undeployment

   To uninstall an application or module, you undeploy it. Undeployment does not delete the JAR file containing the application or module.

Types of J2EE JAR Files

A software provider packages an application or module into a JAR file. To deploy the application or module, you specify the JAR file name. The content and structure of the JAR file is defined by the specifications of the J2EE platform. Types of J2EE JAR 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 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 Admin 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 specifications, a resource adapter is a portable component that enables enterprise beans and Web components 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, 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. With separate files, you can distribute the application by deploying it on different servers. 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 you deploy with the Admin Console, then you specify the name in the Application Name field. If you use the `asadmin deploy` command, the default name of the application or module is the prefix of the JAR file that you deploy. For example, if you deploy the `hello.war` file, the Web application name will be `hello`. You can override the default name by specifying the `--name` option.

Admin Console Tasks for Deploying Applications

- Deploying an Enterprise Application
- Deploying a Web Application
- Deploying an EJB Module
- Deploying an Application Client Module
- Deploying a Connector Module
- Creating a Lifecycle Module

- Deploying an Application Client Module

Deploying an Enterprise Application

An enterprise application is packaged in an EAR file, a type of JAR file that contains module files such as WAR and EJB JAR files.

To deploy (install) an enterprise application:

1. In the tree component, expand the Applications node.
2. Select the Enterprise Applications node.
3. On the Enterprise Applications page, click Deploy.
4. On the Deployment page, specify the EAR file that you want to deploy.

   In the explanation of this step, the server machine is the host that is running the application server. The client machine is the host on which you are running the Admin Console browser. If a file or directory is said to reside on a server or client machine, it might really live on another host, but it must be accessible from the file system of the server or client machine.

   a. For the Upload File button, if the file resides on the client machine, then select Yes. If the file resides on the server machine, then select No. If you want to deploy an unpackaged application from an exploded directory, then select No. The exploded directory must reside on the server machine. Deploying from an exploded directory is for advanced developers and is not recommended for production environments.

   b. In the File Or Directory field, if you are deploying a file then click Browse and locate the file (or type the full path name of the file). If you are deploying from an exploded directory, then type the full path name of the directory.

5. Click Next to display the Deploy Enterprise Application page.
6. On the Deploy Enterprise Application page, specify the settings for the application.

   a. In the Application Name field, you may retain the default name, which is the prefix of the file name, or you may type another name. The application name must be unique.

   b. In the Virtual Servers, you may replace the default `server`. (To view the available virtual servers, in the tree component select HTTP Service -> Virtual Servers.)

c. By default, an application is available as soon as it is deployed. If you want to disable the application so that is unavailable after deployment, deselect the Enabled checkbox.

d. If the application has already been deployed, it will be replaced with the new application unless you deselect the Redeploy checkbox.

e. If you want to verify the structure and contents of the file before deployment, select the Verifier checkbox. Verification of large applications can be time-consuming, but should be done if you suspect the file is corrupt or non-portable.

f. If you want to precompile JSP pages, select the JSPs checkbox. If you do not select this checkbox, the JSP pages will be compiled at runtime when they are first accessed. Because compilation can be time-consuming, in a production environment you should select this checkbox.

7. Click Finish to deploy the application.

Equivalent `asadmin` command: `deploy`

Deploying a Web Application

An Web application is packaged in a WAR file, a type of JAR file that contains components such as servlets and JSP pages.

To deploy (install) a Web application:

1. In the tree component, expand the Applications node.

2. Select the Web Applications node.

3. On the Web Applications page, click Deploy.

4. On the Deployment page, specify the WAR file that you want to deploy.

   In the explanation of this step, the server machine is the host that is running the application server. The client machine is the host on which you are running the Admin Console browser. If a file or directory is said to reside on a server or client machine, it might really live on another host, but it must be accessible from the file system of the server or client machine.

   a. For the Upload File button, if the file resides on the client machine, then select Yes. If the file resides on the server machine, then select No. If you want to deploy an unpackaged application from an exploded directory, then select No. The exploded directory must reside on the server machine. Deploying from an exploded directory is for advanced developers and is not recommended for production environments.

   b. In the File Or Directory field, if you are deploying a file then click Browse and locate the file (or type the full path name of the file). If you are deploying from an exploded directory, then type the full path name of the directory.

5. Click Next to display the Deploy Web Application page.

6. On the Deploy Web Application page, specify the settings for the application.

   a. In the Application Name field, you may retain the default name, which is the prefix of the file name, or you may type another name. (The default name appears if you select Yes for the Upload File button.) The application name must be unique.

   b. In the Context Root field, enter a string that identifies the Web application. In the URL of the Web application, the context root immediately follows the port number (`http://host:port/context-root/...`). Make sure that the context root starts with a forward slash, for example: `/hello`

   c. In the Virtual Servers, you may replace the default `server`. (To view the available virtual servers, in the tree component select HTTP Service -> Virtual Servers.)

   d. By default, an application is available as soon as it is deployed. If you want to disable the application so that is unavailable after deployment, deselect the Enabled checkbox.

   e. If the application has already been deployed, it will be replaced with the new application unless you deselect the Redeploy checkbox.

   f. If you want to verify the structure and contents of the file before deployment, select the Verifier checkbox. Verification of large applications can be time-consuming, but should be done if you suspect the file is corrupt or non-portable.

   g. If you want to precompile JSP pages, select the JSPs checkbox. If you do not select this checkbox, the JSP pages will be compiled at runtime when they are first accessed. Because compilation can be time-consuming, in a production environment you should select this checkbox.

7. Click Finish to deploy the application.

Equivalent `asadmin` command: `deploy`

Deploying an EJB Module

An EJB Module, also called an EJB JAR file, contains enterprise beans.

To deploy (install) an EJB module:

1. In the tree component, expand the Applications node.
2. Select the EJB Modules node.
3. On the EJB Module page, click Deploy.
4. On the Deployment page, specify the JAR file that you want to deploy.

   In the explanation of this step, the server machine is the host that is running the application server. The client machine is the host on which you are running the Admin Console browser. If a file or directory is said to reside on a server or client machine, it might really live on another host, but it must be accessible from the file system of the server or client machine.

   a. For the Upload File button, if the file resides on the client machine, then select Yes. If the file resides on the server machine, then select No. If you want to deploy an unpackaged application from an exploded directory, then select No. The exploded directory must reside on the server machine. Deploying from an exploded directory is for advanced developers and is not recommended for production environments.

   b. In the File Or Directory field, if you are deploying a file then click Browse and locate the file (or type the full path name of the file). If you are deploying from an exploded directory, then type the full path name of the directory.

5. Click Next to display the Deploy EJB Module page.
6. On the Deploy EJB Module page, specify the settings for the application.

   a. In the Application Name field, you may retain the default name, which is the prefix of the file name, or you may type another name. The application name must be unique.

   b. By default, an application is available as soon as it is deployed. If you want to disable the application so that is unavailable after deployment, deselect the Enabled checkbox.

c. If the application has already been deployed, it will be replaced with the new application unless you deselect the Redeploy checkbox.

d. If you want to verify the structure and contents of the file before deployment, select the Verifier checkbox. Verification of large applications can be time-consuming, but should be done if you suspect the file is corrupt or non-portable.

7. Click Finish to deploy the module.

Equivalent `asadmin` command: `deploy`

Deploying a Connector Module

A connector, also known as a resource adapter, is packaged in a type of JAR file called a RAR file.

To deploy (install) a connector module:

1. In the tree component, expand the Applications node.
2. Select the Connector Modules node.
3. On the Connector Modles page, click Deploy.
4. On the Deployment page, specify the RAR file that you want to deploy.

   In the explanation of this step, the server machine is the host that is running the application server. The client machine is the host on which you are running the Admin Console browser. If a file or directory is said to reside on a server or client machine, it might really live on another host, but it must be accessible from the file system of the server or client machine.

   a. For the Upload File button, if the file resides on the client machine, then select Yes. If the file resides on the server machine, then select No. If you want to deploy an unpackaged application from an exploded directory, then select No. The exploded directory must reside on the server machine. Deploying from an exploded directory is for advanced developers and is not recommended for production environments.

   b. In the File Or Directory field, if you are deploying a file then click Browse and locate the file (or type the full path name of the file). If you are deploying from an exploded directory, then type the full path name of the directory.

5. Click Next to display the Deploy Connector Module page.

6. On the Deploy Connector Module page, specify the settings for the application.

   a. In the Application Name field, you may retain the default name, which is the prefix of the file name, or you may type another name. The application name must be unique.

   b. In the Thread Pool Id field, you may specify the thread pool for the connector that you are deploying.

   c. By default, an application is available as soon as it is deployed. If you want to disable the application so that it is unavailable after deployment, deselect the Enabled checkbox.

      When you enable or disable a connector module, you also enable or disable the connector resources and connection pools that point to the module.

   d. If the application has already been deployed, it will be replaced with the new application unless you deselect the Redeploy checkbox.

   e. If you want to verify the structure and contents of the file before deployment, select the Verifier checkbox. Verification of large applications can be time-consuming, but should be done if you suspect the file is corrupt or non-portable.

7. Click Finish to deploy the module.

Equivalent `asadmin` command: `deploy`

Creating a Lifecycle Module

A lifecycle module performs tasks when it is triggered by one or more events in the server lifecycle. These server events are: initialization, start up, ready to service requests, and shut down. Lifecycle modules are not part of the J2EE specification, but are an enhancement to the Sun Java™ System Application Server Platform Edition 8.

To create a lifecycle module:

1. In the tree component, expand the Applications node.

2. Select the Lifecycle Modules node.

3. On the Lifecycle Modules page, click New.

4. On the Create Lifecycle Module page, specify the settings:

   a. In the Name field, type a name that denotes the function of the module.

b. In the Class Name field, type the fully qualified name of the lifecycle module's class file.

c. If the JAR file containing the lifecycle is in the server's classpath, then leave the Classpath field blank. Otherwise, type the fully qualified path.

d. In the Load Order field, type an integer greater than 100 and less than the operating system's `MAXINT` value.

   The integer determines the order in which lifecycle modules are loaded when the server starts up. Modules with smaller integers are loaded sooner.

e. When you start the server, it loads lifecycle modules that have already been deployed. By default, if a load fails, the server will continue the start up operation. If you want to prevent the server from starting up when a load files, then select the On Load Failure checkbox.

f. By default, a module is available as soon as it is deployed. If you want to disable the module so that is unavailable after deployment, deselect the Enabled checkbox.

5. Click OK.

Equivalent `asadmin` command: `create-lifecycle-module`

Deploying an Application Client Module

An application client module, also called a J2EE application client JAR file, contains the server-side routines for the client.

To deploy (install) an application client module:

1. In the tree component, expand the Applications node.
2. Select the App Client Modules node.
3. On the Application Client Modules page, click Deploy.
4. On the Deployment page, specify the JAR file that you want to deploy.

   In the explanation of this step, the server machine is the host that is running the application server. The client machine is the host on which you are running the Admin Console browser. If a file or directory is said to reside on a server or client machine, it might really live on another host, but it must be accessible from the file system of the server or client machine.

   a. For the Upload File button, if the file resides on the client machine, then select Yes. If the file resides on the server machine, then select No. If you want to deploy an unpackaged application from an exploded directory, then select No. The exploded directory must reside on the server machine. Deploying from an exploded directory is for advanced developers and is not recommended for production environments.

   b. In the File Or Directory field, if you are deploying a file then click Browse and locate the file (or type the full path name of the file). If you are deploying from an exploded directory, then type the full path name of the directory.

5. Click Next to display the Deploy Application Client Module page.

6. On the Deploy Application Client Module page, specify the settings for the application.

   a. In the Application Name field, you may retain the default name, which is the prefix of the file name, or you may type another name. The application name must be unique.

   b. If the application has already been deployed, it will be replaced with the new application unless you deselect the Redeploy checkbox.

   c. If you want to verify the structure and contents of the file before deployment, select the Verifier checkbox. Verification of large applications can be time-consuming, but should be done if you suspect the file is corrupt or non-portable.

7. Click Finish to deploy the module.

For the client-side routines:

- Typically, the application provider will ship a JAR file containing the client-side routines.
- The application provider can get the client-side stubs by specifying the `--retrieve` oiption of the `asadmin deploy` command.

Equivalent `asadmin` command: `deploy`

Admin Console Tasks for Listing, Undeploying, and Enabling Applications

- Listing Deployed Applications

- Listing Subcomponents
- Undeploying an Application
- Enabling and Disabling an Application
- Enabling and Disabling Dynamic Reloading

Listing Deployed Applications

1. In the tree component, expand the Applications node.
2. Expand the node for the application or module type.

To view the details of a deployed application or module you can either:

- In the tree component, select the node of the application or module.
- On the page, select an entry in the Application Name column.

Equivalent `asadmin` command: `list-components`

Listing Subcomponents

Enterprise and Web applications contain subcomponents. For example, a Web application may contain one or more servlets. You use the same method for listing the subcomponents of Enterprise and Web applications.

To list the subcomponents of a Web application:

1. In the tree component, expand the Applications node.
2. Expand the Web Applications node.
3. Select the node for a particular Web application.
4. On the Web Application page, note the contents of the Sub Components table.

Equivalent `asadmin` command: `list-components`

Undeploying an Application

When you undeploy an applicaton or module, you uninstall it from the application server. This action will not delete the JAR file in which the application or module has been packaged.

To undeploy an application or module:

1. In the tree component, expand the Applications node.
2. Select the node for the type of application or module want to undeploy.
3. In the table listing the deployed applications, select the checkbox for the application you want to undeploy.
4. Click Undeploy.

Equivalent `asadmin` command: `undeploy`

Enabling and Disabling an Application

If a deployed application or module is enabled, it can be accessed by clients. If it is disabled, then it is still deployed but is not accessible by clients. By default, when you deploy an application or module, it is enabled because the Status checkbox on the Deploy page is selected.

To enable a deployed application or module:

1. In the tree component, expand the Applications node.
2. Expand the node for the application type.
3. Select the node for the deployed application or module.
4. Select the Status checkbox.

To disable a deployed application or module, deselect the Status checkbox.

Equivalent `asadmin` commands: `enable` and `disable`

Enabling and Disabling Dynamic Reloading

If dynamic reloading is enabled, the server periodically checks for changes in the files of the deployed application and automatically reloads the application with the changes. Dynamic reloading is useful in a development environment because it allows code changes to be tested quickly. In a production environment, however, dynamic reloading may degrade performance.

To configure dynamic reloading:

1. In the tree component, select Applications.
2. On the Applications Configuration page, you can configure the following:

- ❍ Reload: You can enable or disable dynamic reloading with the Enabled checkbox.
- ❍ Reload Poll Interval: You can specify how often the server checks for changes in the deployed applications.

Deployment Methods for Developers

- Using Auto Deploy
- Deploying an Unpackaged Application From a Directory
- Using the deploytool Utility
- Using a Deployment Plan

Using Auto Deploy

The auto deploy feature enables you to deploy a pre-packaged application or module by copying it to the *install_dir*/`domains`/*domain_dir*/`autodeploy` directory. For example, you could copy a file named `hello.war` to the *install_dir*/`domains/domain1/autodeploy` directory. To undeploy the application, you would remove the `hello.war` file from the `autodeploy` directory.

To configure the auto deploy feature:

1. In the tree component, select Applications.
2. On the Applications Configuration page, you can configure the following:
    a. You can enable or disable auto deploy by selecting or deselecting the Enabled checkbox.
    b. In the Auto Deploy Poll Interval field, you specify how often the server checks the auto deploy directory for application or module files. Changing the poll interval will not affect the amount of time it takes to deploy an application or module.
    c. In the Auto Deploy Directory, if you specify the directory where you build your application, then you won’t have to copy the file to the default auto deploy directory.
    d. Verifier: To run the verifier before deployment, select the Enabled checkbox. The verifier will examine the structure and content of the file. Verification of large applications can be time-consuming.

e. If you want to precompile JSP pages, select the JSPs checkbox. If you do not select this checkbox, the JSP pages will be compiled at runtime when they are first accessed. Because compilation can be time-consuming, in a production environment you should select this checkbox.

Deploying an Unpackaged Application From a Directory

This feature is for advanced developers.

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 JAR 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 dynamic reloading is enabled, then the server automatically detects the changes and reloads the application. See Enabling and Disabling Dynamic Reloading.

To deploy an unpackaged application from a directory:

1. In the Admin Console, begin the deployment process. See Deploying a Web Application.
2. On the Deployment page, specify the following:
   a. Select No.

   The directory specified on the Deployment page must reside on the server machine.

   b. In the File Or Directory field, enter the name of the exploded directory.

Equivalent `asadmin` command: `deploydir`

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*.

Using a Deployment Plan

This feature is for advanced developers.

A deployment plan is a JAR file that contains only the deployment descriptiors that are specific to the Sun Java™ System Application Server Platform Edition 8. These deployment descriptors, for example `sun-application.xml`, are described in the *Sun Java™ System Application Server Platform Edition 8 Developer's Guide*. The deployment plan is part of the implementation of *JSR 88: J2EE Application Deployment*. You use a deployment plan when you want to deploy an application or module that does not contain the deployment descriptors that are specific to the Sun Java™ System Application Server Platform Edition 8.

To deploy using a deployment plan, you 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 Sun Java™ System Application Server Platform Edition 8 is at the root level. If a stand-alone EJB module contains a CMP bean, then 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
```

Chapter 3

JDBC Resources

This chapter explains how to configure JDBC resources, which are required by applications that access databases. This chapter contains the following sections:

- About JDBC Resources
- Setting Up Database Access
- Admin Console Tasks for JDBC Connection Pools
- Admin Console Tasks for JDBC Resources
- Admin Console Tasks for Persistence Manager Resources

About JDBC Resources

- JDBC Resources
- JDBC Connection Pools
- How JDBC Resources and Connection Pools Work Together

JDBC Resources

To store, organize, and retrieve data, most applications use relational databases. J2EE applications access relational databases through the JDBC API.

A JDBC resource (data source) provides applications with a means of connecting to a database. Typically, you'll create a JDBC resource for each database accessed by the applications deployed in a domain. (However, you may create more than one JDBC resource for a database.)

When you create a JDBC resource, you specify a unique JNDI name that identifies the resource. (See the section JNDI Names and Resources.) The JNDI name of a JDBC resource should be 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 you specify the JNDI name of a JDBC resource in the Admin Console, you enter only `jdbc/`*`name`*. For example, for a payroll database you would specify `jdbc/payrolldb`.

JDBC Connection Pools

When you create a JDBC resource, you specify the connection pool that it is associated with. Multiple JDBC resources may 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.

The properties of connection pools may vary with different database vendors. Some common properties are the database's name (URL), user name, and password.

How JDBC Resources and Connection Pools Work Together

To store, organize, and retrieve data, most applications use relational databases. J2EE 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 datase:

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 application 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 that needs it.

Setting Up Database Access

- General Steps for Setting Up Database Access
- Integrating a JDBC Driver

General Steps for Setting Up Database Access

1. Install a supported database product. For a list of database products supported by the Sun Java™ System Application Server Platform Edition 8, see the link to the *Release Notes* in the section Further Information.
2. Install a JDBC driver for the database product.
3. Make the driver’s JAR file accessible to the domain’s server instance. See Integrating a JDBC Driver.
4. Create the database. Usually, the application provider delivers scripts for creating and populating the database.
5. Create a connection pool for the database. See Creating a JDBC Connection Pool.

6. Create a JDBC resource that points to the connection pool. See Creating a JDBC Resource.

Integrating a JDBC Driver

A JDBC driver translates an application's JDBC calls into the protocol of the database server. To integrate the JDBC driver into an administrative domain, you can do either of the following:

- Make the driver accessible to the common class loader.
  - **a.** Copy the driver's JAR and ZIP files into the *install_dir*`/domains/`*domain_dir*`/lib` directory or copy its class files into the *install_dir*`/domains/`*domain_dir*`/lib/classes` directory
  - **b.** Restart the domain.
- Make the driver accessible to the system class loader.
  - **a.** In the Admin Console's tree view, select Application Server.
  - **b.** On the Application Server page, select the JVM Settings tab.
  - **c.** On the JVM General Settings page, click the Path Settings option.
  - **d.** On the JVM Classpath Settings page, in the Classpath Suffix field, enter the fully-qualified path name for the driver's JAR file.
  - **e.** Click Save.
  - **f.** Restart the server.

Admin Console Tasks for JDBC Connection Pools

- Creating a JDBC Connection Pool
- Editing a JDBC Connection Pool
- Deleting a JDBC Connection Pool

Creating a JDBC Connection Pool

A JDBC connection pool is a group of reusable connections for a particular database. When you create the pool with the Admin Console, 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.

As you step through the Create Connection Pool pages, you'll enter information that depends on the specific JDBC driver and database vendor. Before proceeding, gather the following information:

- database vendor
- 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

To create a JDBC connection pool:

1. In the tree component, expand the JDBC node.
2. Select the Connection Pools node.
3. On the Connection Pools page, click New.
4. On the first Create Connection Pool page, specify the following general settings:
   a. In the Name field, enter a logical name for the pool.

      You will specify this name when you create a JDBC resource.

   b. Select an entry from the Resource Type combo box.
   c. Select an entry from the Database Vendor combo box.
5. Click Next.
6. On the second Create Connection Pool page, specify the value for the DataSource Class Name field.

   If the JDBC driver has a DataSource class for the resource type and database vendor you specified in the previous page, then the value of the DataSource Class Name field is provided.

7. Click Next.

8. On the third and last Create Connection Pool page, perform these tasks:

   a. In the General Settings section verify that the values are correct.

   b. For the fields in the Pool Settings, Connection Validation, and Transaction Isolation sections, you may retain the default values.

      You can change these settings at a later time. See Editing a JDBC Connection Pool.

   c. In the Additonal Properties table, add the required properties, such as database name (URL), user name, and password.

9. Click Finish.

Equivalent `asadmin` command: `create-jdbc-connection-pool`

Editing a JDBC Connection Pool

The Edit JDBC Connection Pool page enables you to change all of the settings for an existing pool, except its name.

To access the Edit JDBC Connection Pool page:

1. In the tree component, expand the JDBC node.

2. Expand the Connection Pools node.

3. Select the node for the pool you want to edit.

4. On the Edit JDBC Connection Pool page, make the necessary changes.

   See the following sections for explanations of the settings that you may change.

5. Click Save.

The Edit JDBC Connection Pool page is divided into these sections:

- General Settings
- Pool Settings
- Connection Validation
- Transaction Isolation
- Properties

General Settings

The values of the general settings depend on the specific JDBC driver that you've installed. These settings are the names of classes or interfaces in the Java programming language.

Table 3-1 General Settings for a JDBC Connection Pool

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.ConnectinPoolDataSource (local transactions, possible performance improvements).

Pool Settings

A set of physical database connections 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.

Table 3-2 Pool Settings for a JDBC Connection 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 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 timeout. Because the default wait time is long, the application might appear to hang indefinitely.

Connection Validation

Optionally, the application server can validate connections before they are passed to applications. This validation allows the application server to automatically re-establish database connections if the database becomes unavailable due to network failure or database server crash. Validation of connections will incur additional overhead and slightly reduce performance.

Table 3-3 Connection Validation Settings for a JDBC Connection Pool

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, meta-data, and table. auto-commit and meta-data: 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. You should check with your driver vendor to determine whether these calls are cached or not. table: The application queries a database table thay you've specified. The table must exist and be accessible, but it doesn't require any rows. You should 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 will close all connections in the pool and then re-establish them. If you do not select the checkbox, then individual connections will be re-established only when they are used.

Transaction Isolation

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.

Table 3-4 Transaction Isolation Settings for a JDBC Connection Pool

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

Properties

In the Additional Properties table, you can specify properties such as the database name (URL), user name, and password. Because the properties vary with database vendor, you should consult the vendor's documentation for details.

Pinging the Database

To verify the connection pool settings:

1. Start the database server.
2. Click Ping.

   The Admin Console will attempt to connect to the database. If you get an error message, you probably forgot to start the database server.

Deleting a JDBC Connection Pool

1. In the tree component, expand the JDBC node.
2. Select the Connection Pools node.
3. On the Connection Pools page, select the checkbox for the pool that you want to delete.
4. Click Delete.

Equivalent `asadmin` command: `delete-jdbc-connection-pool`

Admin Console Tasks for JDBC Resources

- Creating a JDBC Resource
- Editing a JDBC Resource
- Deleting a JDBC Resource

Creating a JDBC Resource

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

To create a JDBC resource:

1. In the tree component, expand the JDBC node.
2. Expand the JDBC Resources node.
3. On the JDBC Resources page, click New.
4. On the Create JDBC Resources page, specify the resource's settings:
   a. In the JNDI Name field, type a uniqe name. By convention, the name should begin with the `jdbc/` string. For example: `jdbc/payrolldb`. Don't forget the forward slash.
   b. From the Pool Name combo box, choose the connection pool that the new JDBC resource will belong to.
   c. By default, the resource is available (enabled) as soon as it is created. If you want the resource to be unavailable, deselect the Enabled checkbox.
5. Click OK.

Equivalent `asadmin` command:`create-jdbc-resource`

Editing a JDBC Resource

1. In the tree component, expand the JDBC node.
2. Expand the JDBC Resources node.
3. Select the node for the JDBC resource that you want to edit.
4. On the Edit JDBC Resources page, you can perform these tasks:

   a. From the Pool Name combo box, select a different connection pool.

   b. Select or deselect the checkbox to enable or disable the resource.

5. Click Save to apply the edits you have made.

Deleting a JDBC Resource

1. In the tree component, expand the JDBC node.
2. Select the JDBC Resources node.
3. On the JDBC Resources page, select the checkbox for the resource that you want to delete.
4. Click Delete.

Equivalent `asadmin` command: `delete-jdbc-resource`

Admin Console Tasks for Persistence Manager Resources

- Creating a Persistence Manager Resource

Creating a Persistence Manager Resource

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

To create a persistence manager resource:

1. In the tree component, select the Persistence Managers node.
2. On the Persistence Managers page, click New.
3. On the Create Persistence Manager page, specify these settings:

   a. In the JNDI Name field, type a uniqe name. gor example: `jdo/mypm`. Don't forget the forward slash.

   b. In the Factory Class field, retain the default class provided with this release, or type in the class of another implmementation.

c. From the Connection Pool combo box, choose the connection pool that the new persistence manager resource will belong to.

d. By default, the new peristence manager resource will be enabled. To disable it, deselct the Enabled check box.

Equivalent `asadmin` command: `create-persistence-resource`

Chapter 4

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
- Admin Console Tasks for JMS Connection Factories
- Admin Console Tasks for JMS Physical Destinations
- Admin Console Tasks for JMS Destination Resources
- Admin Console Tasks for the JMS Provider

About JMS Resources

- The JMS Provider in the Sun Java™ System Application Server Platform Edition 8
- JMS Resources
- The Relationship Between JMS Resources and Connector Resources

The JMS Provider in the Sun Java™ System Application Server Platform Edition 8

The Sun Java™ System Application Server Platform Edition 8 implements the Java Message Service (JMS) API by integrating the Sun Java System Message Queue (formerly Sun ONE Message Queue) into the Application Server. You do not need to know anything about the Message Queue product in order to use the JMS API with the Application Server. However, if you are familiar with Message Queue, you can use the tools provided in the *install_dir*`/imq/bin` directory.

For details about administering Message Queue, see the *Sun ONE Message Queue 3.5 Administrator's Guide* at `http://docs.sun.com/db/doc/817-3727/`.

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 must be created administratively, and how they are created is specific to each implementation of JMS. In the Sun Java™ System Application Server Platform Edition 8, you perform the following tasks:

- You create a connection factory by creating a connection factory resource that an application accesses using the JNDI API
- You create a destination by creating two objects:
  - A physical destination
  - A destination resource that refers to the physical destination and that an application accesses using the JNDI API

A JMS application normally uses at least one connection factory and at least one destination. You need to understand the application or consult with the application developer to learn what resources to create.

There are two categories of administered objects:

- `QueueConnectionFactory` and `Queue` objects, used for point-to-point communication

- TopicConnectionFactory and Topic objects, used for publish-subscribe communication

The chapters on JMS in the *J2EE 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 you create the resources does not matter.

For a J2EE application, you normally specify connection factory and destination resources in the Application Server deployment descriptors. You specify a connection factory JNDI name in a resource-ref or an mdb-connection-factory element. You specify a destination resource JNDI name in the ejb element for a message-driven bean and in the message-destination element.

You normally 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. You also specify it in the message-destination element. (The message-destination-ref element replaces the resource-ref element, which should not be used in new applications.) In the message-destination element of an Application Server deployment descriptor, you link the physical destination name with the destination resource name.

The Relationship Between JMS Resources and Connector Resources

The Sun Java™ System Application Server Platform Edition 8 implements JMS by using a system resource adapter named jmsra. When you create JMS resources, the Application Server also automatically creates connector resources that you can see when you expand the Connectors node in the Admin Console's tree view.

For each JMS connection factory you create, the Application Server creates a corresponding connector connection pool and connector resource. For each JMS destination you create, the Application Server creates an admin object resource. When you delete the JMS resources, the corresponding connector resources are also deleted automatically.

You can also create connector resources for the JMS system resource adapter by using the Connectors node of the Admin Console instead of the Java Message Service node. See Chapter 7, "Connector Resources", for details. You must still use the Java Message Service node to create physical destinations.

Admin Console Tasks for JMS Connection Factories

- Creating a JMS Connection Factory Resource
- Editing a JMS Connection Factory Resource
- Deleting a JMS Connection Factory Resource

Creating a JMS Connection Factory Resource

1. In the tree component, expand the Java Message Service node.
2. Select the Connection Factories node.
3. On the JMS Connection Factories page, click New. The Create JMS Connection Factory page appears.
4. In the JNDI Name field, type the name of the connection factory. For example:

   `jms/QueueConnectionFactory`

   It is a recommended practice to use the naming subcontext prefix `jms/` for JMS resources.
5. From the Type combo box, choose either `javax.jms.QueueConnectionFactory` or `javax.jms.TopicConnectionFactory`.
6. Select the Enabled checkbox to enable the resource at run time.
7. In the Additional Properties area, click Add Property to add a property required by your application. Table 4-1 lists the properties you are most likely to set. Others are listed in the Message Queue documentation.
8. Click OK to save the connection factory.

Equivalent `asadmin` command: `create-jms-resource`

Table 4-1 Additional Properties for JMS Connection Factories

Property Name	Description
`ClientId`	Specifies a client ID for a connection factory that will be used by a durable subscriber.

Table 4-1 Additional Properties for JMS Connection Factories

Property Name	Description
`MessageServiceAddressList`	Specifies the name (and, optionally, port number) of a remote system with which your application will communicate. For example, the value could be `earth` or `earth:7677`. Specify the port number if the JMS service on the remote system is running on a port other than the default (7676).

Editing a JMS Connection Factory Resource

1. In the tree component, expand the Java Message Service node.
2. Expand the Connection Factories node.
3. Select the connection factory to be edited.
4. On the Edit JMS Connection Factory page, you can perform these tasks:
   - Modify the text in the Description field.
   - Select or deselect the Enabled checkbox to enable or disable the resource.
   - Add, remove, or modify properties.
5. Click Save to save your changes.

Deleting a JMS Connection Factory Resource

1. In the tree component, expand the Java Message Service node.
2. Select the Connection Factories node.
3. On the JMS Connection Factories page, select the checkbox next to the name of the connection factory to be deleted.
4. Click Delete.

Equivalent `asadmin` command: `delete-jms-resource`

Admin Console Tasks for JMS Physical Destinations

- Creating a JMS Physical Destination
- Deleting a JMS Physical Destination

Creating a JMS Physical Destination

1. In the tree component, expand the Java Message Service node.
2. Select the Physical Destinations node.
3. On the Physical Destinations page, click New. The Create Physical Destination page appears.
4. In the Physical Destination Name field, type the name of the destination (for example, `PhysicalQueue`).
5. From the Type combo box, choose either `topic` or `queue`.
6. Click OK.

Equivalent `asadmin` command: `create-jmsdest`

Deleting a JMS Physical Destination

1. In the tree component, expand the Java Message Service node.
2. Select the Physical Destinations node.
3. On the Physical Destinations page, select the checkbox next to the name of the destination to be deleted.
4. Click Delete.

Equivalent `asadmin` command: `delete-jmsdest`

Admin Console Tasks for JMS Destination Resources

- Creating a JMS Destination Resource

- Editing a JMS Destination Resource
- Deleting a JMS Destination Resource

Creating a JMS Destination Resource

1. In the tree component, expand the Java Message Service node.
2. Select the Destination Resources node.
3. On the JMS Destination Resources page, click New. The Create JMS Destination Resource page appears.
4. In the JNDI Name field, type the name of the resource. For example:

   `jms/Queue`

   It is a recommended practice to use the naming subcontext prefix `jms/` for JMS resources.
5. From the Type combo box, choose either `javax.jms.Topic` or `javax.jms.Queue`.
6. Select the Enabled checkbox to enable the resource at run time.
7. Under Additional Properties, click Add.
8. In the Name field, type `Name`.
9. In the Value field, type the name of the physical destination to which the resource will refer.
10. Click OK.

Equivalent `asadmin` command: `create-jms-resource`

Editing a JMS Destination Resource

1. In the tree component, expand the Java Message Service node.
2. Expand the Destination Resources node.
3. Select the destination resource to be edited.
4. On the Edit JMS Destination Resource page, you may perform the following tasks:
   - Change the type of the resource.

- Modify the text in the Description field.
- Select or deselect the Enabled checkbox to enable or disable the resource.
- Add, remove, or modify the Name property.

5. Click Save to save your changes.

Deleting a JMS Destination Resource

1. In the tree component, expand the Java Message Service node.
2. Select the Destination Resources node.
3. On the JMS Destination Resources page, select the checkbox next to the name of the destination resource to be deleted.
4. Click Delete.

Equivalent `asadmin` command: `delete-jms-resource`

Admin Console Tasks for the JMS Provider

- Configuring General Properties for the JMS Provider
- Configuring the JMS Host

Configuring General Properties for the JMS Provider

1. In the tree component, select the Java Message Service node to open the JMS Service page.
2. Edit the value in the Initial Timeout field if you want to change the time the application server waits for the JMS service to start before aborting the startup. On a slow or overloaded system, you may need to increase the value from the default (60).
3. From the Type combo box:
   a. Choose LOCAL (the default) to access the JMS service on the local host.
   b. Choose NONE to disable the JMS service the next time the server starts.

c. Choose REMOTE to access the JMS service on another system. If you choose this value, follow the instructions in Configuring the JMS Host to specify the name of the remote host. Selecting this value also means that the JMS service will not start on the local system the next time the server starts.

4. In the Start Arguments field, type arguments to customize the JMS service startup. You can use any arguments that you can specify to the *install_dir*`/imq/bin/imqbrokerd` command.

5. In the Additional Properties area, click Add Property to add a property. You may specify Message Queue broker configuration properties here.

6. When you have finished, click Save to save your changes, or click Load Defaults to restore the default values for the service.

Changing the provider and host to a remote system causes all JMS applications to run on the remote server. If you want to be able to use both the local server and a remote server, create a connection factory resource with the `MessageServiceAddressList` property to create connections that access the remote server.

After you change properties of the JMS provider, you must stop and restart the server in order for the changes to take effect.

Configuring the JMS Host

1. In the tree component, expand the Java Message Service node.

2. Select the JMS Hosts node.

3. On the JMS Hosts page, select `default_JMS_host` to display the Edit JMS Host page.

4. In the Host field, type the name or Internet Protocol (IP) address of the host. If you changed the type of the JMS Service to REMOTE, type the name of the remote host.

5. In the Port field, type the port number of the JMS service. Change this field only if the JMS service you will use is running on a nondefault port. (The default port is 7676.)

6. In the Admin Username and Admin Password fields, type the MQ broker user name and password. These are different from the Application Server user name and password. Edit these fields only if you have used the *install_dir*`/imq/bin/imqusermgr` command to change the MQ broker values.

7. When you have finished, click Save to save your changes, or click Load Defaults to restore the default values for the host.

At this release, the Application Server expects there to be exactly one JMS host, named `default_JMS_host`. You could use the JMS Hosts page to create new hosts, but the Application Server would have no way of knowing about them. Deleting the default JMS host is not recommended.

After you change properties of the JMS host, you must stop and restart the server in order for the changes to take effect.

Chapter 5

JavaMail Resources

This chapter briefly describes how to configure resources for applications that use the JavaMail API. It contains the following sections:

- The JavaMail API
- Admin Console Tasks for JavaMail

About JavaMail

- The JavaMail API

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 and sending email. Service providers implement particular protocols.

The JavaMail API is implemented as a Java platform optional package and is also available as part of the J2EE platform.

The Sun Java™ System Application Server Platform Edition 8 includes the JavaMail API along with a JavaMail service provider that allows an application component to send email notifications over the Internet and to read email from IMAP and POP3 mail servers.

For more information about the JavaMail API, go to the JavaMail website (`http://java.sun.com/products/javamail/`).

Admin Console Tasks for JavaMail

- Creating a JavaMail Session
- Editing a JavaMail Session
- Deleting a JavaMail Session

Creating a JavaMail Session

1. In the tree component, select the Java Mail Sessions node.
2. On the Java Mail Sessions page, click New. The Create Java Mail Session page appears.
3. In the JNDI Name field, type the name of the session. For example:

   `mail/MySession`

   It is a recommended practice to use the naming subcontext prefix `mail/` for JavaMail resources.
4. In the Mail Host field, type the DNS 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.
5. In the Default User field, type the username 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.
6. In the Default Return Address field, type the email address of the default user, in the form *`username@host.domain`*.
7. Deselect the Enabled checkbox if you do not want to enable the mail session at this time.

8. In the Advanced area, change the field values only if you have reconfigured the Application Server's mail provider to use a nondefault store or transport protocol. By default, the Store Protocol is `imap`; the Store Protocol Class is `com.sun.mail.imap.IMAPStore`; the Transport Protocol is `smtp`; and the Transport Protocol Class is `com.sun.mail.smtp.SMTPTransport`.

   Select the Debug 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 will be generated and will be included in the system log file. See Configuring Log Levels for information about setting the log level.

9. In the Additional Properties area, click Add Property to add properties required by your application, such as a protocol-specific host or username property. The JavaMail API documentation lists the properties you might want to set (`http://java.sun.com/products/javamail/javadocs/index.html`).

10. Click OK to save the session.

Equivalent `asadmin` command: `create-javamail-resource`

Editing a JavaMail Session

1. In the tree component, select the Java Mail Sessions node.
2. On the Java Mail Sessions page, select the session to be edited.
3. On the Edit Java Mail Session page, you may do the following:
   - Modify the values in the Mail Host, Default User, Default Return Address, or Description field.
   - Select or deselect the Enabled checkbox to enable or disable the resource.
   - Modify the values of the Advanced fields.
   - Add, remove, or modify properties.
4. Click Save to save your changes, or click Load Defaults to restore the default values for a mail session.

Deleting a JavaMail Session

1. In the tree component, select the Java Mail Sessions node.

2. On the Java Mail Sessions page, select the checkbox next to the name of the session to be deleted.
3. Click Delete.

Equivalent `asadmin` command: `delete-javamail-resource`

Chapter 6

The Naming and Directory Service

This chapter briefly discusses the naming of resources that applications can locate with calls to the Java Naming and Directory Interface (JNDI) API. This chapter contains these sections:

- About JNDI
- Admin Console Tasks for Custom Resources
- Admin Console Tasks for External Resources

About JNDI

- JNDI Names and Resources
- JNDI Subcontexts

JNDI Names and Resources

JNDI is the acronym for the Java Naming and Directory Interface API. By making calls to this API, applications can locate resources and other program objects. A resource is a program object that provides connections to sytems 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: 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. When you create a new resource, a new name-object binding is entered into the JNDI repository.

JNDI Subcontexts

JNDI names are organized hierarchically, much like a file system. In a file system, you can organize your files into subdirectories. In JNDI, resource names are organized into subcontexts. For example, all JDBC resource names must be in `java:comp/env/jdbc` subcontext. The JNDI name for the resource of a payroll database could be `java:comp/env/jdbc/payrolldb`.

Table 6-1 lists the subcontexts for the various resources. In the table, the External System column lists the type of system that an application accesses through the connection provided by the resource.

Table 6-1 JNDI Subcontexts for Various Resources

JNDI Subcontext	Resource	External System
java:comp/env/jdbc	JDBC	database
java:comp/env/jms	JMS	messaging system
java:comp/env/mail	JavaMail	email
java:comp/env/*eis-specific*	connector	EIS (enterprise information system)

Admin Console Tasks for Custom Resources

- Creating a Custom Resource
- Editing a Custom Resource
- Deleting a Custom Resource

Creating a Custom Resource

The J2EE architecture includes standard resources, such as JDBC, JMS, and JavaMail resources. (For a full list of resources, see Table 6-1.) The flexibility of the architecture also allows developers to create custom resources. Because these resources are non-standard, the application developer must provide the administrator with the names of the JNDI subcontext, resource type, and factory class.

To create a custom resource:

1. In the tree component, expand the JNDI node.

2. Select the Custom Resource node.
3. On the Custom Resources page, click New.
4. On the Create Custom Resource page, specify the resource's settings:
    a. In the JNDI Name field, type a unique name in the format *subcontext/name*. The value of the subcontext depends on the specific resource.
    b. In the Resource Type field, type the name of the interface that the Factory Class entry implements.
    c. In the Factory Class field, type the name of the class that provides connections. This class must implement `javax.naming.spi.ObjectFactory`.
    d. By default, the resource is available (enabled) as soon as it is created. If you want the resource to be unavailable, deselect the Enabled checkbox.
    e. If the resource requires additional properties, click Add Properties.
5. Click OK.

Equivalent `asadmin` command: `create-custom-resource`

Editing a Custom Resource

1. In the tree component, expand the JNDI node.
2. Select the Custom Resources node.
3. On the Custom Resources page, click the resource you want to edit.
4. On the Edit Custom Resource page, you can perform these tasks:
    a. In the Resource Type field, type the name of the interface that the Factory Class entry implements.
    b. In the Factory Class field, type the name of the class that provides connections. This class must implement `javax.naming.spi.ObjectFactory`.
    c. Select or deselect the checkbox to enable or disable the resource.
    d. Add or delete resource properties.
5. Click Save to apply the edits you have made.

Deleting a Custom Resource

1. In the tree component, expand the JNDI node.
2. Expand the Custom Resources node.
3. On the Custom Resources page, select the checkbox for the resource that you want to delete.
4. Click Delete.
5. Restart the domain.

Equivalent `asadmin` command: `delete-custom-resource`

Admin Console Tasks for External Resources

- Creating an External Resource
- Editing an External Resource
- Deleting an External Resource

Creating an External Resource

The Application Server includes a naming and directory service that accesses a local JNDI repository. However, an application might need to locate a resource that is named in an external JNDI repository. For example, an application might need to locate an external resource object in a Lightweight Directory Access Protocol (LDAP) repository. The instructions that follow include example values for LDAP.

To create an external resource:

1. In the tree component, expand the JNDI node.
2. Select the External Resources node.
3. On the External Resources page, click New.
4. On the Create External Resource page, specify the resource's settings:
    a. In the JNDI Name field, type a unique name in the format *subcontext/name*. The value of the subcontext depends on the specific resource.

       Example: `test/myResource`.

b. In the JNDI Lookup field, type the name that the application uses to look up the resource.

   Example: `cn=myResource`.

c. In the Resource Type field, type the name of the interface that the Factory Class entry implements.

   Example: `test.myResource`

d. In the Factory Class field, type the name of the class that provides connections.

   Example: `com.sun.jndi.ldap.LdapCtxFactory`

e. By default, the resource is available (enabled) as soon as it is created. If you want the resource to be unavailable, deselect the Enabled checkbox.

f. If the resource requires additional properties, click Add Properties.

   For the addtional properties of an LDAP example, see Table 6-2.

Table 6-2 Additonal Properties for an LDAP External Resource

Name	Value
PROVIDER-URL	ldap://ldapserver:389/o=myObjects
SECURITY_AUTHENTICATION	simple
SECURITY_PRINCIPAL	cn=joeSmith, o=Engineering
SECURITY_CREDENTIALS	changeit

5. Click OK.

Editing an External Resource

1. In the tree component, expand the JNDI node.
2. Select the External Resources node.
3. On the External Resources page, click the resource you want to edit.
4. On the Edit External Resource page, you can perform these tasks:

   a. In the Resource Type field, type the name of the interface that the Factory Class entry implements.

b. In the JNDI Lookup field, type the name that the application uses to look up the resource.

c. In the Factory Class field, type the name of the class that provides connections. This class must implement `javax.naming.spi.ObjectFactory`.

d. Select or deselect the checkbox to enable or disable the resource.

e. Add or delete resource properties.

5. Click Save to apply the edits you have made.

Deleting an External Resource

1. In the tree component, expand the JNDI node.
2. Expand the External Resources node.
3. On the External Resources page, select the checkbox for the resource that you want to delete.
4. Click Delete.
5. Restart the domain.

Chapter 7

Connector Resources

This chapter explains how to configure connectors, which are used to access enterprise information systems (EISs). This chapter contains the following sections:

- About Connectors
- Admin Console Tasks for Connector Connection Pools
- Admin Console Tasks for Connector Resources
- Admin Console Tasks for Administered Object Resources

About Connectors

- Connector Modules, Connection Pools, and Resources

Connector Modules, Connection Pools, and Resources

Also called a resource adapter, a connector module is a J2EE 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. Like other J2EE modules, to install a connector module you deploy it.

A connector connection pool is a group of reusable connections for a particular EIS. When you create a connector connection pool, you 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. When you create a connector resource, you specify its JNDI name and the connection pool that it is associated with. Multiple connector resources may specify a single connection pool. The application locates the resource by looking up its JNDI name. (For more information on JNDI, see the section JNDI Names and Resources.) The JNDI name of a connector resource for an EIS is usually in the `java:comp/env/`*eis-specific* subcontext.

The Application Server implements JMS by using a connector module (resource adapter). See the section, The Relationship Between JMS Resources and Connector Resources.

For more information, including `asadmin` examples, see the link to G*etting Started With J2EE Connectors* in the TBD link Further Info section.

Admin Console Tasks for Connector Connection Pools

- General Steps for Setting Up EIS Access
- Creating a Connector Connection Pool
- Editing a Connector Connection Pool
- Deleting a Connector Connection Pool

General Steps for Setting Up EIS Access

1. Deploy (install) a connector. See Deploying a Connector Module.
2. Create a connection pool for the connector. See Creating a Connector Connection Pool.
3. Create a connector resource that is associated with the connection pool. See Creating a Connector Resource.

Creating a Connector Connection Pool

Before you create the pool, you must deploy the connector module (resource adapter) associated with the pool. The values that you specify for the new pool depend on the connector module that you have deployed.

To create a connector connection pool:

1. In the tree component, expand the Connectors node.
2. Select the Connector Connection Pools node.
3. On the Connector Connection Pools page, click New.
4. On the first Create Connector Connection Pool page, specify the following settings:
    a. In the Name field, enter a logical name for the pool.

       You will specify this name when you create a connector resource.

    b. Select an entry from the Resource Adapter combo box.

       The combo box displays a list of deployed resource adapters (connector modules).
5. Click Next.
6. On the second Create Connector Connection Pool page, select a value from the Connection Definition combo box.

   The choices in the combo box depend on the resource adapter. Typically, you'll specify a type of `ConnectionFactory`, a program object that produces connections.
7. Click Next.
8. On the third and last Create Connector Connection Pool page, perform these tasks:
    a. In the General Settings section verify that the values are correct.

    b. For the fields in the Pool Settings, section, you may retain the default values.

       You can change these settings at a later time. See Editing a Connector Connection Pool.

    c. In the Additonal Properties table, add any required properties.

       In the previous Create Connector Connection Pool page, you selected a class in the Connection Definition combo box. If this class is in the server's classpath, then the Additional Properties table displays default properties.
9. Click Finish.

Equivalent `asadmin` command: `create-connector-connection-pool`

-

Editing a Connector Connection Pool

The Edit Connector Connection Pool page enables you to change the pool settings and the addtional properties.

To access the Edit Connector Connection Pool page:

1. In the tree component, expand the Connectors node.
2. Expand the Connector Connection Pools node.
3. Select the node for the pool you want to edit.
4. On the Edit Connector Connection Pool page, you can change settings that control the number of connections in the pool. See Table 7-1.

Table 7-1 Pool Settings for a Connector Connection 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 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 timeout. 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 re-establish them. If you do not select the checkbox, then individual connections will be re-established only when they are used.

5. In the Additional Properties table, you can specify name-value pairs. The properties you specify depend on the resource adapter used by this pool.
6. Click Save.

Deleting a Connector Connection Pool

1. In the tree component, expand the Connectors node.
2. Select the Connector Connection Pools node.
3. On the Connector Connector Connection Pools page, select the checkbox for the pool that you want to delete.
4. Click Delete.

Equivalent `asadmin` command: `delete-connector-connection-pool`

Admin Console Tasks for Connector Resources

- Creating a Connector Resource
- Editing a Connector Resource
- Deleting a Connector Resource
- Creating an Administered Object Resource

Creating a Connector Resource

A connector resource (data source) provides applications with a connection to an EIS. Before creating a connector resource, you must first create a connector connection pool.

To create a connector resource:

1. In the tree component, expand the Connectors node.
2. Expand the Connector Resources node.
3. On the Connector Resources page, click New.
4. On the Create Connector Resources page, specify the resource’s settings:

a. In the JNDI Name field, type a uniqe name, for example: `eis/myERP`. Don't forget the forward slash.

b. From the Pool Name combo box, choose the connection pool that the new connector resource will belong to.

c. By default, the resource is available (enabled) as soon as it is created. If you want the resource to be unavailable, deselect the Enabled checkbox.

5. Click OK.

Equivalent `asadmin` command: `create-connector-resource`

Editing a Connector Resource

1. In the tree component, expand the Connectors node.
2. Expand the Connector Resources node.
3. Select the node for the connector resource that you want to edit.
4. On the Edit Connector Resources page, you can perform these tasks:
   a. From the Pool Name combo box, select a different connection pool.
   b. Select or deselect the checkbox to enable or disable the resource.
5. Click Save to apply the edits you have made.

Deleting a Connector Resource

1. In the tree component, expand the Connectors node.
2. Select the Connector Resources node.
3. On the Connector Resources page, select the checkbox for the resource that you want to delete.
4. Click Delete.

Equivalent `asadmin` command: `delete-connector-resource`

Admin Console Tasks for Administered Object Resources

- Creating an Administered Object Resource
- Editing an Administered Object Resource
- Deleting an Administered Object Resource

Creating an Administered Object Resource

Packaged within a resource adapter (connector module), an administered object provides specialized functionality for an application. For example, an administered object might provide access to a parser that is specific to the resource adapter and its associated EIS. The object can be administered; that is, it can be configured by an adminisitrator. To configure the object, you add name-value property pairs in the Create or Edit Admin Ojbect Resource pages. When you created an administered object resource, you associate the administered object with a JNDI name.

The Application Server implements JMS by using resource adapter. For each JMS destination you create, the Application Server automatically creates an administered object resource.

To create an administered object resource:

1. In the tree component, expand the Connectors node.
2. Expand the Admin Object Resources node.
3. On the Connector Resources page, click New.
4. On the Admin Object Resources page, specify the settings:
    a. In the JNDI Name field, type a uniqe name that will identify the resource.
    b. In the Resource Type field, enter the Java type for the resource.
    c. From the Resource Adapter combo box, select the resource adapter that contains the administered object.
    d. Select or deselect the checkbox to enable or disable the resource.
5. To configure the administered object with name-value property pairs, click Add Property.
6. Click OK.

Equivalent `asadmin` command: `create-admin-object`

Editing an Administered Object Resource

1. In the tree component, expand the Connectors node.
2. Expand the Administered Object Resources node.
3. Select the node for the administered object resource that you want to edit.
4. On the Edit Administered Object Resources page, you can modify values you specified in Creating an Administered Object Resource.
5. Click Save to apply the edits you have made.

Deleting an Administered Object Resource

1. In the tree component, expand the Connectors node.
2. Select the Administered Object Resources node.
3. On the Administered Object Resources page, select the checkbox for the resource that you want to delete.
4. Click Delete.

Equivalent `asadmin` command: `delete-admin-object`

Chapter 8

J2EE Containers

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

- About the J2EE Containers
- Admin Console Tasks for the J2EE Containers

About the J2EE Containers

- Types of J2EE Containers
- The Web Container
- The EJB Container

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 Sun Java™ System Application Server Platform Edition 8 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.

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 Java Server Pages (JSPs).

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.

Admin Console Tasks for the J2EE Containers

- Configuring the Web Container
- Configuring the General EJB Settings
- Configuring the Message-Driven Bean Settings
- Configuring the EJB Timer Service Settings

Configuring the Web Container

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

Configuring the General EJB Settings

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

- Session Store Location
- Pool Settings

- Cache Settings

You can override the defaults on a per-container basis by adjusting the values in the enterprise bean’s `sun-ejb-jar.xml` file. For details, see the *Sun Java™ System Application Server Platform Edition 8 Developer’s Guide*. (For a link to the guide, see Further Information.)

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.

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.

To adjust the configuration of the container’s pool of enterprise beans:

1. In the tree component expand the J2EE Containers node.
2. Expand the EJB Container node.
3. Under Pool Settings in the Initial and Minimum Pool Size field enter the minimum number of beans the container will create in the pool.
4. In the Maximum Pool Size field enter the maximum number of beans the container will maintain in the pool at any time.
5. In the Pool Resize Quantity field enter the number of beans that will be removed from the pool if they are idle for more than the time specified in Pool Idle Timeout.
6. In the Pool Idle Timeout field enter the time, in seconds, that a bean in the pool can remain idle before it will be removed from the pool.

7. Restart the Application Server.

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.

To adjust the settings for cached enterprise beans:

1. In the tree component expand the J2EE Containers node.
2. Expand the EJB Container node.
3. Adjust the maximum cache size in the Max Cache Size field.

   Increase the maximum number of beans to cache to eliminate the overhead of bean creation and destruction. However, if you increase the cache, the server will consume more memory and resources. Be sure your operating environment is sufficient for your cache settings.

4. Adjust the cache resize quantity in the Cache Resize Quantity field.

   When the maximum number of cached beans is reached, the container will remove a number of passivated beans from the backup store, set to 32 by default.

5. Adjust the rate, in seconds, at which the cache cleanup is scheduled for entity beans in the Cache Idle Timeout field.

   If a cached entity bean has been idle a certain amount of time, it will be passivated. That is, the bean's state will be written to a backup store.

6. Adjust the time, in seconds, after which stateful session beans will be removed from the cache or passivated store in the Removal Timeout field.
7. Configure the policy the container uses to remove stateful session beans in the Removal Selection Policy field.

   The container decides which stateful session beans to remove based on the policy set in the Removal Selection Policy field. There are three possible policies the container uses to remove beans from the cache:

- Not recently used (NRU)
- First in, first out (FIFO)
- Least recently used (LRU)

The NRU policy removes a bean that hasn't been used recently. The FIFO policy removes the oldest bean in the cache. The LRU policy removes the least recently accessed bean. By default, the NRU policy is used by the container.

Entity beans are always removed using the FIFO policy.

8. Restart the Application Server.

Configuring the Message-Driven Bean Settings

The pool for message-driven beans is similar to the pool for session beans described in "Configuring the General EJB Settings."

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

To adjust the configuration of this pool:

1. In the tree component expand the J2EE Containers node.
2. Expand the EJB Container node.
3. Click the MDB Settings tab.
4. Under Pool Settings in the Initial and Minimum Pool Size field enter the minimum number of message beans the container will create in the pool.
5. In the Maximum Pool Size field enter the maximum number of beans the container will maintain in the pool at any time.
6. In the Pool Resize Quantity field enter the number of beans that will be removed from the pool if they are idle for more than the time specified in Pool Idle Timeout.
7. In the Pool Idle Timeout field enter the time, in seconds, that a bean in the pool can remain idle before it will be removed from the pool.
8. Restart the Application Server.

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.

Configuring the Timer Service

1. In the tree component expand the J2EE Containers node.
2. Expand the EJB Container node.
3. Click the EJB Timer Service tab.
4. Set the minimum delivery interval in milliseconds in the Minimum Delivery Interval field. Minimum Delivery Interval is the minimum number of milliseconds allowed before the next timer expiration for a particular timer can occur. Setting this interval too low can cause the server to be overloaded.
5. Set the maximum number of attempts the timer service will make to deliver the notification in the Maximum Redeliveries field.
6. Set the interval, in milliseconds, between redelivery attempts in the Redelivery Interval field.
7. Restart the Application Server.

Using an External Database with the Timer Service

The timer service by default uses an embedded database to store timers.

To use an external database to store timers:

1. Set up a JDBC resource for the database, as described in Creating a JDBC Resource.
2. Enter the JNDI name of the resource in the Timer Datasource field.
3. Restart the Application Server.

Sample timer database creation files are provided for PointBase or Oracle at `<`*`INSTALL_DIR`*`>/lib/install/databases/`.

Chapter 9

Security

This chapter describes some core application server security concepts, and describes how to configure security for the Sun Java™ System Application Server Platform Edition 8. This chapter contains the following topics:

- About Application Server Security
- Admin Console Tasks for Security
- Admin Console Tasks for Realms
- Admin Console Tasks for JACC Providers
- Admin Console Tasks for Audit Modules
- Admin Console Tasks for Listeners
- Security Tasks for Connector Connection Pools
- Working with Certificates and SSL

About Application Server Security

- Overview of Security
- Authentication and Authorization
- Users, Groups, Roles, and Realms
- Introduction to Certificates and SSL
- Firewalls
- Security Managment With the Admin Console

Overview of 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.

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 a administrator, you don't have any control over this mechanism. Generally, programmatic security is discouraged since it hard-codes security configuration 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 applications developer, so this document will not discuss it; declarative security is somewhat less so, and this document will touch on it occasionally. This document is intended for system administrators, and so will focus on system security.

Tools for Managing Security

The Application Server provides two tools for managing security:

- Admin Console, a browser-based tool you use to configure security for the entire server, to manage users, groups, and realms, and perform other system-wide security tasks. For a general introduction to Admin Console, see Tools for Administration. For an overview of the security tasks you can perform with Admin Console, see Security Managment With the Admin Console.

- asadmin, a command-line tool that performs many of the same tasks as the Admin Console. You may be able to do some things with asadmin that you cannot do with Admin Console. You perform asadmin commands either from a command prompt or from a scripts, to automate repetitive tasks. For a general introduction to asadmin, see Tools for Administration
- deploytool, a graphical packaging and deployment tool for editing application deployment descriptors to control individual applications' security. Because deploytool is intended for application developers, this document will 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.

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

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

For more information on using keytool, policytool, and other Java security tools, see *Java2 SDK Tools and Utilities* at http://java.sun.com/j2se/1.4.2/docs/tooldocs/tools.html#security.

Security Responsibilities

Security responsibilities are assigned to the following:

- Application developer
- Application deployer
- System administrator

Application Developer

The application developer is responsible for the following:

- Specifying roles and role-based access restrictions for application components..
- Defining an application's authentication method and specifying the parts of the application that are secured.

An application developer may use tools such as deploytool to edit application deployment descriptors.

Application Deployer

The application deployer is responsible for:

- Mapping users or groups (or both) to security roles.
- Refining the privileges required to access component methods to suit the requirements of the specific deployment scenario.

An application deployer may use tools such as `deploytool` to edit application deployment descriptors.

System Administrator

The system administrator is responsible for:

- Configuring security realms.
- Managing user accounts and groups.
- Managing audit logs.
- Managing server certificates and configuring the server's use of secure sockets layer (SSL).
- Handling other miscellaneous system-wide security features, such as security maps for connector connection pools**,** additional JACC Providers, and so on.

A system administrator may use the Admin Console to manage server security settings, and `keytool` to manage certificates.

Authentication and Authorization

Authentication and authorization are central concepts of application server security.

Authentication

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 username 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 may 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 may 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 Table 9-1. An application specifies the type of authentication it uses in its deployment descriptors. Use `deploytool` to configure the authentication method for an application. For more information, see Tools for Managing Security.

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
Digest	HTTP	Uses server’s built-in pop-up login dialog box	Passwords MD5 encrypted.

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. For information on disabling single sign-on, see Configuring Single Sign-On (SSO).

Authorization

Once a user is authenticated, his or her level of *authorization* determines what operations he can perform. 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 only view their own personal information. For more on roles, see Users, Groups, Roles, and Realms.

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 you 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 JAAC specification. In addition, you may specify additional third-party JAAC 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.

Audit Logging

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, and you can also configure your own custom audit modules. For information on developing custom audit modules, see the *J2EE 1.4 Application Server Developer's Guide*.

Users, Groups, Roles, and Realms

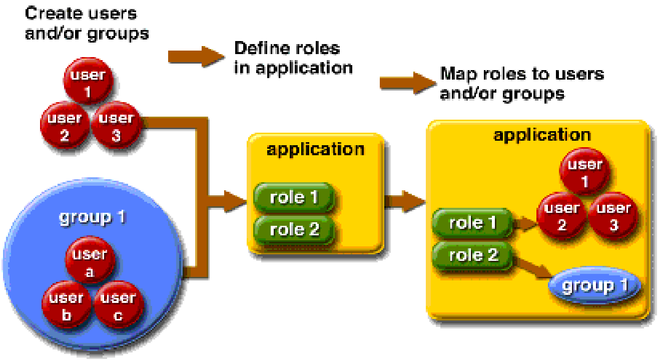
The Application Server enforces its authentication and authorization policies upon the following entities:

- User: An individual identity defined in the Application Server. In general, a user may be a person or a software component such as an EJB, or even a service. A user who has been authenticated is sometimes called a Principal. Users are sometimes referred to as subjects.
- Group: A set of users defined in the Application Server, classified by common traits.
- Role: 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 you are, only that you have the right key.

Note: Users and groups are designated for the entire Application Server, whereas each application defines its own roles. Then, the applications specify mappings between users / groups and roles, as illustrated in Figure 9-1.

A realm is a repository containing user and group information and their security credentials. A realm is also called a security policy domain.

Figure 9-1 Role Mapping



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' and 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 could define at least two roles: employee and manager; only users in the manager role could 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 pre-configured with two realms: file (the initial default realm) and certificate. You can set up two other realms: ldap and solaris, in addition to custom realms. Applications can specify the realm to use in their deployment descriptor. If they do not specify a realm, the Application Server will use its default realm.

In the file realm, the server stores user credentials locally in a file. You can use the Admin Console to manage users in the file realm. For more information, see Managing file Realm Users

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.

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 versions. 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 Creating a Custom Realm.

Introduction to Certificates and SSL

- Digital Certificates
- Secure Sockets Layer

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). For more information on SSL, see Secure Sockets Layer.

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 a 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 to prove their identity and 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 e-mail 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.

Certificate Chains

Web browsers are pre-configured 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 CAs, 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 and 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 may return a certificate 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 you sent the CSR to. Then, the next certificate in the chain will be 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.

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). HTTPS 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 has been 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 then, on the client and the server use the session key to encrypt all their communications with each other. Thus, their communications is 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 in Generating a Server Certificate to set up a digital certificate that your web server can use for SSL.

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 may support different cipher suites. You can 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 should 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 will display a warning. In general, only address-based virtual hosts are commonly used with SSL in a production environment.

Firewalls

A *firewall* controls the flow of data between two or more networks, and manages the links between the networks. A firewall may 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, you must configure you firewalls so that clients can access the necessary TCP/IP ports. For example, if the HTTP listener is operating on port 8080, you must, you must configure the firewall to allow HTTP requests on port 8080. Likewise, if HTTPS requests are setup for port 1043, you must configure your firewalls to allow HTTPS requests on port 1043.

If you want to allow direct RMI/IIOP access from the Internet to EJB modules, you must open the RMI/IIOP listener port as well, but this is strongly discouraged since it creates security risks.

In the 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.

Security Managment With the Admin Console

The Admin Console enables you to manage:

- Server Security Settings
- Realms and file Realm Users
- JACC Providers
- Audit Modules
- HTTP and IIOP Listener Security

Server Security Settings

On the Security Settings page, you can set properties for the server as a whole, such as the default realm, the anonymous role, the default principal user name and password. For more information, see Configuring Security Settings.

Realms and file Realm Users

The concept of realms was introduced in Users, Groups, Roles, and Realms. With the Admin Console, you can:

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

See Admin Console Tasks for Realms for details on these tasks.

JACC Providers

JACC providers were introduced in JACC Providers. With the Admin Console, you can:

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

See Admin Console Tasks for JACC Providers for details on these tasks.

Audit Modules

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.

With the Admin Console, you can:

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

See Admin Console Tasks for Audit Modules 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*. For general information about the HTTP service and HTTP listeners, see What Is the HTTP Service?

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 EJBs and from other CORBA-based clients. For general information on IIOP listeners, see IIOP Listeners.

With the Admin Console you can:

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

See Admin Console Tasks for Listeners for details on these tasks.

Admin Console Tasks for Security

- Configuring Security Settings
- Controlling Access to Administration Tools
- Configuring Mutual Authentication
- Configuring Single Sign-On (SSO)

Configuring Security Settings

The Security page in the Admin Console enables you to set a variety of system-wide security settings. To edit these settings:

1. In the tree component, select the Security node in the Admin Console tree.

   The Security page displays.

2. Modify the values as desired. Table 9-2 describes the settings on this page.
3. Enter additional properties to pass to the Java Virtual Machine (JVM) in the Additional Properties section.
4. Valid properties are dependent upon the type of realm being configured. Select Save to save the changes or Load Defaults to restore the default values.

Table 9-2 General Security Settings

Setting	Description
Audit Logging	Whether audit logging is enabled. If enabled, the server will load and run all the audit modules specified in the Audit Modules setting. Otherwise, the server does not access audit modules. Disabled by default.
Default Realm	The active (default) realm the server uses for authentication. Applications will use this realm unless they specify a different realm in their deployment descriptor. All configured realms appear in the list. The initial default realm is the file realm.
Anonymous Role	The name for the default or anonymous role. The anonymous role is assigned to all users. Applications can use this role in their deployment descriptors to grant authorization to anyone.
Default Principal	Specifies the default user name. The server uses this when no principal is provided. If you enter a value in this field, you must enter a corresponding value in the Default Principal Password field. You do not need to set this attribute for normal server operation.
Default Principal Password	Password of the default principal specified in the Default Principal field. You do not need to set this attribute for normal server operation.

Table 9-2 General Security Settings

Setting	Description
JACC	Class name of a configured JACC provider. See Creating a JACC Provider for Information on adding JACC providers.
Audit Modules	List of audit module provider classes, delimited by commas. A module listed here must be already configured. If Audit Logging is enabled, this setting must list audit modules. By default, the server uses an audit module named `default`. For information on creating new audit modules, see Creating an Audit Module.

Controlling Access to Administration Tools

Only users in the `asadmin` group are able to access Admin Console and `asadmin` command line utility. To give a user access to these administration tools, add them to the `asadmin` group, as follows:

1. Expand the Security node in the Admin Console tree.
2. Expand the Realms node.
3. Select the file node.
4. Click the Manage Users button from the Edit Realm page.

   Initially after installation, only the administrator user name and password entered during installation will be listed. By default, this user belongs to the group `asadmin`, which gives rights to modify the Application Server. Assign users to this group only if you want to grant them administrator privileges for the Application Server.

5. Click New to add a new user to the file realm.
6. Enter the correct information into the User Id, Password, and Group List fields. To authorize a user to make modifications to the Application Server, include the asadmin group in the Group List.
7. Click OK to add this user to the file realm or click Cancel to quit without saving.

Configuring Mutual Authentication

- Enabling Mutual Authentication for the Certificate Realm
- Enabling Mutual SSL Authentication an Application

In mutual authentication, both server and client-side authentication are enabled. To test mutual authentication, you must have a client with a valid certificate. For information on creating a client certificate, see *The J2EE 1.4 Tutorial*.

Enabling Mutual Authentication for the Certificate Realm

The Application Server uses the certificate realm for HTTPS authentication. To specify mutual authentication for all the applications that use this realm, follow this procedure:

1. In the Admin Console tree, expand Security, then expand Realms, and select certificate.
2. Click the Add Property button.
   - In the Name field, enter `clientAuth`.
   - In the Value field, enter `true`.
3. Click Save.
4. Restart the Application Server.

   After you restart the server, it will require client authentication for all applications that use the certificate realm.

Enabling Mutual SSL Authentication an Application

If you to enable mutual authentication for a specific application, use `deploytool` to set the method of authentication to `Client-Certificate`. For more information about using `deploytool`, refer to the Security chapter of the J2EE Tutorial.

Configuring Single Sign-On (SSO)

Single sign-on enables multiple applications to share user sign-on information, rather than requiring each application to have separate user sign-on. Applications using single sign-on authenticate the user one time, and the authentication information is propagated to all other involved applications.

Single sign-on applies to web applications configured for the same realm and virtual server.

Note: Single sign-on uses an HTTP cookie to transmit a token that associates each request with the saved user identity, so it can only be used when the browser client supports cookies.

Single sign-on operates according to the following rules:

- When a user accesses a protected resource in a web application, the server requires the user to authenticate himself or herself, using the method defined for that web application.
- Once authenticated, the Application Server will use the roles associated with the user for authorization decisions across all web applications on the virtual server, without challenging the user to authenticate to each application individually.
- When the user logs out of one web application (explicitly, or because of session expiration), the user's sessions in all web applications become invalid. Thereafter, the user is required to login to access a protected resource in any application.

Single sign-on is enabled by default for the Application Server. To disable it or configure other properties, follow this procedure:

1. In the tree component, expand the HTTP service node.
2. Open the Virtual Servers node, and select the virtual server for which you want to disable single sign-on support.
3. Click Add Property.

   A blank property entry is added to the bottom of the list.
4. Enter `sso-enable` in the new Name field.
5. Enter `false` in the new Property field.
6. Add or change any other single sign-on properties:

Property Name	**Description**	**Values**
sso-max-inactive-seconds	Number of seconds after which a user's single sign-on record becomes eligible for purging if no client activity is received. Access to any of the applications on the virtual server keeps the single sign-on record active.	Default is 300 seconds (5 minutes). A higher value provides longer persistence for users, but consumes more memory on server.
sso-reap-interval-seconds	Interval (in seconds) between purges of expired single sign-on records.	Default is 60

7. Click Save.
8. Restart the Application Server.

Admin Console Tasks for Realms

- Creating a Realm
  - Creating the ldap Realm
  - Creating the solaris Realm
  - Creating a Custom Realm
- Editing a Realm
  - Editing the file Realm
  - Managing file Realm Users
  - Editing the certificate Realm
- Deleting a Realm
- Setting the Default Realm

Creating a Realm

The Application Server comes preconfigured with two realms: file and certificate. You can create two other realms: ldap and solaris, in addition to custom realms. Generally, you will have one realm of each type on a server. However, you can have a different certificate database for each virtual server on your system.

To create a security realm, follow these steps:

1. In the tree component, expand the Security node.
2. Expand the Realms node.
3. On the Realms page, click New.

   The Create Realm page is displayed.
4. Enter a name for the realm in the Name field: file, certificate, ldap, solaris, or the name of a custom realm you are defining.

Specify the class name for the realm you want to create, as shown in the following table:

Realm Name	Class Name
file	`com.sun.enterprise.security.auth.realm.file.FileRealm`
certificate	`com.sun.enterprise.security.auth.realm.certificate.CertificateRealm`

Realm Name	Class Name
ldap	`com.sun.enterprise.security.auth.realm.ldap.LDAPRealm`
solaris	`com.sun.enterprise.security.auth.realm.solaris.SolarisRealm`
custom	Name of LoginModule class.

5. Add the required properties and any desired optional properties for the realm.
   - For a description of ldap realm properties, see Creating the ldap Realm.
   - For a description of solaris realm properties, see Creating the solaris Realm.
   - For a description of custom realm properties, see Creating a Custom Realm.

   To add a property:

   a. Click Add Property.

   a. In the Name field, enter the name of the property.

   b. In the Value field, enter the value of the property.
6. Click OK.

Equivalent `asadmin` command: `create-auth-realm`

Creating the ldap Realm

The ldap realm performs authentication using information from an LDAP server. User information may include user name, password, and the groups to which the user belongs. You must create the desired users and groups in your LDAP directory, or the information must already exist.

You must add the three properties shown in Table 9-3.

Table 9-3 Required properties for ldap realm

Property Name	Description	Value
directory	LDAP URL of the directory server.	LDAP URL of the form `ldap://`*`hostname`*`:`*`port`* For example, `ldap://myldap.foo.com:389`.
base-dn	Base DN for the location of user data, which can be at any level above the user data, since a tree scope search is performed. The smaller the search tree, the better the performance.	Domain for the search, for example: `dc=siliconvalley, dc=BayArea, dc=sun, dc=com`.

Table 9-3 Required properties for ldap realm

Property Name	Description	Value
jaas-context	Type of login module to use for this realm.	Must be `ldapRealm`.

Additionally, you may add any of the optional properties listed in Table 9-4.

Table 9-4 Optional properties for ldap realm

Property Name	Description	Default
search-filter	Search filter to use to find the user.	`uid=%s` (`%s` expands to the subject name)
group-base-dn	Base DN for the location of groups data.	Same as the `base-dn`, but it can be tuned if necessary
group-search-filter	Search filter to find group memberships for the user.	`uniquemember=%d` (`%d` expands to the user element DN)
group-target	LDAP attribute name that contains group name entries.	CN
search-bind-dn	Optional DN used to authenticate to the directory for performing the search-filter lookup. Only required for directories that do not allow anonymous search.	
search-bind-password	LDAP password for the DN given in `search-bind-dn`.	

Example

For example, suppose an LDAP user, Joe Java, is defined in your LDAP directory as follows:

```
uid=jjava,ou=People,dc=acme,dc=com
uid=jjava
givenName=joe
objectClass=top
objectClass=person
objectClass=organizationalPerson
objectClass=inetorgperson
sn=java
cn=Joe Java
```

The required properties in the ldap realm would be:

Property Name	Property Value
directory	LDAP URL to your server, for example: ldap://ldap.acme.com:389
base-dn	ou=People,dc=acme,dc=com. Could be rooted higher, for example dc=acme,dc=com, but searches would traverse a larger part of the tree, reducing performance.
jaas-context	ldapRealm

Creating the solaris Realm

The solaris realm gets user and group information from the underlying Solaris user database, as determined by the system's configuration. The Solaris realm invokes the underlying PAM infrastructure for authenticating. If the configured PAM modules require root privileges, the domain must run as root to use this realm. For details, see your Solaris documentation for security services.

The solaris realm has one required property, `jaas-context` that specifies the type of login module to use. The property value must be `solarisRealm`.

Note: The solaris realm is supported only for Solaris 9 or later.

Creating a Custom Realm

In addition to the four built-in realms, you can also create custom realms that store user data in some other way, such as in a relational database. Development of a custom realm is outside the scope of this document; for more information, see the *Sun Java System Application Server Platform Edition 8 Developer's Guide*. See http://developers.sun.com/prodtech/appserver/reference/techart/as8_authentication/index.html for an example implementation of a custom realm

As an administrator, the main thing you need to know is that a custom realm is implemented by a class derived from the Java Authentication and Authorization Service (JAAS) package: this class is called the *LoginModule*.

To configure the Application Server to use a custom realm:

1. Follow the procedure outline in Creating a Realm, entering the name of your custom realm and the name of the LoginModule class. You can use any unique name for your custom realm, for example `myCustomRealm`.

2. Add the following properties:

Property Name	Property Value
jaas-context	LoginModule class name, for example simpleCustomRealm
auth-type	Description of the realm, for example “A simple example custom realm”.

3. Click OK.

4. Edit the domain's login configuration file *install_dir*/`domains`/*domain_name*/`config/login.conf`, and add the fully-qualified class name of the JAAS LoginModule at the end of the file, as follows:

```
realmName {
    fully-qualified-LoginModule-classname required;
};
```

For example,

```
nyCustomRealm {
    com.foo.bar.security.customrealm.SimpleCustomRealm required;
};
```

5. Copy the LoginModule class and all dependent classes into the directory *install_dir*/`domains`/*domain_name*/`lib/classes`.

6. Restart the Server.

7. Make sure that the realm is properly loaded.

   Check *install_dir*/`domains`/*domain_name*/`logs/server.log` to make sure the server loaded the realm: The server should invoke the realm’s `init()` method.

Editing a Realm

1. In the tree component, expand the Security node.

2. Expand the Realms node.

3. Select the name of an existing realm.

   The Edit Realm page displays.

4. Edit existing properties and their values as desired.

   For information on file realm properties, see Editing the file Realm. To manage users in the file realm, click the Manager Users button; see Managing file Realm Users for more information.

   For information on certificate realm properties, see Editing the certificate Realm.

5. To add additional properties, click the Add Properties button. The page displays a new row. Enter a valid property name and property value. Table 9-6 describes optional properties for the certificate realm. Table 9-3 and Table 9-4 describe optional properties for the ldap realm.

6. Click Save to save the changes or Reload Defaults to discard your changes and restore the Application Server default values.

Editing the file Realm

The server maintains all user, group, and password information in a file called a *keyfile*. The file property specifies the location of the keyfile.

Table 9-5 Required properties for the file realm

Property name	**Description**	**Default Value**
file	Full path and name of the keyfile.	*`install_dir`*`/domains/`*`domain-name/`*`config/keyfile`
jaas-context	Type of login module to use for this realm.	`fileRealm` is the only valid value

Except for a single admin user in the asadmin group, the keyfile is initially empty, so you must add users before you can use the file realm. For instructions, see Managing file Realm Users.

Note: Users in the group asadmin are authorized to use the Admin Console and asadmin tools. Add only users to this group that you want to have server administrative privileges.

Managing file Realm Users

You can manage file realm users with the Admin Console. Users and groups in the file realm are listed in the key file, whose location is specified by the file property.

A user in the file realm can belong to a *J2EE group*, a category of users classified by common traits. For example, customers 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.

Initially after installation of the Application Server, the only user is the administrator entered during installation. By default, this user belongs to the group asadmin, which gives rights to modify the Application Server. Any users you assign to this group will have administrator privileges, that is, they will have access to `asadmin` and Admin Console.

To manage file realm users, follow this procedure:

1. Expand the Security node in the Admin Console tree.
2. Expand the Realms node.
3. Select the `file` node.
4. Click the Manage Users button from the Edit Realm page.

   The File Users page displays. In this page, you can perform the following tasks:

   - Add a user.
   - Edit an existing user.
   - Delete a user.

   Follow the corresponding procedure in the following sections.

Adding a User

In the File Users page, to add a new user:

1. Click New to add a new user to the file realm.
2. Enter the following information on the File Users page:
   - **User Id** *(required)* **-** The name of the user.
   - **Password** *(required)* **-** The user’s password.
   - **Confirm Password** *(required)* **-** The user’s password again, for verification.
   - **Group List** *(optional)* **-** a comma-separated list of the groups to which the user belongs. These groups do not need to be defined elsewhere.
3. Click OK to add this user to the list of users in the file realm. Click Cancel to quit without saving.

Equivalent `asadmin` command: `create-file-user`

Editing a User

In the File Users page, to change a user's information:

1. In the User Id column, click the name of the user you want to modify.
2. Change the user's password by entering a new password in the Password and Confirm Password fields.
3. Change the groups to which the user belongs by adding or deleting groups in the Group List field. Separate group names with commas. Groups need not be previously defined.
4. Click Save to save this user to the list of users in the `file` realm. Click Cancel to quit without saving.

Deleting a User

In the File Users page, to delete a user:

1. Select the checkbox to the left of the name of the user(s) you want to delete.
2. Click Delete.
3. Click Close to return to the Edit Realm page.

Equivalent `asadmin` command: `delete-file-user`

Editing the certificate Realm

The certificate realm supports SSL authentication. This realm sets up the user identity in the Application Server's security context, and populates it with user data obtained from cryptographically verified client certificates in the trust-store and keystore files (see Certificate Files.) You an add users to these files using `keytool`; for more information see the J2EE 1.4 Tutorial chapter titled *Security*.

With the certificate realm, J2EE containers handle authorization processing based on each user's Distinguished Name (DN) from his or her certificate. The DN is the name of the entity whose public key the certificate identifies. This name uses the X.500 standard, so it is intended to be unique across the Internet. For more information on keystores and trust-stores, refer to the keytool documentation at:

```
http://java.sun.com/j2se/1.4.2/docs/tooldocs/solaris/keytool.html
```

Table 9-6 lists the optional properties for the certificate realm.

Table 9-6 Optional properties for certificate realm

Property	Description
assign-groups	A comma-separated list of group names. All clients who present valid certificates are assigned to these groups. For example, `employee,manager`, where these are the names of user groups.
jaas-context	Type of login module to use for this realm. For the certificate realm, the value must be `certificateRealm`.

Deleting a Realm

1. Expand the Security node of the Application Server tree.
2. Select the Realms node.
3. Click in the box beside the realm to be deleted.
4. Click Delete.
5. Click Save to save the changes or Reload Defaults to negate your changes and restore the Application Server default values.

Equivalent `asadmin` command:`delete-auth-realm`

Setting the Default Realm

The *default realm* is the realm that the Application Server will use for authentication and authorization if an application's deployment descriptor does not specify a realm. To set the default realm:

1. In the tree component, select the Security node

   The Security page displays.
2. In the Default Realm field, pick the desired realm from the drop-down list.
3. Click Save to save the changes or Reload Defaults to negate your changes and restore the Application Server default values.
4. Restart the server.

Admin Console Tasks for JACC Providers

- Creating a JACC Provider
- Editing a JACC Provider
- Deleting a JACC Provider
- Setting the Active JACC Provider

Creating a JACC Provider

JACC (Java Authorization Contract for Containers) is part of the J2EE 1.4 specification that defines an interface for pluggable authorization providers. This enables you to set up third-party “plug in” modules to perform authorization. By default, the Application Server provides a simple, JACC-compliant file-based authorization engine. For more information, see JACC Providers.

To create a JACC provider:

1. In the tree component, expand the Security node.
2. Select the JACC Providers node.
3. On the JACC Providers page, click New.
4. On the Create JACC Provider page, enter the following:
   - **Name** – The application name as displayed in the deployed application list.
   - **Policy Configuration** – The name of the class that implements the policy configuration factory.
   - **Policy Provider** – The name of the class that implements the policy factory.
5. Add additional properties to the provider by clicking the Add Property button. Valid properties include:
   - repository: the directory that contains the policy file. For the default provider, this value is *install_dir*/*domain_dir*/`generated/policy`.
6. Click OK to save this configuration, or click Cancel to quit without saving.

Editing a JACC Provider

1. In the tree component, expand the Security node.
2. Expand the JACC Providers node.
3. Select the node of the JACC provider that you want to edit.
4. On the Edit JACC Provider page, modify the provider information as desired:
   - **Policy Configuration** – The name of the class that implements the policy configuration factory.
   - **Policy Provider** – The name of the class that implements the policy factory.
5. To add additional properties, click the Add button. Enter the name and value for the property. Valid entries include:
   - repository: the directory that contains the policy file. For the default provider, this value is *install_dir*/*domain_dir*/`generated/policy`.
6. To delete an existing property, click in the checkbox to the left of the property, then click Delete Properties.
7. Click Save.

Deleting a JACC Provider

1. In the tree component, expand the Security node.
2. Select the JACC Providers node.
3. Click in the checkbox to the left of the JACC provider you wish to delete.
4. Click Delete.

Setting the Active JACC Provider

To specify the JACC provider that the server uses:

1. In the tree component, select the Security node

   The Security page displays.
2. In the JACC field, enter the name of the JACC provider that you want the server to use. The tree component lists all the configured JACC providers.

3. Select Save to save the changes.
4. Restart the Application Server.

Admin Console Tasks for Audit Modules

- Creating an Audit Module
- Editing an Audit Module
- Deleting an Audit Module
- Setting the Active Audit Module
- Enabling and Disabling Audit Logging

Creating an Audit Module

The Application Server provides a simple default audit module; for more information, see Using the Default Audit Module.

To create a new audit module:

1. In the tree component, expand the Security node.
2. Select the Audit Modules node.
3. On the Audit Modules page, click New.
4. On the Create Audit Module page, enter the following information:
   - **Name** – The name of the new audit module.
   - **Classname** – The fully-qualified name of the class that implements this module.
5. To add additional JVM properties to this module, click Add Property. Specify a name and value for each property.
6. Click OK to save your entries, or click Cancel to quit without saving.

Editing an Audit Module

To edit an audit module:

1. In the tree component, expand the Security node.
2. Expand the Audit Modules node.
3. Click the node of the audit module that you want to edit.
4. On the Edit Audit Module page, modify the class name if desired.
5. Enter any additional properties for the module by selecting the Add button and entering the name and value of the property. Valid values include:
   - auditOn: specifies whether or not to use this audit module. Possible values are `true` and `false`.
6. Modify any existing properties by selecting the name or value to be modified, and entering the changes directly into the text field.
7. Delete a property by selecting the checkbox to the left of the property and clicking Delete Properties.
8. Click Save.

Deleting an Audit Module

To delete an audit module:

1. In the tree component, expand the Security node.
2. Select the Audit Modules node.
3. Click in the checkbox to the left of the audit module you want to delete.
4. Click Delete.

Setting the Active Audit Module

To specify the audit module that the server uses:

1. In the tree component, select the Security node

   The Security page displays.
2. In the Audit Modules field, enter the name of the audit module that you want the server to use. (The built-in audit module is called `default`.)
3. Select Save to save the changes.
4. Restart the Application Server.

Using the Default Audit Module

The default audit module logs authentication and authorization requests to the server log file. For information on changing the location of the log file, see Configuring General Logging Settings.

Authentication log entries include the following information:

- Names of users who attempted to authenticate.
- The realm that processed the access request.
- The requested web module URI or EJB component.
- Success or failure of the request.

Regardless of whether audit logging is enabled, the Application Server logs all denied authentication events.

Authorization log entries include the following information:

- Names of authenticated users, if any.
- The requested web URI or EJB component.
- Success or failure of the requests.

Enabling and Disabling Audit Logging

To specify the audit module that the server uses:

1. In the tree component, select the Security node.

   The Security page displays.

2. To enable logging, select the Audit Logging check box. To disable it, deselect it.
3. Select Save to save the changes.
4. Restart the Application Server.

Enabling and Disabling the Default Audit Module

In addition to enabling logging, you must set any properties required by the specific audit module you want to use. In the case of the default audit module, follow this procedure:

1. In the tree component, select the Security node.
2. Expand the Audit Modules node.

3. Click the default node.
4. Set the value of the auditOn property to `true`.
5. Select Save to save the changes.
6. Restart the Application Server.

Admin Console Tasks for Listeners

- Configuring Security for HTTP Listeners
- Configuring Security for IIOP Listeners
- Setting Listener Security Properties

Configuring Security for HTTP Listeners

Each virtual server in the HTTP service provides network connections through one or more *HTTP listeners*. With the Admin Console, you can create new HTTP listeners and edit the settings of existing HTTP listeners.

To edit security settings for an existing HTTP listener:

1. Expand the HTTP Service node.
2. Select the HTTP Listeners node.
3. Click the name of the HTTP listener for which you want to enable security.

   Alternatively, if you want to create a new listener, click New and follow the procedure in Creating an HTTP Listener.
4. Follow the procedure in Setting Listener Security Properties to set security properties.
5. Click Save to save the changes, or click Load Defaults to restore the listener to its default values.

Equivalent `asadmin` command: `create-http-listener`

Configuring Security for IIOP 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 EJBs and from other CORBA-based clients. With the Admin Console, you can create new IIOP listeners and edit the settings of existing IIOP listeners.

To edit security properties for an IIOP listener:

1. Expand the ORB node.
2. Select the IIOP Listeners node.
3. Click the name of the IIOP listener for which you want to enable security.

   Alternatively, if you want to create a new listener, click New and follow the procedure in Creating an IIOP Listener.
4. Follow the procedure in Setting Listener Security Properties to set security properties.
5. When you are done setting all properties, click Save to save the changes, or click Load Defaults to restore the properties to their default values.

If you created a new listener, it will now be listed in the Current Listeners table on the IIOP Listeners page.

Equivalent `asadmin` command: `create-iiop-listener`

Setting Listener Security Properties

Follow this common procedure for setting both HTTP listener and IIOP listener security properties:

1. In the Edit HTTP Listener or Edit IIOP Listener page, go to the section labeled Security.
2. Check the Enabled box in the Security field. When this option is selected, you must select SSL3/TLS to specify which type of security is enabled.
3. If you want clients to authenticate themselves to the Application Server when using this listener, check the Enabled box in the Client Authentication field.

4. Enter the name of an existing server keypair and certificate in the Certificate Nickname field.

   You can find the Certificate Nickname with the `keytool` command `keytool -list -v -keystore keystore.jks.`

   If you have changed the name of your keystore file, then use that name instead of `keystore.jks`.

5. In the SSL3/TLS section:

   a. Disable SSL3 or TLS if desired, but you must enable at least one. By default, both SSL3 and TLS will be enabled.

   b. Enable individual cipher suites if desired. By default, all supported cipher suites are enabled.

Security Tasks for Connector Connection Pools

- About Connector Connection Pools
- Creating a Security Map
- Mapping Principals

About Connector Connection Pools

A *connector module* (also called a resource adapter) enables J2EE applications to interact with enterprise information systems (EISs). A *connector resource* provides an application with a connection to an EIS. A *connector connection pool* is a group of reusable connections for a particular EIS.

Security maps enable you to create a mapping between J2EE users and groups and EIS users and groups. Use the `asadmin` command line utility to create, update, list, and delete security maps for connector connection pools.

Note: In this context, users are referred to as principals.

Creating a Security Map

A security map for a connector connection pool maps application users and groups (principals) to EIS principals. Use a security map when an application user needs to execute EIS operations that require a specific identity in the EIS.

Use the asadmin command create-connector-security-map to create or update security maps for a connector connection pool. If a security map already exists for a connector connection pool, this command appends the existing security map to the map provided by create-connector-security-map. This command supports supports the use of the wild-card asterisk (*) to indicate all users or all user groups.

Mapping Principals

When an application principal initiates a request to an EIS, the application server first checks for an exact principal using the security map defined for the connector connection pool to determine the mapped backend EIS principal. If there is no exact match, then the application server uses the wild card character specification, if any, to determine the mapped backend EIS principal.

For example, to map app_principal1 and app_principal2 in the application to dbuser1 in the EIS, use the following asadmin command:

```
asadmin> create-connector-security-map --connectionpoolid TESTPOOL
--principal app_principal1, app_principal2 --mappedusername dbuser1
```

Other Tasks for Security Maps

The asadmin utility also provides the following commands for working with security maps:

- delete-connector-security-map: deletes the specified security map.
- list-connector-security-maps: lists all security maps defined for the specfied connector connection pool.
- update-connector-security-map: modifies the specified security map.

Working with Certificates and SSL

- Certificate Files
- Using Keytool
- Generating a Server Certificate
- Signing a Digital Certificate
- Deleting a Certificate

Certificate Files

Installation of the Application Server generates a digital certificate suitable for internal testing. By default, the Application Server stores its certificate information in two files in the *install_dir*`/domains/`*domain_name*`/config` directory:

- **Keystore file**, by default named `keystore.jks`, containing the Application Server's digital certificate, including its private key. The keystore file is protected with a password, initially "changeit". You can change the password with `keytool`.

  Each keystore entry has a unique alias. After installation, the Application Server keystore has a single entry with alias "s1as."

- **Trust-store file**, by default named `cacerts.jks`, containing 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).

Changing the Location of Certificate Files

By default, the keystore and trust-store files are stored in the *install_dir*`/domains/`*domain_name*`/config` directory. To change this default location:

1. In the Admin Console tree component, select the Application Server node.
2. Click the JVM Settings tab.
3. Click the JVM Options sub-tab.
4. On the JVM Options page, edit the Debug Options field as follows:

   ```
   -Djavax.net.ssl.keyStore=${com.sun.aas.instanceRoot}/path/ks_name
   -Djavax.net.ssl.trustStore=${com.sun.aas.instanceRoot}/path/ts_name
   ```

   where *ks_name* is the keystore file name and *ts_name* is the trust-store file name.
5. Click Save.
6. Restart the Application Server.

Using Keytool

You can use `keytool to` set up and work with digital certificates. `keytool` ships with the J2SE software and enables you 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, you must have configured your shell environment properly such that the path refers to the J2SE `/bin` directory. For more information on keytool, see the keytool documentation at:

`http://java.sun.com/j2se/1.4.2/docs/tooldocs/solaris/keytool.html`

Generating a Server Certificate

You can use `keytool` to generate, import, and export certificates. By default, `keytool` creates a keystore file in the directory where you run it.

To generate a server certificate:

1. Change to the directory in which you want to generate the server certificate.

   Always generate the certificate in the directory containing the server's keystore and trust-store files, by default *install_dir*`/domains/`*domain_name*`/config`. For information on changing the location of these files, see Changing the Location of Certificate Files.

2. Enter the following `keytool` command to generate the server 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.

   You will be prompted for your name, organization, and other information that `keytool` uses to generate the certificate.

3. Enter the following `keytool` command to export the generated server certificate to the file `server.cer`:

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

4. If you want to have the certificate signed by a certificate authority, see Signing a Digital Certificate for more information.

5. To create the trust-store file `cacerts.jks` and add the server certificate to the trust-store, enter the following `keytool` command:

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

   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.

6. Type `yes`, then press Enter.

   Then `keytool` displays something like this:

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

7. Restart the Application Server.

For complete information about using `keytool`, see the `keytool` documentation at:

`http://java.sun.com/j2se/1.4.2/docs/tooldocs/solaris/keytool.html`

Signing a Digital Certificate

After creating a digital certificate, the owner must be 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 you simply want to ensure private secure communications, you can save the time and expense involved in obtaining a CA certificate and use a self-signed certificate.

Using a Certificate From a CA

To use a digital certificate signed by a CA:

1. Follow the instructions on the CA's website for generating certificate keypairs.

2. Download the generated certificate keypair.

   Save the certificate in the directory containing the server keystore and trust-store files, by default *install_dir*/`domains/domain1/config` directory. See Changing the Location of Certificate Files for instructions on changing this location.

3. In your shell, change to the directory where you have saved the certificate.

4. Use `keytool` to import the certificate into your local keystore and, if necessary, your local trust-store.

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

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

5. Restart the Application Server.

For complete information about using keytool, see the keytool documentation at:

`http://java.sun.com/j2se/1.4.2/docs/tooldocs/solaris/keytool.html`

Deleting a Certificate

You can delete a certificate with `keytool`. Follow this procedure:

1. Use `keytool -delete` to delete the existing certificate.

   ```
   keytool -delete
   -alias alias_name
   -keystore keystore_name
   -storepass password
   ```

   For a complete list of possible options for the `-delete` command, refer to the keytool documentation at:

Chapter 10

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:

- About Transactions
- Admin Console Tasks for Transactions

About Transactions

- What is a Transaction?
- Transactions in J2EE Technology

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 will be 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 completes 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) and the User Application. Each of these entities contribute to reliable transaction processing by implementing 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 run-time 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.

Admin Console Tasks for Transactions

- Configuring Transactions

Configuring Transactions

This section explains how to configure the following transaction attributes:

- Transaction Recovery
- Transaction Timeouts
- Transaction Logging

Transaction Recovery

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. Sun Java™ System Application Server Platform Edition 8 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 may contact the other servers to get the outcome of the transactions. If the other servers are unreachable, it will use the Heuristic Decision field to determine the outcome.

To configure how the Application Server recovers from transactions:

1. In the tree component select the Transaction Service node.
2. To enable the recovery of incomplete transactions check Recover in the On Restart field.
3. Set the amount of time, in seconds, the Application Server should try to connect to the unreachable server in the Retry Timeout field. The default value is 10 minutes.
4. Set the policy for unreachable servers in a transaction in the Heuristic Decision field.

   Unless you have a good reason to set this field to Commit, leave Heuristic Decision set to Rollback. Committing indeterminate transactions can compromise the data integrity of your application.

5. Click Save.

6. Restart the Application Server.

Transaction Timeouts

By default, the server will not timeout a transaction. That is, the server will wait 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 will rollback the transaction.

To set a timeout value:

1. In the tree component select the Transaction Service node.

2. Enter the number of seconds before the transaction times out in the Transaction Timeout field.

   The default value of Transaction Timeout is 0 seconds, which disables transaction timeouts.

3. Click Save.

4. Restart the Application Server.

Transaction Logging

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 meant to be human readable.

To set the location of the transaction logs:

1. In the tree component select the Transaction Service node.

2. Enter the location of the transaction logs in the Transaction Log Location field.

   A `tx` subdirectory will be created and transaction logs will be kept under that directory.

3. Click Save.

4. Restart the Application Server.

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, 1000) will result in larger transaction log files, but fewer keypoint operations, and potentially better performance. A smaller keypoint interval (for example, 20) results in smaller log files but slightly reduced performance due to the greater frequency of keypoint operations.

To set the keypoint interval:

1. In the tree component select the Transaction Service node.
2. Enter the number of transactions between keypoint operations in the Keypoint Interval field.
3. Click Save.
4. Restart the Application Server.

Chapter 11

The HTTP Service

This chapter describes how to configure virtual servers and HTTP listeners for the HTTP service of the Sun Java™ System Application Server Platform Edition 8.

- About the HTTP Service
- Admin Console Tasks for the HTTP Service
- Admin Console Tasks for HTTP Listeners
- Admin Console Tasks for Virtual Servers

About the HTTP Service

- What Is the HTTP Service?
- Virtual Servers
- HTTP Listeners

What Is the HTTP Service?

The HTTP service is the component of the Sun Java™ System Application Server Platform Edition 8 that provides facilities for deploying Web applications and for making deployed Web applications accessible by HTTP clients. (See Deploying a Web Application.) These facilities are provided by means of two kinds of related objects, virtual servers and HTTP listeners.

Virtual Servers

A virtual server, sometimes called a virtual host, is an object that allows you to use 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 Sun Java™ System Application Server Platform Edition 8 is running.

Note: Do not confuse an Internet domain with the administrative domain of the Application Server. (See Administrative Domains.)

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 will be 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 will be mapped to virtual host `www.aaa.com`, the second URL will be mapped to virtual host `www.bbb.com`, and the third will be mapped to virtual host `www.ccc.com`.

On the other hand, the following URL will result in a 404 return code, because `web3` wasn't registered with `www.bbb.com`:

```
http://www.bbb.com:8080/web3
```

For this mapping to work, you have to 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, you would add these domains to your `/etc/hosts` file (if the setting for `hosts` in your `/etc/nsswitch.conf` file includes `files`).

When you start the Application Server, it starts two 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 Admin Console)

For development, testing, and deployment of Web services in a non-production environment, `server` may be the only virtual server you need. You normally use additional virtual servers to 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 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, it 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, or with different IP addresses and the same port number. For example, you could specify 1.1.1.1:8081 and 1.1.1.1:8082. Additionally, you could specify 1.1.1.1:8081 and 1.2.3.4:8081, if your machine was configured to respond to both these addresses.

However, if you use 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 you have an HTTP listener using 0.0.0.0:8080 (all IP addresses on port 8080), you cannot also create an HTTP listener that uses 1.2.3.4:8080.

Since you typically have access to only one IP address for the system running the Application Server, you typically create listeners that use the 0.0.0.0 IP address and different port numbers, with each port number serving a different purpose. If you do have access to more than one IP address, you could instead use each address to serve a different purpose.

By default, when you start the Application Server, 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`.

All these listeners use the IP address 0.0.0.0 and the port numbers you specified as your HTTP server port numbers during installation. If you accepted the default values for the port numbers, `http-listener-1` uses port 8080, `http-listener-2` uses port 1043, and `admin-listener` uses port 4848.

Each HTTP listener has a default virtual server, which is the server to which it 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, you 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 where they are then picked up by worker threads. Ideally, you want to have 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. In the Application Server, there is no distinction between acceptor and request processing (worker) threads, i.e., each HTTP listener thread is responsible for accepting and processing requests. For this reason, the HTTP listeners in the Application Server's default configuration use 100 acceptor threads.

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 should be the alias name if your server uses an alias. If a client sends a `Host:` header, that host name supersedes the HTTP listener's server name value in redirects. You can also specify a redirect port if you do not want to use the port number specified in the original request.

A *redirect* occurs in one of these situations:

- When the resource you are trying to access no longer exists at the specified URL (that is, the resource has moved to another location), the server may redirect 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 may redirect the request to the SSL-enabled port. In this case, the server will return a new URL in the Location response header, in which the original nonsecure port has been replaced with the SSL-enabled port. The client will then connect to this new URL.

You also specify whether security is enabled for an HTTP listener and what kind of security you are using (for example, which SSL protocol and which ciphers).

This means that if you deploy a Web application on the Application Server, you access it with the URL `http://localhost:8080/` (or `http://localhost:1043/` if it is a secure application), along with the context root you specified for the Web application. You access the Admin Console with the URL `http://localhost:4848/` or `http://localhost:4848/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, you must create at least one HTTP listener before you create a new virtual server.

Admin Console Tasks for the HTTP Service

- Configuring the HTTP Service

Configuring the HTTP Service

1. In the tree component, select the HTTP Service node.
2. On the HTTP Service page, you can set properties that apply to all of the service's HTTP listeners.

   Table 11-1 lists these properties.
3. Click Save.

Table 11-1 HTTP Service Properties

Property Name	Description	Default Value
`bufferSize`	The size (in bytes) of the buffer to be provided for input streams created by HTTP listeners.	2048
`connectionTimeout`	The number of milliseconds HTTP listeners will wait, after accepting a connection, for the request URI line to be presented.	60000 (60 seconds)

Table 11-1 HTTP Service Properties *(Continued)*

Property Name	Description	Default Value
`maxKeepAliveRequests`	The maximum number of HTTP requests that can be pipelined until the connection is closed by the server. Setting this attribute to 1 will disable HTTP/1.0 keep-alive, as well as HTTP/1.1 keep-alive and pipelining.	100
`tcpNoDelay`	If set to true, the `TCP_NO_DELAY` option will be set on all HTTP listeners. This option improves performance under most circumstances.	true

Admin Console Tasks for Virtual Servers

- Creating a Virtual Server
- Editing a Virtual Server
- Deleting a Virtual Server

Creating a Virtual Server

1. In the tree component, expand the HTTP Service node.
2. Select the Virtual Servers node.
3. On the Virtual Servers page, click New. The Create Virtual Server page appears.
4. In the ID field, type the name of the domain. This value is used to identify the virtual server internally. It is not exposed to HTTP clients. The host names that are exposed to HTTP clients must be specified in the Hosts field.
5. In the Hosts field, type the host name or names for the machine on which the server is running. You may use either actual or virtual host names that are registered with the DNS server for your network (and, on a UNIX system, in your `/etc/hosts` file).
6. In the area opposite State, select either On, Off, or Disabled. The default is On.

7. Leave the HTTP Listeners field empty. It will be filled in automatically when you create an HTTP listener and associate it with this server.

   (If you use this field, you must specify an existing HTTP listener. You may not, however, specify a listener that is used by another virtual server; if you do, the Application Server will not restart and you will have to delete the new virtual server manually from the `domain.xml` file. Since a listener must be associated with an existing virtual server when it is created, all existing listeners are used by another virtual server.)

8. In the Default Web Module combo box, choose the deployed Web module (if any) that will respond to all requests that cannot be mapped to other web modules deployed to the virtual server.

   If you do not specify a default web module, the web module that has an empty context root is used. If there is no web module with an empty context root, a system default web module is created and used.

9. In the Log File field, type the path name of the file where logging messages from this virtual server will appear. Leave this field empty if you want logging messages to go to the default server log (*install_dir*`/domains/domain1/config/server.log`).

10. In the Additional Properties area, click Add Property to add a property for the virtual server. Whether you specify properties or not, the new server will have the default properties `docroot` and `accesslog`, set to default values.

11. Click OK to save the virtual server.

You must stop and restart the Application Server in order to use the new virtual server.

You can set the properties listed in the following table.

Table 11-2 Virtual Server Properties

Property Name	Description
`docroot`	Absolute path to root document directory for server. Default is *install_dir*`/domains/domain1/docroot`
`accesslog`	Absolute path to server access logs. Default is *install_dir*`/domains/domain1/logs/access`.

Table 11-2 Virtual Server Properties ***(Continued)***

Property Name	Description
`sso-enabled`	If false, single sign-on is disabled for this virtual server, and users must authenticate separately to every application on the virtual server. Single sign-on across applications on the Application Server is supported by servlets and JSP pages. This feature allows multiple applications that require the same user sign-on information to share this information, rather than have the user sign on separately for each application. Default is true.
`sso-max-inactive-seconds`	Specifies the number of seconds after which a user’s single sign-on record becomes eligible for purging if no client activity is received. Since single sign-on applies across several applications on the same virtual server, access to any of the applications keeps the single sign-on record active. Default is 300 seconds (5 minutes). Higher values provide longer single sign-on persistence for users at the expense of more memory use on the server.
`sso-reap-interval-seconds`	Specifies the number of seconds between purges of expired single sign-on records. Default is 60.

Equivalent `asadmin` command: `create-virtual-server`

Editing a Virtual Server

1. In the tree component, expand the HTTP Service node.
2. Select the Virtual Servers node.
3. Select the virtual server to be edited.
4. On the Edit Virtual Server page, you can perform these tasks:
   - Change the host name in the Hosts field.
   - Change the value of the State setting.
   - Add or remove an HTTP listener.
   - Change the Default Web Module selection.
   - Change the Log File value.

- Add, remove, or modify properties.

5. Click Save to save your changes.
6. Stop and restart the Application Server.

Deleting a Virtual Server

1. In the tree component, expand the HTTP Service node.
2. Select the Virtual Servers node.
3. On the Virtual Servers page, select the checkbox next to the name of the virtual server to be deleted.
4. Click Delete.
5. Stop and restart the Application Server.

It is possible to delete the `server` and `__asadmin` virtual servers, but this is not recommended. If you plan to do so, you should first copy the `virtual-server` elements of the Application Server's `domain.xml` file to a safe place so that you can restore the settings if needed.

Equivalent `asadmin` command: `delete-virtual-server`

Admin Console Tasks for HTTP Listeners

- Creating an HTTP Listener
- Editing an HTTP Listener
- Deleting an HTTP Listener

Creating an HTTP Listener

1. In the tree component, expand the HTTP Service node.
2. Select the HTTP Listeners node.
3. On the HTTP Listeners page, click New. The Create HTTP Listener page appears.
4. In the Name field, type a name for the listener.

5. In the Listener field, deselect the Enabled checkbox if you do not want to enable the listener when you restart the server.

6. In the Network Address field, type 0.0.0.0 if you want the listener to listen on all IP addresses for the server, using a unique port value. Otherwise, type a valid IP address for the server.

   Note: The values any, ANY, and INADDR_ANY are also valid values for this field, but are currently unsupported (bug #5004719).

7. In the Listener Port field, type a unique port value if the Network Address field is 0.0.0.0, or the desired port value if you are using another IP address.

8. Choose a virtual server from the Default Virtual Server combo box.

9. In the Server Name field, type the host name to be used in the URLs the server sends to the client. This name should be the alias name if your server uses an alias. If your server does not use an alias, leave this field empty.

   Note: The empty string is a valid value for this field, but the Admin Console does not allow you to leave the field empty (bug #5007292). Workaround: After you save the listener, edit the field and remove the string you entered.

10. In the Advanced area, you may perform the following tasks:

    - To redirect requests to another port, type a value in the Redirect Port field. The Application Server will automatically redirect the request if these two conditions exist: First, this listener is supporting non-SSL requests; and second, a request is received for which a matching security constraint requires SSL transport. By default, the Application Server uses the port number specified in the original request.

    - Change the number of Acceptor Threads (the default is 1, but 100 is a better value).

    - Deselect the Powered By checkbox to disable the inclusion of the `X-Powered-By: Servlet/2.4` header in servlet-generated HTTP response headers.

      The Java Servlet 2.4 Specification defines this header, which containers may add to servlet-generated responses. Similarly, the JavaServer Pages™ (JSP™) 2.0 Specification defines an `X-Powered-By: JSP/2.0` header to be added (on an optional basis) to responses that use JSP technology. The inclusion of the `X-Powered-By: JSP/2.0` header is enabled by default for web applications. The goal of these headers is to aid Web site administrators in gathering statistical data about the use of Servlet and JSP technology.

For information on enabling and disabling the `X-Powered-By` header for JSP pages, see the chapter entitled “Deployment Descriptor Files” in the *Sun Java™ System Application Server Platform Edition 8 Developer’s Guide*. See Further Information for a link to this document.

Production environments may decide to omit the generation of `X-Powered-By` headers to hide their underlying technology.

11. If you don’t want to make the listener secure, click OK.

In the Security section of this page, you can configure the listener to use SSL, TLS, or both SSL and TLS security.

To set up a secure listener, do the following:

1. Check the Enabled box in the Security field.
2. If you want clients to authenticate themselves to the server when using this listener, check the Enabled box in the Client Authentication field.
3. Enter the name of an existing server keypair and certificate in the Certificate NickName field. See the Security chapter for more information.
4. In the SSL3/TLS section:
   a. Check the security protocol(s) you want to enable on the listener. You must check either SSL3 or TLS, and can enable both protocols.
   b. Check the cipher suite used by the protocol(s). To enable all cipher suites, check All Supported Cipher Suites. You may also enable individual cipher suites.

The listener will now be listed in the HTTP Listeners field for the virtual server that you specified as the Default Virtual Server.

You must stop and restart the Application Server in order to use the new HTTP listener.

Equivalent `asadmin` command: `create-http-listener`

Editing an HTTP Listener

1. In the tree component, expand the HTTP Service node.

2. Select the HTTP Listeners node.
3. Select the HTTP listener to be edited.
4. On the Edit HTTP Listener page, modify any of the properties you set when you created the listener.
5. Click Save to save your changes.
6. Stop and restart the Application Server.

Deleting an HTTP Listener

1. In the tree component, expand the HTTP Service node.
2. Select the HTTP Listeners node.
3. On the HTTP Listeners page, select the checkbox next to the name of the HTTP listener to be deleted.
4. Click Delete.
5. Stop and restart the Application Server.

It is possible to delete the `http-listener-1`, `http-listener-2`, and `admin-listener` HTTP listeners, but this is not recommended. If you plan to do so, you should first copy the `http-listener` elements of the Application Server's `domain.xml` file to a safe place so that you can restore the settings if needed.

Equivalent `asadmin` command: `delete-http-listener`

Chapter 12

The Object Request Broker

This chapter describes how to configure the Object Request Broker (ORB) and IIOP listeners. It has the following sections:

- About the Object Request Broker
- Admin Console Tasks for the ORB
- Admin Console Tasks for IIOP Listeners

About the Object Request Broker

- CORBA
- What is the ORB?
- IIOP Listeners

CORBA

The Sun Java™ System Application Server Platform Edition 8 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 EJB components and from other CORBA-based clients. You can configure multiple IIOP listeners for the Application Server. For each listener, you specify a port number, a network address, and optionally, security attributes. For more information, see Creating an IIOP Listener.

Admin Console Tasks for the ORB

- Configuring the ORB

Configuring the ORB

1. In the tree component select the ORB node.

2. Select the thread pool the ORB uses in the Thread Pool ID combo box.

   The ORB uses thread pools to respond to requests from remote clients of EJB modules and other clients that communicate via RMI-IIOP. For more information, see the sections, Thread Pools in the Application Server and Creating Thread Pools.

3. in the Max Message Fragment Size field, set the maximum fragment size for IIOP messages.

   Messages larger than this size will be fragmented.

4. In the Total Connections field, set the maximum number of incoming connections for all IIOP all listeners.

5. Restart the server.

Admin Console Tasks for IIOP Listeners

- Creating an IIOP Listener
- Editing an IIOP Listener
- Deleting an IIOPListener

Creating an IIOP Listener

1. In the tree component expand the ORB node.
2. Select IIOP Listeners.
3. Click New.
4. Enter a name to identify the listener in the Name field.
5. Enter the network address of the listener in the Network Address field.

   This can be an IP address or a DNS resolvable host name.

6. Enter the port number the listener will listen on in the Listener Port field.
7. Check the Enabled box in the Listener field to enable the listener.
8. If you don’t want to make the listener secure, click OK.

In the Security section of this page, you can configure the listener to use SSL, TLS, or both SSL and TLS security.

To set up a secure listener, do the following:

1. Check the Enabled box in the Security field.
2. If you want clients to authenticate themselves to the server when using this listener, check the Enabled box in the Client Authentication field.
3. Enter the name of an existing server keypair and certificate in the Certificate NickName field.
4. In the SSL3/TLS section:
    a. Check the security protocol(s) you want to enable on the listener. You must check either SSL3 or TLS, and can enable both protocols.
    b. Check the cipher suite used by the protocol(s). To enable all cipher suites, check All Supported Cipher Suites. You may also enable individual cipher suites.
5. Click OK.

The listener will now be listed in the Current Listeners table on the IIOP Listeners page.

Editing an IIOP Listener

1. In the tree component expand the ORB node.
2. Select the IIOP Listeners node.
3. Select the listener you want to modify in the Current Listeners table.
4. Modify the listener's settings. See Creating an IIOP Listener for descriptions of the fields you can modify.

Deleting an IIOPListener

1. In the tree component expand the ORB node.
2. Select the IIOP Listeners node.
3. Check the listener(s) you want to delete in the Current Listeners table.
4. Click Delete.

Chapter 13

Thread Pools

This chapter describes how to create, edit, and delete thread pools. It has the following sections:

- About Thread Pools
- Admin Console Tasks for Thread Pools

About Thread Pools

- Thread Pools in the Application Server

Thread Pools in the Application Server

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. You can 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.

You can 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 you specify 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 you specify.

Increasing the number of threads available to a process allows the process to respond to more application requests simultaneously.

You can 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.

Admin Console Tasks for Thread Pools

- Creating Thread Pools
- Editing Thread Pools
- Deleting Thread Pools

Creating Thread Pools

1. In the tree component select the Thread Pools node.
2. Under Current Pools click New.
3. Enter the name of the thread pool in the Thread Pool ID field.
4. 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.
5. 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.
6. Enter the number, in seconds, after which idle threads are removed from pool in the Idle Timeout field.
7. Enter the total number of work queues that are serviced by this thread pool in the Number of Work Queues field.
8. Click OK.
9. Restart the Application Server.

Editing Thread Pools

1. In the tree component select the Thread Pools node.
2. Under Current Pools select the name of the thread pool you want to change.
3. 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.
4. 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.
5. Enter the number, in seconds, after which idle threads are removed from pool in the Idle Timeout field.
6. Enter the total number of work queues that are serviced by this thread pool in the Number of Work Queues field.
7. Click Save.
8. Restart the Application Server.

Deleting Thread Pools

1. In the tree component select the Thread Pools node.
2. Check the thread pool name you want to delete in the Current Pools table.
3. Click Delete.
4. Restart the Application Server.

Chapter 14

Logging

This chapter briefly describes how to use the Admin Console to configure logging and view the server log. It contains the following sections:

- About Logging
- Admin Console Tasks for Logging

About Logging

- Log Records
- The Logger Namespace Hierarchy

Log Records

The Sun Java™ System Application Server Platform Edition 8 uses the Java 2 platform Logging API specified in JSR 047. Logging messages are recorded in the server log, normally found at *install_dir*`/domains/domain1/logs/server.log`.

The components of the Application Server generate logging output. You may also specify loggers for your own application components.

Log records follow a uniform format:

```
[#|yyyy-mm-ddThh:mm:ss.SSS-Z|Log Level|ProductName_Version|LoggerName|Key
Value Pairs|MessageId: Message|#]
```

For example:

```
[#|2004-01-29T11:43:43.516-0500|INFO|sun-appserver-pe8.0|javax.enterprise.
system.core|_ThreadID=14;|CORE5004: Resource Deployed:
[ccp:jms/DurableTopicConnectionFactory].|#]
```

In this example,

- [# and #] mark the beginning and end of the record.
- The vertical bar (|) separates the record fields.
- 2004-01-29T11:43:43.516-0500 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-pe8.0.
- The *LoggerName* is a hierarchical logger namespace that identifies the source of the log module, in this case javax.enterprise.system.core. All logger names begin with either javax.enterprise.system or javax.enterprise.core.
- The *Key Value Pairs* are key names and values, typically a thread ID such as _ThreadID=14;.
- The *MessageId* is an identifier that is provided for all SEVERE and WARNING messages and many INFO messages. Typically it consists of a module code and a numerical value, such as ADM1042 or, in this case, CORE5004.
- The *Message* is the text of the log message.

The log record format may be changed or enhanced in future releases.

The Logger Namespace Hierarchy

The Sun Java™ System Application Server Platform Edition 8 provides a logger for each of its modules. Table 14-1 lists the names of the modules and the namespace for each logger in the order in which they appear on the Log Levels page (see Configuring Log Levels). The last four modules in the table do not appear on the Log Levels page.

Table 14-1 Application Server Logger Namespaces

Module Name	Namespace
Root	javax.enterprise
Server	javax.enterprise.system
EJB Container: Enterprise JavaBeans components	javax.enterprise.system.container.ejb
MDB Container: Message-driven beans	javax.enterprise.system.container.ejb.mdb
Web Container	javax.enterprise.system.container.web

Table 14-1 Application Server Logger Namespaces

Module Name	Namespace
Classloader	`javax.enterprise.system.core.classloading`
Configuration	`javax.enterprise.system.core.config`
Naming: Java Naming and Directory Interface	`javax.enterprise.system.core.naming`
Security	`javax.enterprise.system.core.security`
JTS: Java Transaction Service (transaction manager)	`javax.enterprise.system.core.transaction`
JTA: Java Transaction API	`javax.enterprise.resource.jta`
Admin: Server administration commands and tools	`javax.enterprise.system.tools.admin`
Deployment: Deployment of resources and applications	`javax.enterprise.system.tools.deployment`
Verifier	`javax.enterprise.system.tools.verifier`
JAXR: Java API for XML Registries	`javax.enterprise.resource.webservices.registry`
JAX-RPC: Java API for XML-Based RPC	`javax.enterprise.resource.webservices.rpc`
SAAJ: SOAP with Attachments API for Java	`javax.enterprise.resource.webservices.saaj`
CORBA	`javax.enterprise.resource.corba`
JavaMail	`javax.enterprise.resource.javamail`
JMS: Java Message Service	`javax.enterprise.resource.jms`
Connector: J2EE Connector Architecture	`javax.enterprise.resource.resourceadapter`
JDO: Java Data Objects	`javax.enterprise.resource.jdo`
CMP: Entity beans that use container-managed persistence	`javax.enterprise.system.container.cmp`
Core	`javax.enterprise.system.core`
Util	`javax.enterprise.system.util`
System Output (System.out.println)	`javax.enterprise.system.stream.out`
System Error (System.err.println)	`javax.enterprise.system.stream.err`

Admin Console Tasks for Logging

- Configuring General Logging Settings

- Configuring Log Levels
- Viewing the Server Log

Configuring General Logging Settings

1. In the tree component, select the Application Server node.
2. Click the Logging tab.

On the Logging Settings page, you can customize logging using the following fields:

- **Log File**: To specify an alternative name or location for the server log file, type the new path name in the text field. By default, the location is *install_dir*`/domains/domain1/logs/server.log`.
- **Log Messages to Standard Error**: To send logging messages to both standard error and the server log file whether or not you use the `--verbose` option to start the Application Server, select the Enabled checkbox. By default, when you do not use the `--verbose` option, the messages go only to the server log file.
- **Write to System Log**: On Solaris systems only, to send logging output to the Solaris `syslog` facility in addition to the server log, select the Enabled checkbox.
- **Log Handler**: To send logs to a destination other than `server.log` or `syslog`, you can plug in a custom log handler. The custom handler must extend the class `java.util.logging.Handler` (a JSR 047 compliant API). Type the absolute class name of the handler in the Log Handler field. Also put the handler class in the Application Server classpath so that the handler can be installed during server startup. The log records from the custom handler will have the format described in Log Records.
- **Log Filter**: To filter log records that are sent to destinations such as `server.log`, `syslog`, or a destination specified by a custom log handler, you can plug in a custom log filter. The custom filter must implement the interface `java.util.logging.Filter`. Type the absolute class name of the filter in the Log Filter field. Also put the filter class in the Application Server classpath so that the filter can be installed during server startup.

- **File Rotation Limit**: When the server log reaches the specified size in bytes, create a new, empty file named `server.log` and rename the old file `server.log_`*`date`*, where *`date`* is the date and time when the file was rotated. The default value is 2 gigabytes. The minimum value for the limit is 500 kilobytes; if you specify a lower value, the file will rotate when it reaches 500 Kbytes.

Click Save to save your changes.

When you make changes to this page, you must stop and restart the server in order for the changes to take effect.

Configuring Log Levels

1. In the tree component, select the Application Server node.
2. Click the Logging tab.
3. On the Logging Settings page, click Log Levels.
4. On the Module Log Levels page, choose a new value from the combo box opposite the module or modules whose log level you wish to change. The default level is `INFO`, meaning that messages at that level or higher (`WARNING`, `SEVERE`) will appear in the log. You may specify any of the following values (listed from highest to lowest):
   - `SEVERE`
   - `WARNING`
   - `INFO`
   - `CONFIG`
   - `FINE`
   - `FINER`
   - `FINEST`
   - `OFF`
5. Use the Additional Properties area to configure log levels for any application loggers you specified on the Logging Settings page. The property name is the logger namespace, and the value is one of the eight possible levels.

   Also use this area to change the log level for a submodule, such as the transport submodule of the CORBA module:

`javax.enterprise.resource.corba.`*`ORBId`*`.transport`

6. Click Save to save your changes, or click Load Defaults to restore the default values.

Calls to `System.out.println` will be logged at the `INFO` level using the logger name `javax.enterprise.system.stream.out`. Calls to `System.err.println` will be logged at the `WARNING` level using the logger name `javax.enterprise.system.stream.err`. To turn off the logs from these sources, specify the logger name with the value `OFF` in the Additional Properties area.

Changes to the Log Level settings take effect immediately. They are also saved in the `domain.xml` file to be used when the server restarts.

Viewing the Server Log

1. In the tree component, select the Application Server node.
2. Click the Logging tab.
3. On the Logging Settings page, click Open Log Viewer. The Log Viewer opens in a new window.

The most recent 40 entries in the server log appear, with the settings you specified on the Logging Settings and Log Levels pages.

Click the triangle next to the Timestamp header to sort the messages so that the most recent one appears last.

To view a formatted version of any message, click the link marked

```
(details)
```

A window labeled Log Entry Detail appears, with a formatted version of the message.

Note

Click Modify Search to go to an area that allows you to customize and filter the log viewer. Use the fields as follows:

- **Timestamp**: To view the most recent messages, select Most Recent (the default). To view messages only from a certain period of time, select Specific Range and type a date and time value in the From and To fields. For the Time value, the syntax must take the following form (*SSS* stands for milliseconds):

  `hh``:`*`mm`*`:`*`ss`*`.`*`SSS`*

  For example:

```
17:10:00.000
```

- **Log Level**: To filter messages by log level, choose a log level from the combo box and select either This Log Level and Coarser or This Log Level Only.

  By default, all log messages specified by the Log Levels page will appear.

- **Logger**: To filter by module, select one or more namespaces from the combo box. You can use either shift-click or control-click to choose multiple namespaces.

  Choosing a namespace at a higher level selects all the namespaces below it. For example, if you select `javax.enterprise.system`, you get the loggers for all the modules under that namespace: `javax.enterprise.system.core`, `javax.enterprise.system.tools.admin`, and so on.

- **Custom Logger**: To view messages from loggers specific to your application, type the logger names in the text field, one per line. If your application has several modules, you may view any or all of them. For example, suppose your application has loggers with the following names:

  ```
  com.mycompany.myapp.module1
  com.mycompany.myapp.module2
  com.mycompany.myapp.module3
  ```

  To view messages from all modules in the application, type `com.mycompany.myapp`. To view messages from `module2` only, type `com.mycompany.myapp.module2`.

  To view the messages from Tomcat, type `org.apache` in this field.

  If you specify one or more custom loggers, messages from the Application Server modules will not appear.

- **Name-Value Pairs**: To view output from a specific thread, type the key name and value for that thread in the text field. The key name is `_ThreadID`. For example:

  ```
  _ThreadID=13
  ```

  Suppose that `com.mycompany.myapp.module2` runs in several threads. You can refine the log viewer to show only the output from a single thread by specifying that module's logger in the Custom Logger field and then specifying the thread ID in this field.

- **Display**: To view more than 40 messages at a time (the default), choose another of the available values from the combo box (100, 250, or 1000).

To view stack traces, deselect the “limit excessively long messages” checkbox. By default, stack traces do not appear in the viewer.

Chapter 15

Monitoring

This chapter contains information about monitoring components in the Sun Java™ System Application Server Platform Edition 8. This chapter contains the following sections:

- About Monitoring
- Enabling and Disabling Monitoring
- Viewing Monitoring Data With the asadmin Tool

About Monitoring

- Monitoring the Application Server
- General Steps for Monitoring
- The Tree Structure of Monitorable Objects
- Statistics for Monitored Components and Services

Monitoring the Application Server

You use monitoring to observe the runtime state of various components of the Application Server. With the information on the state of runtime components and processes, you can 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.

General Steps for Monitoring

To monitor the Application Server, you perform these steps:

1. Using either the Admin Console or the `asadmin` tool, enable the monitoring of specific services and components. See Enabling and Disabling Monitoring.
2. In a terminal window, type the `asadmin list --monitor "server.*"` command to view the names of the objects that can be monitored.
3. Type the `asadmin get --monitor` command and specify a name displayed by the `list` command in the preceding step. The `get` command displays the monitoring statistics.

   For more information on the `get` and `set` commands, see Viewing Monitoring Data With the asadmin Tool.

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 this release, just one server instance is permitted.)

The following command displays the top level of the tree:

```
asadmin> list --monitor server
password>
server.applications
server.http-service
server.jvm
server.orb
server.resources
server.thread-pools
server.transaction-service
```

The following sections describe these sub-trees:

- The Applications Tree
- The HTTP Service Tree
- The Resources Tree
- The ORB Tree

- The Thread Pool Tree

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 (*). See EJB Container Statistics and Web Container Statistics.

Figure 15-1 Applications Node Tree Structure

```
applications
   |--- application1
   |    |--- ejb-module-1
   |    |        |--- ejb1 *
   |    |                  |--- cache (for entity/sfsb) *
   |    |                  |--- pool (for slsb/mdb/entity) *
   |    |                  |--- method
   |    |                       |---method1 *
   |    |                       |---method2 *
   |    |--- web-module-1
   |    |        |--- virtual-server-1
   |    |                       |---servlet1  *
   |    |                       |---servlet2  *
   |--- standalone-web-module-1
   |    |        |----- virtual-server-1
   |    |                       |---servlet3 *
   |    |                       |---servlet4 *
   |    |        |----- virtual-server-2
   |    |                      |---servlet3 *(same servlet on different vs)
   |    |                      |---servlet5 *
   |--- standalone-ejb-module-1
   |    |        |--- ejb2 *
   |    |                  |--- cache (for entity/sfsb) *
   |    |                  |--- pool (for slsb/mdb/entity) *
   |    |                  |--- method
   |    |                       |--- 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 The HTTP Service Tree.

Figure 15-2 HTTP Service Schematic

```
http-service
            |        |--- virtual-server-1
            |        |           |--- http-listener-1 *
            |        |           |--- http-listener-2 *
            |        |--- virtual-server-2
            |        |           |--- http-listener-1 *
            |        |           |--- http-listener-2 *
```

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 and Connector Connection Pools Statistics.

Figure 15-3 Resources Schematic

```
resources
       | |---connection-pool1(either connector-coonection-pool or jdbc)*
       | |---connection-pool2(either connector-coonection-pool or jdbc)*
```

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 ORB Statistics.

Figure 15-4 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.

Figure 15-5 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:

- EJB Container Statistics
- Web Container Statistics
- The HTTP Service Tree
- JDBC Connection Pools Statistics
- Connector Connection Pools Statistics
- ORB Statistics
- Thread Pools Statistics
- Transaction Service Statistics
- Java Virtual Machine (JVM) Statistics

EJB Container Statistics

The statistics that are available for the EJB container are described in Table 15-1.

Table 15-1 EJB Container Statistics

Attribute Name	Data Type	Description
`createcount`	Count Statistic	Number of times an EJB's `create` method was called.
`removecount`	Count Statistic	Number of times an EJB's `remove` method was called.
`pooledcount`	Range Statistic	Number of entity beans in pooled state.
`readycount`	Range Statistic	Number of entity beans in ready state.
`messagecount`	Count Statistic	Number of messages received for a message-driven bean.
`methodreadycount`	Range Statistic	Number of stateful or stateless session beans (as the case may be) that are in the `MethodReady` state.
`passivecount`	Range Statistic	Number of stateful session beans that are in `Passive` state.

The statistics available for EJB method invocations are listed in Table 15-2.

Table 15-2 EJB Method Statistics

Attribute Name	Datatype	Description
`methodstatistic`	Time Statistic	Number of times an operation was called, the total time that was spent during the invocation, and so on.
`totalnumerrors`	Count Statistic	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.
`totalnumsuccess`	Count Statistic	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`	Count Statistic	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 available for EJB pools are listed in Table 15-3.

Table 15-3 EJB Pool Statistics

Attribute Name	Data Type	Description
`poolresizequantity`	Count Statistic	The increment by which pool grows or shrinks.
`idletimeoutseconds`	Boundary Statistic	Defines the rate at which the pool cleaning thread is executed. Only objects that have not been accessed for more than `idletimeoutseconds` are candidates for removal
`numbeansinpool`	Bounded Range Statistic	Number of EJB's in the associated pool, providing an idea about how the pool is changing.
`numthreadswaiting`	Bounded Range Statistic	Number of threads waiting for free beans, giving an indication of possible congestion of requests.
`totalbeanscreated`	Count Statistic	Number of beans created in associated pool since the gathering of data started.
`totalbeansdestroyed`	Count Statistic	Number of beans destroyed from associated pool since the gathering of data started.
`jmsmaxmessagesload`	Count Statistic	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 Table 15-4.

Table 15-4 EJB Cache Statistics

Attribute Name	Datatype	Description
`cachemisses`	Bounded Range Statistic	The number of times a user request did not find a bean in the cache.
`cachehits`	Bounded Range Statistic	The number of times a user request found an entry in the cache.
`numbeansincache`	Bounded Range Statistic	The number of beans in the cache. This is the current size of the cache.
`numpassivations`	Count Statistic	Number of passivations. Applies only to stateful session beans.

Table 15-4 EJB Cache Statistics *(Continued)*

Attribute Name	Datatype	Description
numpassivationerrors	Count Statistic	Number of errors during passivation. Applies only to stateful session beans.
numexpiredsessionsremoved	Count Statistic	Number of expired sessions removed by the cleanup thread. Applies only to stateful session beans.
numpassivationsuccess	Count Statistic	Number of times passivation completed successfully. Applies only to stateful session beans.

Web Container Statistics

The Web container fits into the tree of objects as shown in Figure 15-1. Web container statistics are displayed for each individual web application.

Table 15-5 Web Container (Servlet) Statistics

Statistic	Units	Data Type	Comments
errorcount	Number	CountStatistic	Cumulative umber of cases where the response code was greater than or equal to 400.
maxtime	Milliseconds	CountStatistic	The maximum amount of time the Web container will wait for requests.
processingtime	Milliseconds	CountStatistic	Cumulative value of the amount of time required to process each request. The processing time is the average of request processing times divided by the request count.
requestcount	Number	CountStatistic	The total number of requests processed so far.

HTTP Service Statistics

The statistics available for the HTTP service are shown in Table 15-6.

Table 15-6 HTTP Service Statistics

Statistic	Units	Data Type	Comments
`bytesreceived`	Bytes	Count Statistic	The cumulative value of the bytes received by each of the request processors.
`bytessent`	Bytes	Count Statistic	The cumulative value of the bytes sent by each of the request processors.
`currentthreadcount`	Number	Count Statistic	The number of processing threads currently in the listener thread pool.
`currentthreadsbusy`	Number	Count Statistic	The number of request processing threads currently in use in the listener thread pool serving requests.
error-count	Number	Count Statistic	The cumulative value of the error count, which represents the number of cases where the response code was greater than or equal to 400.
`maxsparethreads`	Number	Count Statistic	The maximum number of unused response processing threads that will be allowed to exist.
`minsparethreads`	Number	Count Statistic	The minimum number of unused response processing threads that will be allowed to exist.
`maxthreads`	Number	Count Statistic	The maximum number of request processing threads created by the listener.
max-time	Milliseconds	Count Statistic	The maximum amount of time for processing threads.
min-spare-time	Milliseconds	Count Statistic	The minimum amount of time for processing spare threads.
processing-time	Milliseconds	Count Statistic	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	Count Statistic	The total number of requests processed so far.

JDBC Connection Pools Statistics

You can 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 Table 15-7.

Table 15-7 JDBC Connection Pool Statistics

Statistic	Units	Data Type	Description
`numconnfailedvalidation`	Number	Count Statistic	The total number of connections in the connection pool that failed validation from the start time until the last sample time.
`numconnused`	Number	Range Statistic	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	Count Statistic	The total number of free connections in the pool as of the last sampling.
`numconntimedout`	Number	Bounded Range Statistic	The total number of connections in the pool that timed out between the start time and the last sample time.

Connector Connection Pools Statistics

The statistics available for the connector connection pool are shown in Table 15-8.

Table 15-8 Connector Connection Pool Statistics

Statistic	Units	Data Type	Description
`numconnfailed validation`	Number	Count Statistic	The total number of connections in the connection pool that failed validation from the start time until the last sample time.

Table 15-8 Connector Connection Pool Statistics

Statistic	Units	Data Type	Description
`numconnused`	Number	Range Statistic	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	Count Statistic	The total number of free connections in the pool as of the last sampling.
`numconntimedout`	Number	Bounded Range Statistic	The total number of connections in the pool that timed out between the start time and the last sample time.

ORB Statistics

The statistics available for the ORB are shown in Table 15-9.

Table 15-9 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	BoundedRange Statistic	Total number of connections to the ORB.

Thread Pools Statistics

The statistics available for the thread pool are shown in Table 15-10.

Table 15-10 Thread Pool Statistics

Statistic	Units	Data Type	Description
`averagetimeinqueue`	Milliseconds	RangeStatistics	The average amount of time in milliseconds a request waited in the queue before getting processed.

Table 15-10 Thread Pool Statistics

Statistic	Units	Data Type	Description
`averageworkcompletion-time`	Milliseconds	RangeStatistics	The average amount of time taken to complete an assignment, in milliseconds.
`currentnumberofthreads`	Number	BoundedRange Statistic	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.
`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 Table 15-11.

Table 15-11 Transaction Service Statistics

Statistic	Units	Data Type	Description
`activecount`	Number	CountStatistic	Number of transactions currently active.
`activeids`			The ID's of the transactions that are currently active. Every such transaction can be rolled back after freezing the transaction service.
`committedcount`	Number	CountStatistic	Number of transactions that have been committed.
`rolledbackcount`	Number	CountStatistic	Number of transactions that have been rolled back.
`state`			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 Table 15-12.

Table 15-12 JVM Statistics

Statistic	Units	Data Type	Description
`heapsize`	Bytes	BoundedRange Statistic	The resident memory footprint with the higher and lower bounds of the JVM's memory heap size.
`uptime`	Milliseconds	CountStatistic	The amount of time the JVM has been running.

Enabling and Disabling Monitoring

- Enabling and Disabling Monitoring With the Admin Console
- Enabling and Disabling Monitoring With the asadmin Tool

Enabling and Disabling Monitoring With the Admin Console

1. In the tree component, select the Application Server node.
2. Click the Monitoring tab.
3. On the Monitoring Service page, choose the appropriate value from the combo box opposite the component(s) or service(s) whose monitoring level you wish to change.

   By default, monitoring is turned off for all components and services except for the Java Virtual Machine (JVM), which is always monitorable. To turn monitoring on, select LOW or HIGH from the combo box. To turn monitoring off, select OFF from the combo box. You can turn monitoring on or off for the following components and services:

   - JDBC Connection Pool - set the monitoring level to LOW for this option to monitor all JDBC connection pools.

- EJB Container - set the monitoring level to LOW for this option to monitor all deployed EJBs, EJB pools, and EJB caches. Set this method to HIGH to also monitor EJB business methods.
- Thread Pool - set the monitoring level to LOW for this option to monitor all thread pools.
- Web Container - set the monitoring level to LOW for this option to monitor all deployed servlets.
- HTTP Service - set the monitoring level to LOW for this option to monitor all HTTP listeners and virtual servers.
- Connector Connection Pool - set the monitoring level to LOW for this option to monitor all created connector connection pools.
- ORB - set the monitoring level to LOW for this option to monitor the system ORB used by the Application Server core and its connection managers.
- Transaction Service - set the monitoring level to LOW for this option to monitor any transaction subsystem.

4. Click Save.

There are no Additional Monitoring Service Properties in this release, so you can ignore the Additional Properties table.

Enabling and Disabling Monitoring With the asadmin Tool

1. Use the `get` command to find out what services and components currently have monitoring enabled:

```
asadmin> get server.monitoring-service.module-monitoring-levels.*
```

Returns:

```
server.monitoring-service.module-monitoring-levels.
connector-connection-pool = OFF
server.monitoring-service.module-monitoring-levels.ejb-container = OFF
server.monitoring-service.module-monitoring-levels.http-service = OFF
server.monitoring-service.module-monitoring-levels.jdbc-connection-pool =
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
```

2. Use the `set` command to enable monitoring.

   For example, to enable monitoring for the HTTP service:

```
asadmin> set --user admin_user --password admin_password
server.monitoring-service.module-monitoring-levels.http-service=HIGH
```

3. To disable monitoring, use the `set` command and specify `OFF` for the monitoring level.

Viewing Monitoring Data With the asadmin Tool

- Dotted Names
- Examples of the list and get Commands
- Expected Output for list and get Commands at All Levels

Dotted Names

In the `asadmin list` and `get` commands, you 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 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
```

You are not expected to know the valid dotted names for `asadmin list` and `get` commands. The `list` command lets you inspect available monitorable objects, while the `get` command used with a wildcard parameter allows you to inspect 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 (`*`) will get as its result the current node's immediate children. For example, `list --monitor server` will list all immediate children belonging to the `server` node.
- Any `list` command that has a dotted name followed by a wildcard of the form `.*` will get as its result a hierarchical tree of children nodes from the current node. For example, `list --monitor server.applications.*` will list 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 *`** will get 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 `*` will get as its result the set of attributes and their values belonging to the current node whose name is sought to be matched.

For more information, read "Expected Output for list and get Commands at All Levels" on page 189.

Examples of the list and get Commands

This section contains the following topics:

- Examples for the list --monitor Command
- Examples for the get --monitor Command
- Petstore Example

Examples for the list --monitor 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 sub-components 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 189.

Example

```
asadmin> list --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.applications
server.http-service
server.thread-pools
server.transaction-service
```

You can also list applications that are currently monitored in the specified server instance. This can be useful when particular monitoring statistics are sought from an application using the get command.

Example

```
asadmin> list --monitor server.applications
```

Returns:

```
server.applications.MEjbApp
server.applications._ebj_container_timer_app
server.applications.myApp
```

For a more comprehensive example, see "Petstore Example" on page 185.

Examples for the get --monitor Command

This 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 189 for more information on the use of the get command.

Example

Attempt to get all attributes from a subsystem for a specific object:

```
asadmin> get --monitor server.jvm.*
```

Returns:

```
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

Attempt to get all attributes from a J2EE application:

```
asadmin> get --monitor server.applications.converter.*
```

Returns:

```
No matches resulted from the wildcard expression.
CLI137 Command get failed.
```

There are no monitorable attributes exposed at the J2EE-application level, therefore the command notifies you of this.

Example

Attempt to get a specific attribute from a subsystem:

```
asadmin> get --monitor server.transaction-service.activecount-count
```

Returns:

```
server.transaction-service.activecount-count = 0
```

Example

Attempt to get an unknown attribute from within a subsystem attribute:

```
asadmin> get --monitor server.transaction-service.badname
```

Returns:

```
No such attribute found from reflecting the corresponding Stats
interface: [badname]
CLI137 Command get failed.
```

Petstore Example

The following example illustrates how you might use the `asadmin` tool 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 Sun Java™ System Application Server Platform Edition 8. 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=4848
   ```

3. List monitorable components for instance `server`:

   ```
   asadmin>list --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. You can also get a specific statistic, such as execution time.

```
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 15-13 Top Level

Command	Dotted Name	Output

`list -m`	`server`	`server.applications`
		`server.thread-pools`
		`server.resources`
		`server.http-service`
		`server.transaction-service`
		`server.orb.connection-managers`
		`server.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnections`
		`server.jvm`

`list -m`	`server.*`	Hierarchy of child nodes below this node.
`get -m`	`server.*`	No output, but message saying there are no attributes at this node.

Table 15-14 Applications Level

Command	Dotted Name	Output

`list -m`	`server.applications`
	or
	`*applications`
		`appl1`
		`app2`
		`web-module1_war`
		`ejb-module2_jar`
		`...`

`list -m`	`server.applications.*`
	or
	`*applications.*`
		Hierarchy of child nodes below this node.

`get -m`	`server.applications.*`
	or
	`*applications.*`
		No output, but message saying there are no attributes at this node.

Table 15-15 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. Not applicable is only a standalone module is deployed.
		`ejb-module1_jar`
		`web-module2_war`
		`ejb-module3_jar`
		`web-module3_war`
		`...`

`list -m`	`server.applications.app1.*`
	or
	`*app1.*`
		Hierarchy of child nodes below this node.

`get -m`	`server.applications.app1.*`
	or
	`*app1.*`
		No output, but 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`
		`bean1`
		`bean2`
		`bean3`
		`...`

`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, but 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-pool`
		`bean-cache`
		`bean-method`

Table 15-15 Applications - Enterprise Applications and Standalone Modules

Command	Dotted Name	Output
`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.

`get -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.
		The following attributes and their associated values:
		`CreateCount_Count`
		`CreateCount_Description`
		`CreateCount_LastSampleTime`
		`CreateCount_Name`
		`CreateCount_StartTime`
		`CreateCount_Unit`
		`MethodReadyCount_Current`
		`MethodReadyCount_Description`
		`MethodReadyCount_HighWaterMark`
		`MethodReadyCount_LastSampleTime`
		`MethodReadyCount_LowWaterMark`
		`MethodReadyCount_Name`
		`MethodReadyCount_StartTime`
		`MethodReadyCount_Unit`
		`RemoveCount_Count`
		`RemoveCount_Description`
		`RemoveCount_LastSampleTime`
		`RemoveCount_Name`
		`RemoveCount_StartTime`
		`Attribute RemoveCount_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 “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-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 15-3.
`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.”

Table 15-15 Applications - Enterprise Applications and Standalone Modules

Command	Dotted Name	Output
`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) will not appear.*	List of attributes and values corresponding to EJB Cache attributes as described in Table 15-4.
`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 15-2.
`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, but 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, but 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 15-5.

Table 15-16 HTTP-Service Level

Command	Dotted Name	Output
`list -m`	`server.http-service`	List of virtual servers.

Table 15-16 HTTP-Service Level

Command	Dotted Name	Output
`get -m`	`server.http-service.*`	No output, but 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, but message saying there are no attributes at this node.
`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 as described in Table 15-6.

Table 15-17 Thread-Pools Level

Command	Dotted Name	Output
`list -m`	`server.thread-pools`	List of thread-pool names.
`get -m`	`server.thread-pools.*`	No output, but 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 as described in Table 15-10.

Table 15-18 Resources Level

Command	Dotted Name	Output
`list -m`	`server.resources`	List of pool names.
`get -m`	`server.resources.*`	No output, but message saying there are no attributes at this node.

Table 15-18 Resources Level

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 as described in Table 15-7.

Table 15-19 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 as described in Table 15-11.

Table 15-20 ORB Level

Command	Dotted Name	Output
`list -m`	`server.orb`	`server-orb.connection-managers`
`get -m`	`server.orb.*`	No output, but 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, but 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 15-20 ORB Level

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 as described in Table 15-9.

Table 15-21 JVM Level

Command	Dotted Name	Output
`list -m`	`server.jvm`	No attributes, but a message saying "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 as described in Table 15-12.

Chapter 16

Java Virtual Machine and Advanced Settings

This chapter expains how to configure the Java Virtual Machine (JVM™) and other advanced settings. It contains the following sections:

- Admin Console Tasks for JVM™ Settings
- Admin Console Tasks for Advanced Settings

Admin Console Tasks for JVM™ Settings

- Configuring the JVM General Settings
- Configuring the JVM Classpath Settings
- Configuring the JVM Options
- Disabling the Security Manager
- Configuring the JVM Profiler Settings

Configuring the JVM General Settings

The Java Virtual Machine (JVM) is included in the Java 2 Standard Edition (J2SE™) software, which is required by the Application Server. Because incorrect JVM settings will prevent the server from running, you should take care when changing these settings.

To configure the general settings of the JVM used by the Application Server:

1. In tree component, select the Application Server node.

2. Click the JVM Settings tab.

3. By default, the General link located below the tabs is already selected.

4. On the JVM General Settings page, you may specify the following:

   a. In the Java Home field, enter the name of the installation directory of the Java 2 Standard Edition (J2SE) software.

      The Application Server relies on the J2SE software. To verify that the J2SE version you specify is supported in this release, refer to the Release Notes. (See the link in the Further Information section.)

      Note: If you enter a non-existent directory name or the installation directory name of an unsupported version of the J2SE software, then the Application Server will not start.

   b. In the Javac field, type the command-line options for the Java programming language compiler.

      The Application Server runs the compiler when EJB components are deployed.

   c. To set up debugging with the JPDA (Java Platform Debugger Architecture), you select the Debug Enabled checkbox and specify options in the Debug Options field.

      JPDA is used by application developers. For more information, see the Debugging J2EE Applications chapter of the Sun Java™ System Application Server Platform Edition 8 Developer's Guide. (For a link to the guide, see Further Information.)

   d. In the RMI Compile Options field, type the command-line options for the rmic compiler.

      The Application Server runs the rmic compiler when EJB components are deployed.

   e. In the Bytecode Preprocessor field, type 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 might require entries in the Bytecode Preprocessor field. Profilers generate information used to analyze server performance. For more information about profiling, see the Debugging J2EE Applications chapter of the Sun Java™ System Application Server Platform Edition 8 Developer's Guide.

5. Click Save.
6. Restart the server.

Configuring the JVM Classpath Settings

The classpath is the list of JAR files that the Java runtime environment searches for classes and other resource files.

To configure the Application Server's JVM classpath:

1. In tree component, select the Application Server node.
2. Click the JVM Settings tab.
3. Select the Path Settings link below the tabs.
4. On the JVM Classpath Settings page, you may specify the following:
   a. In the Environment Classpath checkbox, retain the default selection to ignore the CLASSPATH environment variable.

      The CLASSPATH environment variable is convenient for basic tutorials in programming, but is not recommended for enterprise environments.

   b. To view the Application Server's classpath, examine the read-only contents of the Server Classpath field.

   c. To insert a JAR file into the beginning of the server's classpath, enter the full path name of the file in the Classpath Prefix field.

   d. To add a JAR file to the end of the server's classpath, enter the full path name of the file in the Classpath Suffix field.

      For example, you would specify the JAR file of a database driver. See Integrating a JDBC Driver.

   e. In the Native Library Path Prefix and Suffix fields, you may prepend or append entries to the native library path.

      The native library path is a concatenation of the server's relative path for its native shared libraries, the standard JRE native library path, the shell environment setting (LD_LIBRARY_PATH on UNIX), and any path specified on the JVM Profiler Settings page.

5. Click Save.
6. Restart the server.

Configuring the JVM Options

On the JVM Options page, you may specify the options of the Java application launcher (java tool) that runs the Application Server. The -D options designate properties that are specific to the Application Server.

To configure the JVM options:

1. In tree component, select the Application Server node.
2. Click the JVM Settings tab.
3. Select the JVM Options link below the tabs.
4. On the JVM Options page, to modify an option you edit the Value field.
5. To add an option:
   a. Click Add JVM Option.
   b. In the blank row that appears, type the information in the Value field.
6. To remove an option:
   a. Select the checkbox next to the option.
   b. Click Delete.
7. Click Save.
8. Restart the server.

For more information about about JVM options, see:

- http://java.sun.com/j2se/1.4.2/docs/tooldocs/tools.html
- http://java.sun.com/docs/hotspot/VMOptions.html

Disabling the Security Manager

Disabling the Application Server's security manager may improve performance for some types of applications. The J2EE authorization and authentication features will still work even if the security manager has been disabled. You may disable the security manager in a development environment, but you should not disable it in a production environment.

To disable the security manager:

1. Go to the JVM Options page of the Admin Console.

   For instructions, see Configuring the JVM Options.

2. On the JVM Options page, remove this option:

   ```
   -Djava.security.policy
   ```

3. Click Save.
4. Restart the server.

Configuring the JVM Profiler Settings

A profiler tool generates data that is used to analyze performance and identify potential bottlenecks.

To configure profiler settings for the Application Server:

1. In tree component, select the Application Server node.
2. Click the JVM Settings tab.
3. Select the Profiler link below the tabs.
4. The information that you specify on the JVM Profiler Settings page depends on which profiler product you're using.

   For examples and instructions, see the Debugging J2EE Applications chapter of the Sun Java™ System Application Server Platform Edition 8 Developer's Guide. (For a link to the guide, see Further Information.)

5. Click Save.
6. Restart the server.

Admin Console Tasks for Advanced Settings

- Setting the Admin Session Timeout and the Locale

Setting the Admin Session Timeout and the Locale

1. In the tree component, select the Application Server node.

2. Select the Advanced tab.
3. On the Advanced Configuration page, you may do the following:
    a. In the Admin Session Timeout field, type an integer for the number of minutes that an Admin Console session remains active.

       When the timeout expires, you have to log in again to start a new session. If the interval is zero, then the session will not time out. In a production environment where you want to restrict administrative access, you should not set the timeout to zero.

    b. Typically, you will leave the Locale field blank because the Sun Java™ System Application Server Platform Edition 8 will use the default locale of the host.

       A locale is an identifier that specifies a particular combination of language and region. For example, the locale for American English is en_US, and for Japanese it is ja_JP. In order to use a non-English locale, the Application Server must be localized, a process that includes translating English into another language.
4. Click Save.
5. Restart the server.