Sun Java™ System

Application Server Enterprise Edition 8.1 Administration Guide

2005Q1

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

Part No: 817-6088

Contents

Preface

This guide describes how to configure and administer the Application Server. This preface contains information about the following topics:

Who Should Use This Book

This *Administration 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

Before You Read This Book

Application Server is a component of Sun Java™ Enterprise System, a software infrastructure that supports enterprise applications distributed across a network or Internet environment. You should be familiar with the documentation provided with Sun Java Enterprise System, which can be accessed online at `http://docs.sun.com/coll/entsys.05q1#hic`.

How This Book 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.

Conventions Used in This Book

The tables in this section describe the conventions used in this book.

Typographic Conventions

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

Table 1 Typographic Conventions

Typeface	Meaning	Examples
AaBbCc123 (Monospace)	API and language elements, HTML tags, web site URLs, command names, file names, directory path names, onscreen computer output, sample code.	Edit `your.login` file. Use `ls -a` to list all files. `% You have mail.`
AaBbCc123 (Monospace bold)	What you type, when contrasted with onscreen computer output.	`% su` `Password:`

Table 1 Typographic Conventions *(Continued)*

Typeface	Meaning	Examples
AaBbCc123 (Italic)	Book titles, new terms, words to be emphasized. A placeholder in a command or path name to be replaced with a real name or value.	Read Chapter 6 in the *User's Guide*. These are called *class* options. Do *not* save the file. The file is located in the *install-dir*`/bin` directory.

Symbols

The following table describes the symbol conventions used in this book.

Table 2 Symbol Conventions

Symbol	Description	Example	Meaning
`[ ]`	Contains optional command options.	`ls [-l]`	The `-l` option is not required.
`{ | }`	Contains a set of choices for a required command option.	`-d {y|n}`	The `-d` option requires that you use either the `y` argument or the `n` argument.
-	Joins simultaneous multiple keystrokes.	Control-A	Press the Control key while you press the A key.
+	Joins consecutive multiple keystrokes.	Ctrl+A+N	Press the Control key, release it, and then press the subsequent keys.
>	Indicates menu item selection in a graphical user interface.	File > New > Templates	From the File menu, choose New. From the New submenu, choose Templates.

Default Paths and File Names

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

Table 3 Default Paths and File Names

Term	Description

install_dir	By default, the Application Server installation directory is located here:
	• Sun Java Enterprise System installations on the Solaris™ platform:
	`/opt/SUNWappserver/appserver`
	• Sun Java Enterprise System installations on the Linux platform:
	`/opt/sun/appserver/`
	• Other Solaris and Linux installations, non-root user:
	user's home directory`/SUNWappserver`
	• Other Solaris and Linux installations, root user:
	`/opt/SUNWappserver`
	• Windows, all installations:
	SystemDrive`:\Sun\AppServer`

domain_root_dir	By default, the directory containing all domains is located here:
	• Sun Java Enterprise System installations on the Solaris platform:
	`/var/opt/SUNWappserver/domains/`
	• Sun Java Enterprise System installations on the Linux platform:
	`/var/opt/sun/appserver/domains/`
	• All other installations:
	install_dir`/domains/`

domain_dir	By default, each domain directory is located here: *domain_root_dir*`/`*domain_dir* In configuration files, you might see *domain_dir* represented as follows: `${com.sun.aas.instanceRoot}`
instance_dir	By default, each instance directory is located here: *domain_dir*`/`*instance_dir*

Shell Prompts

The following table describes the shell prompts used in this book.

Table 4 Shell Prompts

Shell	Prompt
C shell on UNIX or Linux	*machine-name*%
C shell superuser on UNIX or Linux	*machine-name*#
Bourne shell and Korn shell on UNIX or Linux	$
Bourne shell and Korn shell superuser on UNIX or Linux	#
Windows command line	C:\

Related Documentation

The http://docs.sun.com[SM] web site enables you to access Sun technical documentation online. You can browse the archive or search for a specific book title or subject.

You can find a directory of URLs for the official specifications at *install_dir*/docs/index.htm. Additionally, the following resources might be useful.

General J2EE Information:

The J2EE 1.4 Tutorial:

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

The J2EE Blueprints:

http://java.sun.com/reference/blueprints/index.html

Core J2EE Patterns: Best Practices and Design Strategies by Deepak Alur, John Crupi, & Dan Malks, Prentice Hall Publishing

Java Security, by Scott Oaks, O'Reilly Publishing

Programming with Servlets and JSP files:

Java Servlet Programming, by Jason Hunter, O'Reilly Publishing

Java Threads, 2nd Edition, by Scott Oaks & Henry Wong, O'Reilly Publishing

Programming with EJB components:

Enterprise JavaBeans, by Richard Monson-Haefel, O'Reilly Publishing

Programming with JDBC:

Database Programming with JDBC and Java, by George Reese, O'Reilly Publishing

JDBC Database Access With Java: A Tutorial and Annotated Reference (Java Series), by Graham Hamilton, Rick Cattell, & Maydene Fisher

Books in This Documentation Set

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

The following table summarizes the books included in the Application Server core documentation set.

Table 5 Books in This Documentation Set

Book Title	Description
Release Notes	Late-breaking information about the software and the documentation. Includes a comprehensive, table-based summary of the supported hardware, operating system, JDK, and JDBC/RDBMS.
Quick Start Guide	How to get started with theSun Java SystemApplication Server product.
Installation Guide	Installing the Application Server software and its components.
Deployment Planning Guide	Evaluating your system needs and enterprise to ensure that you deploy Sun Java System Application Server in a manner that best suits your site. General issues and concerns that you must be aware of when deploying an application server are also discussed.
Developer's Guide	Creating and implementing Java™ 2 Platform, Enterprise Edition (J2EE™ platform) applications intended to run on the Application Server that follow the open Java standards model for J2EE components and APIs. Includes general information about developer tools, security, assembly, deployment, debugging, and creating lifecycle modules.
J2EE 1.4 Tutorial	Using J2EE 1.4 platform technologies and APIs to develop J2EE applications and deploying the applications on the Sun Java System Application Server.
Administration Guide	Configuring, managing, and deploying the Application Server subsystems and components from the Administration Console.
High Availability Administration Guide	Configuring and managing the Sun Java System Application Server high availability featuers.
Administration Reference	Editing the Sun Java System Application Server configuration file, `domain.xml`.

Table 5 Books in This Documentation Set *(Continued)*

Book Title	Description
Upgrade and Migration Guide	Migrating your applications to the new Sun Java System Application Server programming model, specifically from Application Server 6.x and 7. This guide also describes differences between adjacent product releases and configuration options that can result in incompatibility with the product specifications.
Performance Tuning Guide	Tuning the Sun Java System Application Server to improve performance.
Troubleshooting Guide	Solving Sun Java System Application Server problems.
Error Message Reference	Solving Sun Java System Application Server error messages.
Reference Manual	Utility commands available with the Sun Java System Application Server; written in manpage style. Includes the `asadmin` command line interface.

Other Server Documentation

For other server documentation, go to the following:

- Message Queue documentation
  http://docs.sun.com/db?p=prod/s1.s1msgqu
- Directory Server documentation
  http://docs.sun.com/coll/DirectoryServer_04q2
- Web Server documentation
  http://docs.sun.com/coll/S1_websvr61_en

Accessing Sun Resources Online

For product downloads, professional services, patches and support, and additional developer information, go to the following:

- Download Center
  http://wwws.sun.com/software/download/
- Professional Services
  http://www.sun.com/service/sunps/sunone/index.html
- Sun Enterprise Services, Solaris Patches, and Support
  http://sunsolve.sun.com/
- Developer Information
  http://developers.sun.com/prodtech/index.html

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in the product documentation, go to http://www.sun.com/service/contacting.

Related Third-Party Web Site References

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

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions.

To share your comments, go to http://docs.sun.com and click Send Comments. In the online form, provide the document title and part number. The part number is a seven-digit or nine-digit number that can be found on the title page of the book or at the top of the document. For example, the title of this book is *Sun Java System Application Server 2005Q1 Administration Guide*, and the part number is 817-6088.

Chapter 1

Getting Started

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

About the Sun Java System Application Server

What is the Application Server?

The Application Server 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 Application Server Enterprise Edition provides advanced clustering and failover technologies. These features enable you to run scalable and highly available J2EE applications.

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

  Horizontal scaling is acheived by adding Application Server instances to a cluster, thereby increasing the capacity of the system. It is possible to add Application Server instances to a cluster without disrupting service. The HTTP, RMI/IIOP, and JMS load balancing systems distribute requests to healthy Application Server instances in the cluster.

- **High Availability -** Availability allows for failover protection of Application Server instances in a cluster. If one application server instance goes down, another Application Server instance takes over the sessions that were assigned to the unavailable server. Session information is stored in the high-availability database (HADB). HADB supports the persistence of HTTP sessions and stateful session beans.

Application Server Architecture

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

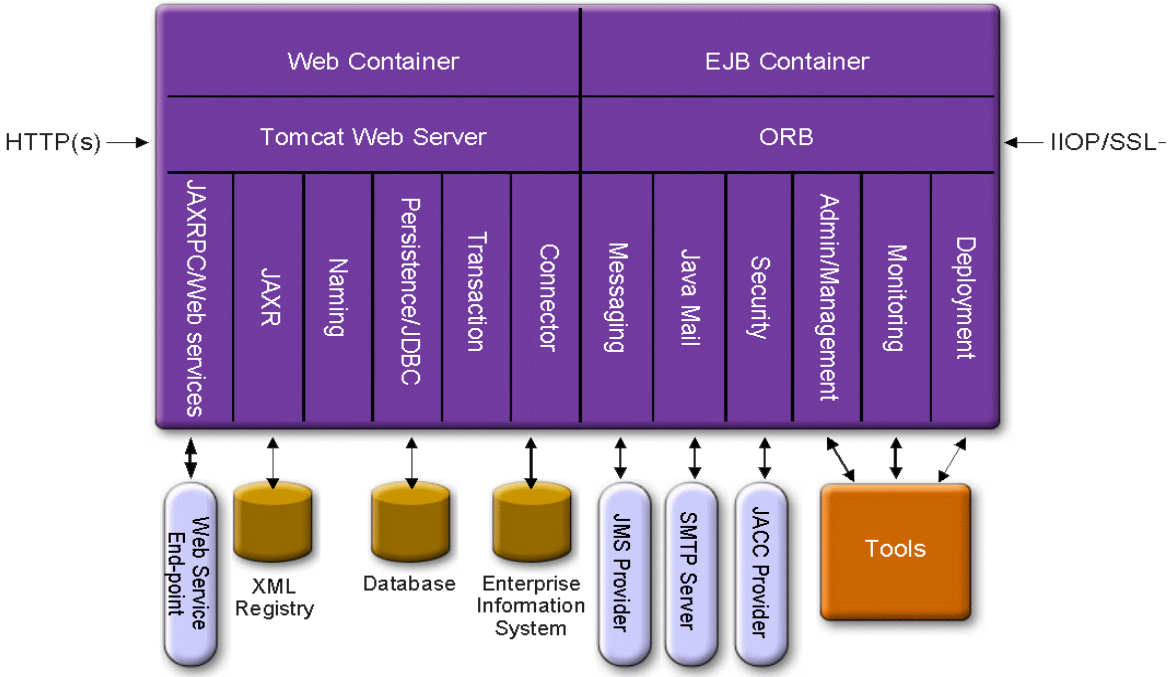


Figure 1-1 Application Server Architecture

- **Containers** - A container is a runtime environment that provides services such as security and transaction management to J2EE components. Figure 1-1 shows the two types of J2EE containers: Web and EJB. Web components, such as JSP pages and servlets, run within the Web container. Enterprise 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 of a specific port number.

- **Web Services** - On the J2EE platform, it is possible to deploy a Web application that provides a Web service implemented by Java API for XML-Based RPC (JAX-RPC). A J2EE application or component can also be a client to other Web services. Applications access XML registries through the Java API for XML Registries (JAXR).
- **Services for Applications** - The J2EE platform was designed so that the containers provide services for applications. Figure 1-1 shows the following services:
  - Naming - A naming and directory service binds objects to names. A J2EE application locates an object by looking up its JNDI name. JNDI stands for the Java Naming and Directory Interface API.
  - Security - The Java Authorization Contract for Containers (JACC) is a set of security contracts defined for the J2EE containers. Based on the client's identity, the containers restrict access to the container's resources and services.
- **Transaction management** - A transaction is an indivisible unit of work. For example, 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. One of the responsibilities of an administrator is resource configuration. The J2EE platform enables access to external systems through the following APIs and components:

- **JDBC** - A database management system (DBMS) provides facilities for storing, organizing, and retrieving data. Most business applications store data in relational databases, which applications access via the JDBC API. The information in databases is often described as persistent because it is saved on disk and exists after the application ends. The Application Server bundle includes the PointBase DBMS.
- **Messaging** - Messaging is a method of communication between software components or applications. A messaging client sends messages to, and receives messages from, any other client. Applications access the messaging provider through the Java Messaging Service (JMS) API. The Application Server includes a JMS provider.

- **Connector** - The J2EE Connector architecture enables integration between J2EE applications and existing Enterprise Information Systems (EIS). An application accesses an EIS through a portable J2EE component called a connector or resource adapter.
- **JavaMail** - Through the JavaMail API, applications connect to an SMTP server in order to send and receive email.
- **Server Administration** - The lower right-hand corner of Figure 1-1 shows some of the tasks performed by the administrator of the Application Server. For example, an administrator deploys (installs) applications and monitors the server's performance. These tasks are performed with the administration tools provided by the Application Server.
- Tools for Administration
- The Application Server includes three administrative tools:
  - Admin Console
  - asadmin Utility
  - Application Server Management Extension (AMX)

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 to use the Admin Console.

When the Application Server was installed, you chose a port number for the server, or used the default port of 4849. You also specified a user name and master password.

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

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

For example:

```
https://kindness.sun.com:4949
```

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

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

The installation program creates the default administrative domain (named `domain1`) with the default port number 4849, as well as an instance separate from the domain administration server (DAS). After installation, additional administration domains can be created. Each domain has its own domain administration server, which has a unique port number. When specifying the URL for the Admin Console, be sure to use the port number for the domain to be administered.

If your configuration includes remote server instances, create node agents to manage and facilitate remote server instances. It is the responsibility of the node agent to create, start, stop, and delete a server instance. Use the command line interface (CLI) commands to set up node agents.

asadmin Utility

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

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

To start the `asadmin` utility:

```
$ asadmin
```

To list the commands available within `asadmin`:

```
asadmin> help
```

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

```
$ asadmin help
```

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

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

The `asadmin help` information for a given command displays the Unix man page of the command. These man pages are also available in HTML format.

Application Server Management Extension (AMX)

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

For more information on using the Application Server Management Extension, see the Using the Java Management Extensions (JMX) API chapter in the *Sun Java System Application Server Developers Guide.*

Application Server Configuration

Configuring the Application Server

Application Server domains are logical or physical units created to help the administrator manage a system configuration. A domain is broken down into smaller units including instances and node agents. A server instance is a single Java Virtual Machine (JVM) that runs the Application Server on a single physical machine. Each domain has one or more instances. A domain must also have at least one associated node agent for the instance to function properly. Domains can be grouped together to create a cluster. Clusters allow the administrator to manage groups of hardware and software.

Configuring 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 the configuration is changed for one domain, the configurations of other domains are not affected.

Each Administration console session allows you to configure and manage the domain. If you have created multiple domains, you must start an additional Administation Console session to manage each domain. Each domain has its own Domain Administration Server (DAS), with a unique port number. Each administrative domain can have multiple application server instances. However, an application server instance can belong to just one domain. When the Application Server is installed, an administrative domain named `domain1` is automatically created.

Creating a Domain

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

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

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

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

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

install_dir`/domains/mydomain`

To create the domain's directory in another location, 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 the Domain

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

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

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

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

On Windows, to start the default domain:

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

Restarting the Server or Domain

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

Stopping the Domain

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

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

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

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

To stop the domain using the Admin Console:

- In the tree component, select server (Admin Server) under the Standalone Instances node.
- On the General Information page, click Stop Server.

On Windows, to stop the default domain:

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

Recreating the Domain Administration Server

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

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

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

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

Steps for DAS Migration

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

1. Setup the third machine by installing the same bits of the application server on the third machine, as is installed on the first machine.

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

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

     `./` *`bundle_filename`* `-console`

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

   - De-select the option to install default domain.

     Restoration of backed up domains is only supported on two machines with same architecture and **exactly** the same installation paths (i.e., use same *install_dir* on both machines).

2. Copy the backup ZIP file from the first machine into *`install_dir`*`/domains` on the third machine. You may also FTP the file.
3. Execute `asadmin restore-domain` command to restore the ZIP file onto the third machine:

   ```
   asadmin restore-domain --filename
   install_dir/domains/sjsas_backup_v00001.zip domain1
   ```

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

4. Change *install_dir*/domains/domain1/generated/tmp directory permissions on the third machine to match the permissions of the same directory on first machine.

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

   For example:

   ```
   chmod 700 install_root/domains/domain1/generated/tmp
   ```

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

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

6. Update the *install_root*/domains/*domain1*/config/domain.xml on the third machine.

   For example:

   Search for *machine1* and replace it with *machine3*. So, you can change:

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

   to:

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

7. Change:

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

   to:

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

8. Start the restored domain on machine3:

   ```
   asadmin start-domain --user admin_user --password admin_password
   domain1
   ```

9. Change the DAS host values for properties under nodeagent on machine2:

10. Change agent.das.host property value in *install_dir*/nodeagents/*nodeagent*/agent/config/das.properties in machine2.

11. Restart nodeagent on machine2.

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

Application Server Instances

About Application Server Instances

Sun Java System Application Server creates one application server instance, called server at the time of installation. You can delete the server instance and create a new instance with a different name if you prefer.

Each Sun Java System Application Server instance has its own J2EE configuration, J2EE resources, application deployment areas, and server configuration settings. Changes to one application server instance have no effect on other application server instances. You can have many application server instances within one administrative domain.

For many users, one application server instance meets their needs. However, depending upon your environment, you might want to create one or more additional application server instances. For example, in a development environment you can use different application server instances to test different Sun Java System Application Server configurations, or to compare and test different application deployments. Because you can easily add or delete an application server instance, you can use them to create temporary “sandbox” areas to experiment with while developing.

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

In operational deployments, for many purposes you can use virtual servers instead of multiple application server instances. However, if virtual servers do not meet your needs, you can also use multiple application server instances.

A Sun Java System Application Server instance is not started automatically. Once you start an instance, the instance runs until you stop it. When you stop an application server instance, it stops accepting new connections, then waits for all outstanding connections to complete. If your machine crashes or is taken offline, the server quits and any requests it was servicing may be lost.

Defining Application Server Instances

Application server instances form the basis of an application deployment. Each instance belongs to a single domain and has its own directory structure, configuration, and deployed applications. Each server instance also includes the J2EE platform web and EJB containers. Every new server instance must contain a reference to a node agent name defining the machine on which the instance will reside.

There are three typres of server instances that can be created. Each server instance can only be of one type:

- In the **standalone server** instance the configuration is not shared by any other server instance or clusters.
- In the **shared server instance** the configuration is shared with other instances or clusters.
- In the **clustered server instance** the configuration is shared with other instances in the cluster.

Figure 1-2 shows an application server instance in detail. The application server instance is a building block in the clustering, load balancing, and session persistence features of the Application Server Enterprise Edition.

- **Defining and Using Clusters** - A cluster is a group of server instances sharing the same set of applications, resources, and configuration information. A server instance can belong to only one cluster. Among other things, the cluster is used to facilitate load balancing, through ditribution of a load across multiple machines, and high availability, through instance level failover.

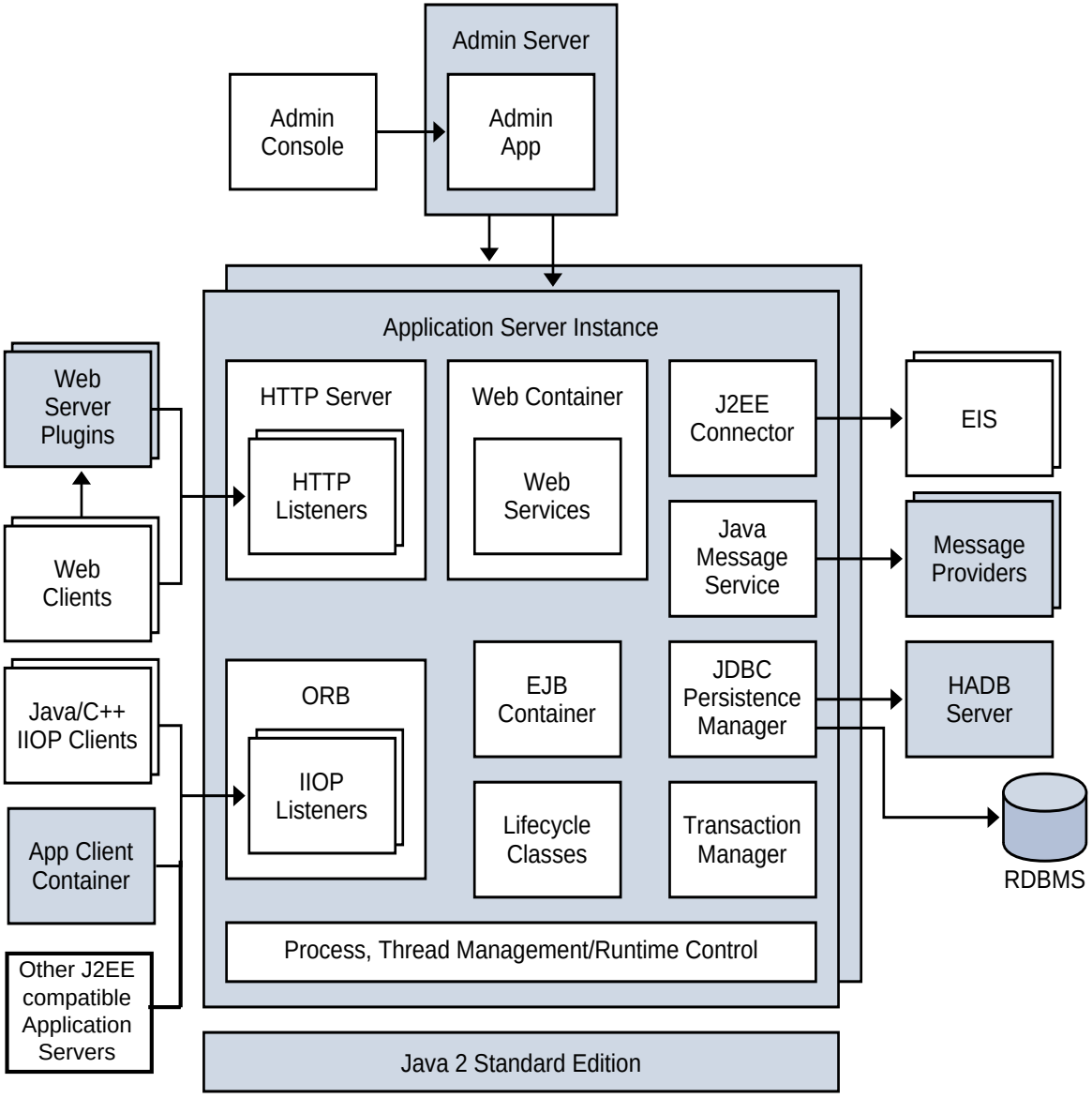


Figure 1-2 Sun Java System Application Server Instance

About Stand-Alone Instances

A Sun Java System Application Server instance is not started automatically. Once you start an instance, the instance runs until you stop it. When you stop an application server instance, it stops accepting new connections, then waits for all outstanding connections to complete. If your machine crashes or is taken offline, the server quits and any requests it was servicing may be lost.

Viewing General Server Information

From the General Tab you can perform the following tasks:

- Click Start Instance to start the instance.
- Click Stop Instance to stop the instance.
- Click View Log Files to open the server log viewer.
- Click Rotate Log File to rotate the log file for the instance. This action schedules the log file for rotation. The actual rotation takes place the next time an entry is written to the log file. The rotation happens immediately for the default server (the DAS) but is delayed for other stand-alone server.
- Click JNDI Browsing to browse the JNDI tree for a running instance.
- Click Recover Transactions to recover incomplete transactions.

In addition you can select the following tabs to perform thse additional tasks:

- Applications Tab: deploy a selected application.
- Resources Tab: manage a selected resource.
- Properties Tab: configure instance specific properties.
- Monitor Tab: view monitoring data for JVM, Server, Thread Pools, HTTP Service, and Transaction Service.
- Advanced Tab: set general properties for deploying applications.

Managing Applications

From the Applications Tab, you can enable, disable, and deploy selected applications associated with the instance.

To deploy an application:

1. Select the checkbox for the desired application.
2. From the Deploy drop down menu, select the type application module you want to deploy:
   - Enterprise Application: a J2EE application in an EAR(Enterprise Application Archive) file or directory.
   - Web Application: a collection of Web resources such as JavaServer Pages (JSPs), servlets, and HTML pages that are packaged in a WAR (Web Application Archive) file or directory.
   - EJB Module: one or more Enterprise JavaBeans (EJBs) contained in an EJB JAR (Java Archive) file or directory.
   - Connector Module: connects to an Enterprise Information System (EIS) and is packaged in a RAR(Resource Adapter Archive) file or directory.
   - Lifecycle Module: performs tasks when it is triggered by one or more events in the server's lifecycle.
   - App Client Module: also called a J2EE application client JAR file, contains the server-side routines for the client.

Managing Resources

From the Recources Tab, you can enable, disable, and create a new resource type to associate with the instance.

To create a new resource type:

1. Select the checkbox for the desired resource.
2. From the New drop down menu, select the resource type you want to create and associate with that instance:
   - JDBC: provide applications with a means of connecting to a database.
   - Persistence Manager: required for applications with container-managed persistence beans (needed for backward compatibility).

- JMS Connection Factory: objects that allow an application to create other JMS objects programmatically.
- JMS Destination: represents a mail session in the JavaMail API, which provides a platform-independent and protocol-independent framework to build mail and messaging applications.
- JavaMail: provides a platform-independent and protocol-independent framework to build mail and messaging applications.
- Custom: represents non-standard resources with a defined JNDI subcontext, resource type, and factory class.
- External: enables an application to locate an external resource object in a Lightweight Directory Access Protocol (LDAP) repository.
- Connector: a program object that provides an application with a connection to an enterprise information systems (EIS).
- Admin Object: configures a JSR-160 compliant remote JMX connector

Administration Server Advanced Settings

The Administration Server Advanced settings allow you to set general properties for deploying applications. These properties enable you to ensure and monitor that changes to deployed applications are detected and the modified classes reloaded.

Setting Applications Configurations

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.

Dynamic reloading is intended for development environments. It is incompatible with session persistence, a production environment feature. Do not enable session persistence if dynamic deployment is enabled.

NOTE Dynamic reloading is only available for the default server instance.

To configure dynamic reloading from the Applications Configuration page, configure the following:

- Reload: Enable or disable dynamic reloading with the Enabled checkbox.
- Reload Poll Interval: Specify how often the server checks for changes in the deployed applications.
- Admin Session Timeout: Specify the amount of time before the Admin Session times out and you have to log in again.

Setting the 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, copy a file named hello.war to the *install_dir*/domains/domain1/autodeploy directory. To undeploy the application, remove the hello.war file from the autodeploy directory.

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

NOTE Auto deploy is only available for the default server instance.

To configure the auto deploy settings from the the Applications Configuration page:

1. Enable or disable auto deploy by selecting or deselecting the Enabled checkbox.
2. In the Auto Deploy Poll Interval field, 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.
3. 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.

   The default is a directory called autodeploy in the server instance's root directory. By default a variable is used to eliminate the need to manually change the directory for multiple server instances.
4. To run the verifier before deployment, select the Verifier Enabled checkbox. The verifier examines the structure and content of the file. Verification of large applications is often time-consuming.

5. To precompile JSP pages, select the JSPs checkbox. If you do not select this checkbox, the JSP pages are compiled at runtime when they are first accessed. Because compilation is often time-consuming, in a production environment select this checkbox.

Setting Additional Properties

Click the Add Property button to specify additional settings.

Setting Domain Attributes

The following domain attributes properties are available.

Table 1-1 Domain Attributes values

Property	Definition
com.sun.aas.installRoot	Directory where the application server is installed.
com.sun.aas.instanceRoot	Top level directory for a server instance.
com.sun.aas.hostName	Name of the host (machine).
com.sun.aas.javaRoot	.J2SE installation directory.
com.sun.aas.imqLib	Library directory of the Sun Java System Message Queue.
com.sun.aas.configName	Name of the configuration being used by a server instance.
com.sun.aas.instanceName	Name of the server instance. This property is not available for the default-config but can be used for customized configurations.
com.sun.aas.clusterName	Name of the cluster. This property is only set on the clustered server instances. This property is not available for the default-config but can be used for customized configurations.
com.sun.aas.domainName	Name of the domain. This property is not available for the default-config but can be used for customized configurations.

Instance Specific Configuration Properties

The instance specific Configuration Properties override the values for this instance.

NOTE The default values are defined in the configuration bound to the instance.

To revert the value back to the default value:

1. Remove the override value.
2. Click Save.

If no override value is set, the default value is used.

Adding or Deleting Instance Properties

To add properties:

- Click the Add Property button to specify additional settings.

The following property attribute name / value pairs for configuring the resource are available:

Table 1-2 Property Attribute Name / Value Pairs

Property	Definition
HTTP_LISTENER_PORT	This port is used to listen for HTTP requests. This property specifies the port number for http-listener-1. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.
HTTP_SSL_LISTENER_PORT	This port is used to listen for HTTPS requests. This property specifies the port number for http-listener-2. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.
IIOP_LISTENER_PORT	This property specifies which ORB listener port for IIOP connections orb-listener-1 listens on.
IIOP_SSL_LISTENER_PORT	This port is used for secure IIOP connections.
JMX_SYSTEM_CONNECTOR_PORT	This property specifies the port number on which the JMX connector listens. Valid values are 1-65535. On UNIX, creating sockets that listen on ports 1-1024 requires superuser privileges.
IIOP_SSL_MUTUALAUTH_PORT	This property specifies which ORB listener port for IIOP connections the IIOP listener called SSL_MUTUALAUTH listens on.

To delete properties:

1. Click the for the property you wish to delete.
2. Click the Delete Property button.

Creating an Instance

To create an instance:

1. In the tree component, select the Standalone Instances node.
2. On the Stand Alone Server Instances page, click New.
3. In the Name field, identify a unique name for the new instance.
4. Select a Node Agent.

   The node agent must be started using the `asadmin start-node-agent` command on the node agent's host machine so that the server instance you are creating can be associated with that node agent.
5. Select a desired configuration.
   - Reference an existing configuration. No new configuration is added.
   - Make a copy of an existing configuration. A new configuration is added when the server instance or cluster is added.
6. By default, new instances are created with configurations copied from the default-config configuration. To copy from a different configuration, specify it when creating the new instance.
7. For a server instance, the new configuration is named `instance_name-config`.

   The configuration default-config is the default configuration that acts as a template for creating standalone server instance. No unclustered server instances or clusters are allowed to refer to the default-config configuration; it can only be copied to create new configurations. Edit the default configuration to ensure that new configurations copied from it have the correct initial settings.

Equivalent `asadmin` command: `create-instance`.

Starting an Instance

To start an instance, perform these steps:

1. In the tree component, expand the Stand-Alone Instances node.
2. Select the Instance you wish to start.
3. On the General tab, click Start to start the instance.

The node agent associated with the instance must be started using the `asadmin start-node-agent` command before you can successfully start the instance.

Once an instance is started, it is possible to perform the following tasks from the General tab:

- Click Stop Instance to stop the instance.
- Click JNDI Browsing to view the JNDI entries for that instance.
- Click View Log Files to view the Log Viewer and specify logging options.
- Click Rotate Log File.
- Click Recover Transactions to recover incomplete transactions.

Equivalent asadmin command: start-instance.

Recovering Transactions

Transactions might be incomplete either because the server crashed or a resource manager crashed. It is essential to complete these stranded transactions and recover from the failures. Application Server is designed to recover from these failures and complete the transactions upon server startup.

If the selected server is running, then recovery will be done by the same server. If the selected server is not running, then the selected Destination Server will do the recovery.

Stopping an Instance

To stop an instance, perform these steps:

1. In the tree component, expand the Standalone Instances node.
2. Select the Instance you wish to stop.
3. On the General tab, click Stop to stop the instance.

Equivalent asadmin command: stop-instance.

Shutting Down the Server Instance

To shut down the Administration Server:

1. In the tree component, select the Standalone Instances node.
2. Select the Administation Server Instance
3. Click Stop.

A confirmation dialog displays to confirm that you wish to shutdown the Administration Server.

Configuration Changes

Changing Application Server Configuration

When making any of these configuration changes, 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 or Domain.

With dynanamic configuration, most changes take effect while the server is running. To make the following configuration changes, do *NOT* 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.

Ports in the Application Server

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

Table 1-1 Application Server Listeners that Use Ports

Listener	Default Port Number	Description
Administrative server	4848	A domain's administrative server is accessed by the Admin Console and the asadmin utility. For the Admin Console, specify the port number in the URL of the browser. When executing an asadmin command remotely, specify the port number with the --port option.
HTTP	8081	The Web server listens for HTTP requests on a port. To access deployed Web applications and services, clients connect to this port.
HTTPS	8181	Web applications configured for secure communications listen on a separate port.
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 an instance under the Standalone Instances node.
2. Select the Properties tab.
3. On the Instance Specific page, the default port numbers are identified. It is possible to set the configuration to override these values.

Changing the Administrative Server Port

1. In the tree component, expand the Configurations node.
2. Expand the server-config (Admin Config) node
3. Expand the HTTP Service node.
4. Expand the HTTP Listeners node.
5. Select the admin-listener node.
6. On the Edit HTTP Listener page, change the value of the Listener Port field.
7. Restart the server.

Changing an HTTP Port

1. In the tree component, expand the HTTP Service node.
2. Expand the HTTP Listeners node.
3. Select the HTTP 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 Configurations node.
2. Expand the server-config (Admin Config) node
3. Expand the ORB node.
4. Expand the IIOP Listeners node.
5. Select the listener whose port number you want to change.
6. On the Edit IIOP Listener page, change the value of the Listener Port field.
7. Click Save.
8. Restart the server.

Configuring a JMX Connector Using the Admin Service

Use the Admin Service to configure a JSR-160 compliant remote JMX connector, which handles communication between the domain administration server and the node agents, which manage server instances on a host machine, for remote server instances.

The Admin Service determines whether the server instance is a regular instance, a domain administration server (DAS), or a combination. A DAS is similar to a J2EE server instance, except that user applications and resources are not deployed to a DAS, though it is capable of serving user application requests. The only significant difference between a DAS and a J2EE Server Instance is that the former can not be a part of a cluster, the homogeneous unit of server instances.

To configure the JMX connector:

1. Select Configurations from the tree.
2. **2.** Select the instance to configure:
    a. To configure a particular instance, select the instance's config node. For example, for the default instance, server, select the server-config node.
    b. **b.** To configure the default settings for future instances that use a copy of default-config, select the default-config node.
3. Select Admin Service from the tree.
4. From the Type drop-down menu, select what you want the Admin service to configure: DAS , DAS and server, or server. Selecting DAS and server is the same as selecting DAS. The server selection selects a non-DAS server instance.
5. In the JMX Connector Name field, enter the name of the JMX connector used internally. The name of the connector is system.

Editing the JMX Connector Configuration

With the Edit JMX Connector screen you can edit the configuration of the JSR 160-compliant JMX Connector.

1. Select Configurations from the tree.
2. Select the instance to configure:

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

b. To configure the default settings for future instances that use a copy of default-config, select the default-config node.

3. Expand the Admin Service node and click system, which is the JMX connector used internally.
4. Enter the port of the JMX connector server. The JMX service URL is a function of the protocol, port, and address, as defined by the JSR 160 1.0 Specification
5. Enter the protocol that this JMX connector should support. The Application Server version 8.1 supports the rmi_jrmp protocol only.
6. In the Realm Name field, enter the name that represents the special administrative realm. All authentication will be handled by this realm.
7. Select the Enabled checkbox to indicate that transport layer security should be used in the JMX connector. If you enable transport layer security, fill out the SSL section by following the instructions in Configuring Security for the Admin Service's JMX Connector.

Changing the J2SE Software

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

Using Online Help

The Admin Console's online help is context-sensitive: When clicking the Help link in the upper right corner, the help browser window displays a topic related to the current Admin Console page. If the current page has no help information, the Using Online Help topic is displayed.

The online help includes conceptual topics that are not context-sensitive. To view one of these topics, select it from the table of contents in the help browser window.

To go back to the previous help screen:

1. From within the help browser window, right-click to display a selection menu.
2. Select Back.

If you do not find the information you're looking for, check the Application Server Administration Guide, available online at:

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

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
  ```

- *Application Server 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/
  ```

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

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

- *Application Server Release Notes* - Available online at:

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

- *Getting Started With J2EE Connectors* - This document has instructions 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.

Chapter 2

Configuring Clusters

This chapter describes how to use the Admin Console to configure clusters. It contains the following sections:

About Clusters

What is a Cluster?

A cluster is a collection of zero or more server instances that work together as one logical entity. A cluster provides a runtime environment for one or more J2EE applications.

The use of clusters in the Sun Java System Application Server Enterprise Edition 8.1 environment provides the following:

- High availability, by allowing for failover protection for the server instances in a cluster. If one server instance goes down, other server instances take over the requests that the unavailable server instance was serving.

- Scalability, by allowing for the addition of server instances to a cluster, thus increasing the capacity of the system. The load balancer of the Application Server distributes requests to the available server instances within the cluster. No disruption in service is required as an administrator adds more server instances to a cluster.

A cluster has the following properties:

- All instances in the cluster reference the same configuration.
- All instances in the cluster have the same set of deployed applications (for example, a J2EE application EAR file, a web module WAR file, or an EJB JAR file).
- All instances in the cluster have the same set of resources, resulting in the same JNDI namespace.

Every cluster in the domain has a unique name; furthermore, this name must be unique across all node agent names, server instance names, cluster names, and configuration names. The name must not be `domain`. Administrators perform the same operations on a cluster (for example, deploying applications and creating resources) that they perform on an unclustered server instance.

For an overview of the use of clusters, node agents, and server instances, see the *Deployment Planning Guide*. For details about the load balancer, see Configuring Load Balancing and Failover.

Cluster Types

There are two types of clusters: *stand-alone clusters* and *shared clusters*.

- A stand-alone cluster has its own configuration shared by no other server instances or clusters. By default, the name of this configuration is *cluster_name*`-config`, where *cluster_name* represents the name of the cluster.
- A shared cluster shares its configuration with one or more other clusters or unclustered instances.

Clusters, Instances, Load Balancers, and Sessions

Clusters, server instances, load balancers, and sessions are related as follows in the Application Server:

- It is not mandatory that a server instance be part of a cluster. However, an instance that is not part of a cluster cannot take advantage of high availability through transfer of session state from one instance to other instances.
- The server instances within a cluster can be hosted on different machines or on the same machine. That is, you can group server instances across different machines into a cluster.
- A particular load balancer can forward requests to server instances on multiple clusters. You can use this ability of the load balancer to perform an online upgrade without loss of service. For more information, see "Using Multiple Clusters for Online Upgrades Without Loss of Service" on page 60.
- A single cluster can receive requests from multiple load balancers. If a cluster is served by more than one load balancer, you must configure the cluster in exactly the same way on each load balancer.
- Each session is tied to a particular cluster. What this implies is that although an application can be deployed on multiple clusters, sessions will only be failed over to server instances within the same cluster.
- The Application Server supports failover for HTTP sessions and stateful session bean (SFSB) sessions. Failover of certain J2EE object references that are stored within HTTP sessions is also supported.

  For more information on failover of HTTP sessions, see "Configuring Availability and Session Persistence" on page 149

The cluster thus acts as a safe boundary for session failover for the server instances within the cluster. You can use the load balancer and upgrade components within the Application Server without loss of service. For more information, see "Using Multiple Clusters for Online Upgrades Without Loss of Service" on page 60.

Admin Console Tasks for Clusters

- Migrating EJB Timers
- Creating Server Instances for a Cluster
- Configuring Clustered Server Instances
- Configuring Applications for a Cluster
- Configuring Resources for a Cluster
- Deleting a Cluster
- Using Multiple Clusters for Online Upgrades Without Loss of Service

Creating a Cluster

To create a cluster, perform these steps:

1. In the tree component, select the Clusters node.
2. On the Clusters page, click New. The Create Cluster page appears.
3. In the Name field, type a name for the cluster. The name
   - Must consist only of uppercase and lowercase letters, numbers, underscores, hyphens, and periods (.)
   - Must be unique across all node agent names, server instance names, cluster names, and configuration names
   - Must not be `domain`
4. In the Configuration field, choose a configuration from the drop-down list.

   To create a stand-alone cluster, choose `default-config`, and leave the radio button labeled “Make a copy of the selected Configuration” selected. The copy of the default configuration will have the name *cluster_name*`-config`.

   To reference another configuration, choose the configuration from the drop-down list and select the radio button labeled “Reference the selected Configuration.” This action creates a shared cluster if the configuration is used by another cluster.
5. You can add server instances now, or you can wait until after the cluster is created. Before you create server instances for the cluster, first create one or more node agents or node agent placeholders. See “Creating a Node Agent Placeholder” on page 104 for details.

   To create server instances, perform the following steps:

a. In the Server Instances To Be Created area, click Add.

b. Type a name for the instance in the Instance Name field

c. Choose a node agent from the Node Agent drop-down list.

6. Click OK.

7. Click OK on the Cluster Created Successfully page that appears.

For more details on how to administer clusters, server instances, and node agents, see "Deploying Node Agents" on page 97.

Equivalent `asadmin` command: `create-cluster`

Configuring a Cluster

To configure a cluster, perform these steps:

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

2. Select the node for the cluster. On the General Information page, you can perform these tasks:

   - Click Start Instances to start the clustered server instances.
   - Click Stop Instances to stop the clustered server instances.
   - Click Migrate EJB Timers to migrate the EJB timers from a stopped server instance to another server instance in the cluster.

Equivalent `asadmin` commands: `start-cluster`, `stop-cluster`, `migrate-timers`

Migrating EJB Timers

If a server instance stops running abnormally or unexpectedly, it can be necessary to move the EJB timers installed on that server instance to a running server instance in the cluster. To do so, perform these steps:

1. From the Source drop-down list, choose the stopped server instance from which to migrate the timers.

2. Optionally, from the Destination drop-down list, choose the running server instance to which to migrate the timers. If this field is left blank, a running server instance will be randomly chosen.

3. Click OK.

4. Stop and restart the Destination server instance.

An error message appears if the Source server instance is running or if the Destination server instance is not running.

Equivalent `asadmin` command: `migrate-timers`

Creating Server Instances for a Cluster

Before you can create server instances for a cluster, you must first create a node agent or node agent placeholder. See "Creating a Node Agent Placeholder" on page 104 for details.

To create a server instance for a cluster, perform these steps:

1. In the tree component, expand the Clusters node.
2. Select the node for the cluster.
3. Click the Instances tab to bring up the Clustered Server Instances page.
4. Click New to bring up the Create Clustered Server Instance page.
5. In the Name field, type a name for the server instance.
6. Choose a node agent from the Node Agents drop-down list.
7. Click OK.

Equivalent `asadmin` command: `create-instance`

Configuring Clustered Server Instances

To make changes to a clustered server instance after creating it, perform these steps:

1. In the tree component, expand the Clusters node.
2. Expand the node for the cluster that contains the server instance, then select the server instance node to be edited.
3. On the General Information page, you can perform these tasks:
   - Click Start Instance to start the instance.
   - Click Stop Instance to stop a running instance.
   - Click JNDI Browsing to browse the JNDI tree for a running instance.

- Click View Log Files to open the server log viewer.
- Click Rotate Log File to rotate the log file for the instance. This action schedules the log file for rotation. The actual rotation takes place the next time an entry is written to the log file.
- Click Recover Transactions to recover incomplete transactions.
- Click the Properties tab to modify the port numbers for the instance.
- Click the Monitor tab to change monitoring properties.

You can also perform operations on a server instance as follows:

1. In the tree component, expand the Clusters node.
2. Expand the node for the cluster that contains the server instance.
3. Click the Instances tab to go to the Clustered Server Instances page. On this page you can perform the following tasks:
   - Select the checkbox for an instance and click Delete, Start, or Stop.
   - Click the name of the instance to bring up the General Information page.

To delete a clustered server instance from a cluster, select the checkbox next to the instance name and click Delete.

Configuring Applications for a Cluster

To configure applications for a cluster, perform these steps:

1. In the tree component, expand the Clusters node.
2. Select the node for the cluster.
3. Click the Applications tab to bring up the Applications page. On this page, you can perform these tasks:
   - Select the checkbox next to an application and choose Enable or Disable to enable or disable the application for the cluster.
   - From the Deploy drop-down list, select a type of application to deploy. On the Deployment page that appears, specify the application.
   - From the Filter drop-down list, select a type of application to display in the list.

To edit an application, click the application name.

Configuring Resources for a Cluster

To configure resources for a cluster, perform these steps:

1. In the tree component, expand the Clusters node.
2. Select the node for the cluster.
3. Click the Resources tab to bring up the Resources page. On this page, you can perform these tasks:
   - Select the checkbox next to a resource and click Enable or Disable to enable or disable the resource globally. This action does not remove the resource.
   - From the New drop-down list, select a type of resource to create. Make sure to specify the cluster as a target when you create the resource.
   - From the Filter drop-down list, select a type of resource to display in the list.

To edit a resource, click the resource name.

Deleting a Cluster

To delete a cluster, perform these steps:

1. In the tree component, select the Clusters node.
2. On the Clusters page, select the checkbox next to the name of the cluster to be deleted.
3. Click Delete.

Equivalent `asadmin` command: `delete-cluster`

Using Multiple Clusters for Online Upgrades Without Loss of Service

You can use the load balancer and multiple clusters to upgrade components within the Application Server without any loss of service. A component can, for example, be a JVM, the Application Server, or a web application.

To perform this task, do the following:

1. Stop one of the clusters using the Stop Cluster button on the General Information page for the cluster.
2. Upgrade the component in that cluster.
3. Start the cluster using the Start Cluster button on the General Information page for the cluster.
4. Repeat the process with the other clusters, one by one.

Because sessions within one cluster will never fail over to sessions within another cluster, there is no risk of version mismatch caused by a session's failing over from a server instance that is running one version of the component to another server instance (in a different cluster) that is running a different version of the component. A cluster in this way acts as a safe boundary for session failover for the server instances within it.

NOTE This approach is not possible in the following cases:

- When you change the schema of the high-availability database (HADB). For more information, see the Administering High Availability Database chapter in the *Sun Java System Application Server High Availability Administration Guide.*
- When you perform an application upgrade that involves a change to the application database schema.

CAUTION Upgrade all server instances in a cluster together. Otherwise, there is a risk of version mismatch caused by a session failing over from one instance to another where the instances have different versions of components running.

Chapter 3

Configuring Load Balancing and Failover

This chapter describes how to set up load balancing of HTTP requests in the Sun Java System Application Server. It also explains how to configure failover between server instances controlled by the load balancer. In addition, it discusses RMI-IIOP load balancing and Failover.

It contains the following sections:

About HTTP Load Balancing and Failover

HTTP Load Balancing and Failover

The goal of load balancing is to evenly distribute the workload between multiple Sun Java System Application Server instances, stand-alone or clustered, thereby increasing the overall throughput of the system.

Using a load balancer also enables requests to fail over from one server instance to another. For HTTP session information to persist, configure HTTP session persistence. For more information, see “Configuring Availability and Session Persistence”

Use the asadmin tool, not the Admin Console, to configure HTTP load balancing.

Requirements for HTTP Load Balancing

You must meet the following requirements before using the load balancer plug-in for HTTP requests:

- Sun Java System Application Server Enterprise Edition is installed.
- A web server is installed and configured.

  For more information, see “Configuring Web Servers for HTTP Load Balancing” on page 68.
- A load balancer plug-in is installed.
- The Application Server is configured.
  - Application Server instances or clusters participating in load balancing are created.
  - Applications are deployed to all Application Server instances or clusters participating in load balancing.
  - Server instances and clusters participating in load balancing must have a homogenous environment. Usually that means that the server instances reference the same server configuration and have the same applications deployed to them.

Understanding Assigned Requests and Unassigned Requests

When a request first comes in from an HTTP client to the load balancer, it is a request for a new session. A request for a new session is called an *unassigned* request. The load balancer routes this request to an application server instance in the cluster according to a round-robin algorithm. For more information, see "HTTP Load Balancing Algorithm" on page 65.

Once a session is created on an application server instance, the load balancer routes all subsequent requests for this session only to that particular instance. A request for an existing session is called an *assigned* or a *sticky* request.

HTTP Load Balancing Algorithm

The Sun Java System Application Server load balancer uses a sticky round robin algorithm to load balance incoming HTTP and HTTPS requests. All requests for a given session are sent to the same application server instance. With a sticky load balancer, the session data is cached on a single application server rather than being distributed to all instances in a cluster.

Therefore, the sticky round robin scheme provides significant performance benefits that normally override the benefits of a more evenly distributed load obtained with pure round robin.

About the Sticky Round Robin Load Balancing Algorithm

When a new HTTP request is sent to the load balancer plug-in, it's forwarded to an application server instance based on a simple round robin scheme. Subsequently, this request is "stuck" to this particular application server instance, either by using cookies, or explicit URL rewriting.

From the sticky information, the load balancer plug-in first determines the instance to which the request was previously forwarded. If that instance is found to be healthy, the load balancer plug-in forwards the request to that specific application server instance. Therefore, all requests for a given session are sent to the same application server instance.

The load balancer plug-in uses the following methods to determine session stickiness:

- Cookie-Based Method
- Explicit URL Rewriting Method

Cookie-Based Method

In the cookie-based method, the load balancer plug-in uses a separate cookie to record the route information.

NOTE The HTTP client must support cookies to use the cookie based method.

Explicit URL Rewriting Method

In the explicit URL rewriting method, the sticky information is appended to the URL. This method works even if the HTTP client does not support cookies.

Load Balancing and Failover Sample Applications

The following directories contain sample applications that demonstrate load balancing and failover:

install_dir`/samples/ee-samples/highavailability`
install_dir`/samples/ee-samples/failover`

The `ee-samples` directory also contains information for setting up your environment to run the samples.

Overview of HTTP Load Balancing Set-up

Use the `asadmin` tool to configure load balancing in your environment. Follow these steps:

1. Complete the "Requirements for HTTP Load Balancing" on page 64, including installing and configuring a web server and Application Server instances and or clusters.
2. Create a load balancer configuration using the `asadmin` command `create-http-lb-config`.
3. Add references to clusters and stand-alone server instances for the load balancer to manage using `asadmin create-http-lb-ref`.

   If you created the load balancer configuration with a target, and that target is the only cluster or stand-alone server instance the load balancer references, skip this step.

4. Enable the clusters or stand-alone server instances reference by the load balancer using `asadmin enable-http-lb-server`.
5. Enable applications for load balancing using `asadmin enable-http-lb-application`.

   These applications must already be deployed and enabled for use on the clusters or stand-alone instances that the load balancer references. Enabling for load balancing is a separate step from enabling them for use.
6. Create a health checker using `asadmin create-health-checker`.

   The health checker monitors unhealthy server instances so that when they become healthy again, the load balancer can send new requests to them.
7. Generate the load balancer configuration file using `asadmin export-http-lb-config`.

   This command generates a configuration file to use with the load balancer plug-in shipped with the Sun Java System Application Server.
8. Copy the load balancer configuration file to your web server `config` directory where the load balancer plug-in configuration files are stored.

Configuring Web Servers for HTTP Load Balancing

About Web Server Configuration

The load balancer plug-in installation program makes a few modifications to the web server's configuration files. The changes made depend upon the web server.

NOTE The load balancer plug-in can be installed either along with Sun Java System Application Server Enterprise Edition, or separately, on a machine running the supported web server.

For complete details on the installation procedure, see *Sun Java System Application Server Installation Guide.*

Modifications to Sun Java System Web Server

The installation program makes the following changes to the Sun Java System Web Server's configuration files:

1. Adds the following load balancer plug-in specific entries to the web server instance's `magnus.conf` file:

```
##EE lb-plugin
Init fn="load-modules"
shlib="web_server_install_dir/plugins/lbplugin/bin/libpassthrough.so"
funcs="init-passthrough,service-passthrough,name-trans-passthrough"
Thread="no"

Init fn="init-passthrough"

##end addition for EE lb-plugin
```

2. Adds the following entries specific to the load balancer plug-in to the web server instance's `obj.conf` file:

```
<Object name=default>

NameTrans fn="name-trans-passthrough" name="lbplugin"
config-file="web_server_install_dir/web_server_instance/config/loadbalancer.xml"

<Object name="lbplugin">
ObjectType fn="force-type" type="magnus-internal/lbplugin"
PathCheck fn="deny-existence" path="*/WEB-INF/*"
Service type="magnus-internal/lbplugin" fn="service-passthrough"
Error reason="Bad Gateway" fn="send-error" uri="$docroot/badgateway.html"
</object>
```

`lbplugin` is a name that uniquely identifies the `Object`, and *web_server_install_dir/web_server_instance*`/config/loadbalancer.xml` is the location of the XML configuration file for the virtual server on which the load balancer is configured to run.

After installing, configure the load balancer as described in "Overview of HTTP Load Balancing Set-up" on page 67.

Modifications to Apache Web Server

Before installing the load balancer plug-in on Apache, see information on compiling and configuring Apache in Appendix A, "Compiling and Configuring Apache Web Server."

Modifications Made by the Installer

The load balancer plug-in installation program extracts the necessary files to the `libexec` (Apache 1.3) or `modules` (Apache 2.0) folder under the web server's root directory. It adds the following entries specific to the load balancer plug-in to the web server instance's `httpd.conf` file:

```
<VirtualHost machine_name:443>

##Addition for EE lb-plugin

LoadFile /usr/lib/libCstd.so.1

LoadModule apachelbplugin_module libexec/mod_loadbalancer.so
#AddModule mod_apachelbplugin.cpp
<IfModule mod_apachelbplugin.cpp>
       config-file webserver_instance/conf/loadbalancer.xml
locale en
</IfModule>

<VirtualHost machine_ip_address>
DocumentRoot "webserver_instance/htdocs"
ServerName server_name
</VirtualHost>

##END EE LB Plugin ParametersVersion 7
```

NOTE

- On Apache 1.3, when more than one Apache child processes runs, each process has its own load balancing round robin sequence.

  For example, if there are two Apache child processes running, and the load balancing plug-in load balances on to two application server instances, the first request is sent to instance #1 and the second request is also sent to instance #1. The third request is sent to instance #2 and the fourth request is sent to instance #2 again. This pattern is repeated (instance1, instance1, instance2, instance2, etc.)

  This behavior is different from what you might expect, that is, instance1, instance2, instance1, instance2, etc. In Sun Java System Application Server, the load balancing plug-in for Apache instantiates a load balancer instance for each Apache process, creating an independent load balancing sequence.

- Apache 2.0 has multithreaded behavior if compiled with the `--with-mpm=worker` option.

Modifications After Installation

Apache web server must be run in secure mode to ensure that it works well with the load balancer plug-in. Create a directory called `sec_db_files` under *apache_install_dir* and copy *application_server_domain_dir*`/config/`*security_db_files* to *apache_install_dir* / *sec_db_files*.

Additional Modifications on Microsoft Windows

If you are running Apache on Microsoft Windows, after installing the plug-in, some environment variable changes are required:

Add a new path to the Path environment variable by clicking Start->Settings->Control Panel->System->Advanced->Environment Variables->System Variables. Edit the Path variable to include the following:

application_server_install_dir`/bin`

In addition, set the environment variable NSPR_NATIVE_THREADS_ONLY to 1 before starting Apache web server.

On the Environment Variables window, under System Variables, click New.Enter the following name and value pair:

Variable name: NSPR_NATIVE_THREADS_ONLY

Variable value: 1

Restart the machine.

Modifications to Microsoft IIS

To configure Microsoft Internet Information Services (IIS) to use the load balancer plug-in, modify certain properties in Windows Internet Services Manager. The Internet Services Manager is located in the Administrative Tools folder in the Control Panel folder.

Make these modifications after installing the Sun Java System Application Server.

1. Open the Internet Services Manager.
2. Select the web site for which you want to enable the plug-in. This web site is typically named the Default Web Site.
3. Right click on the web site and select Properties to open the Properties notebook.
4. To add a new ISAPI filter, open the ISAPI Filters tab, click Add, and follow the steps given below:
   a. In the Filter Name field, enter Application Server
   b. In the Executable field, type `C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.dll`
   c. Click OK, and close the Properties notebook.
5. Create and configure a new virtual directory:
   a. Right click on the default web site, select New, and then Virtual Directory.

      The Virtual Directory Creation Wizard opens.
   b. In the Alias field, type `sun-passthrough`.
   c. In the Directory field, type `C:\Inetpub\wwwroot\sun-passthrough`
   d. Check the Execute Permission checkbox. Leave all other permission-related check boxes are left unchecked.
   e. Click Finish.
6. Add the path of `sun-passthrough.dll` file and *application_server_install_dir*`/bin` to the system's `PATH` environment variable. Restart the machine.

7. Stop and start the web server for the new settings to take effect.

   To stop the web server, right click on the web site and select `Stop`. To start the web server, right click on the web site and select `Start`.

   Next, type the following in a web browser to access the web application context root:

   `http://`*webserver_name*`/`*web_application*

   where *webserver_name* is the hostname or IP address of the web server and */web_application* is the context root that you listed in the `C:\Inetpub\wwwroot\sun-passthrough\sun-passthrough.properties` file. Verify that the web server, load balancer plug-in, and Application Server are operating correctly.

The installer automatically configures the following properties in `sun-passthrough.properties`. You can change the default values.

Table 3-1 Automatically configured sun-passthrough.properties for Microsoft IIS

Property	Definition	Default Value
lb-config-file	Path to the load balancer configuration file	*IIS_www_root*`\sun-passthrough\loadbalancer.xml`
log-file	Path to the load balancer log file	*IIS_www_root*`\sun-passthrough\lb.log`
log-level	Log level for the web server	INFO

Configuring Multiple Web Server Instances

The Sun Java System Application Server installer does not allow the installation of multiple load balancer plug-ins on a single machine. To have multiple web servers with the load balancer plug-in on a single machine, in either a single cluster or multiple clusters, a few manual steps are required to configure the load balancer plug-in.

1. Configure the new web server instance to use the load balancer plug-in, as described in "Modifications to Sun Java System Web Server" on page 69, "Modifications to Apache Web Server" on page 69, or "Modifications to Microsoft IIS" on page 71.

2. Copy the `sun-loadbalancer_1_1.dtd` file from the existing web server instance's `config` directory to the new instance's `config` directory.

3. To use the same load balancer configuration, copy the `loadbalancer.xml` file from the existing web server instance's `config` directory to the new instance's `config` directory.
4. To use a different load balancer configuration:
   a. Create a new load balancer configuration using `asadmin create-http-lb-config`.
   b. Export the new configuration to a `loadbalancer.xml` file using `asadmin export http-lb-config`.
   c. Copy that `loadbalancer.xml` file to the new web server's `config` directory.

   For information on creating a load balancer configuration and exporting it to a `loadbalancer.xml` file, see "HTTP Load Balancer Configuration Tasks."

HTTP Load Balancer Configuration Tasks

Creating an HTTP Load Balancer Configuration

A load balancer configuration is a named configuration in the `domain.xml` file that defines a load balancer.

Load balancing configuration is extremely flexible:

- Each load balancer configuration can have multiple load balancers associated with it, though each load balancer has only one load balancer configuration.
- A load balancer services only one domain, though a domain can have multiple load balancers associated with it.

Create the configuration using the `asadmin` command `create-http-lb-config`. Specify the following parameters:

- response timeout

  The time in seconds within which a server instance must return a response. If no response is received within the time period, the server is considered unhealthy. The default is 60.

- HTTPS routing

  Specifies whether HTTPS requests to the load balancer result in HTTPS or HTTP requests to the server instance.

  For more information, see "Configuring HTTP and HTTPS Session Failover" on page 82.

- reload interval

  The interval between checks for changes to the load balancer configuration file `loadbalancer.xml`. When the check detects changes, the configuration file is reloaded. A value of 0 disables reloading.

  For more information, see "Enabling Dynamic Reconfiguration" on page 79.

- monitor

  Specifies whether monitoring is enabled for the load balancer.

  For more information, see "Monitoring the HTTP Load Balancer Plug-in" on page 84.

- routecookie

  Specifies the name of the cookie the load balancer plug-in uses to record the route information. The HTTP client must support cookies. If your browser is set to ask before storing a cookie, the name of the cookie is JROUTE.

- target

  Specifies the target for the load balancer configuration. If you specify a target, it is the same as adding a reference to it. Targets can be clusters or stand-alone instances.

For more information see the documentation for `create-http-lb-config`, `delete-http-lb-config`, and `list-http-lb-configs`.

Creating an HTTP Load Balancer Reference

When you create a reference in the load balancer to a stand-alone server or cluster, the server or cluster is added to the list of target servers and clusters the load balancer controls. The referenced server or cluster still needs to be enabled (using `enable-http-lb-server`) before requests to it are load balanced. If you created the load balancer configuration with a target, that target is already added as a reference.

Create a reference using `create-http-lb-ref`. You must supply the load balancer configuration name and the target server instance or cluster.

To delete a reference, use `delete-http-lb-ref`. Before you can delete a reference, the referenced server or cluster must be disabled using `disable-http-lb-server`.

For more information, see the documentation for `create-http-lb-ref` and `delete-http-lb-ref`.

Enabling Server Instances for Load Balancing

After creating a reference to the server instance or cluster, enable the server instance or cluster using `enable-http-lb-server`. If you used a server instance or cluster as the target when you created the load balancer configuration, you must enable it.

For more information, see the documentation for `enable-http-lb-server`.

Enabling Applications for Load Balancing

All servers managed by a load balancer must have homogenous configurations, including the same set of applications deployed to them. Once an application is deployed and enabled for access (this happens during or after the deployment step) you must enable it for load balancing. If an application is not enabled for load balancing, requests to it are not load balanced and failed over, even if requests to the servers the application is deployed to are load balanced and failed over.

When enabling the application, specify the application name and target. If the load balancer manages multiple targets (for example, two clusters), enable the application on all targets.

For more information, see the online help for `enable-http-lb-application`.

If you deploy a new application, you must also enable it for load balancing and export the load balancer configuration again.

Creating the HTTP Health Checker

The load balancer's health checker periodically checks all the configured Application Server instances that are marked as unhealthy. A health checker is not required, but if no health checker exists, or if the health checker is disabled, the periodic health check of unhealthy instances is not performed.

The load balancer's health check mechanism communicates with the application server instance using HTTP. The health checker sends an HTTP request to the URL specified and waits for a response. A status code in the HTTP response header between 100 and 500 means the instance is healthy.

Creating a Health Checker

To create the health checker, use the `asadmin create-http-health-checker` command. Specify the following parameters:

- url

  Specifies the listener's URL that the load balancer checks to determine its state of health. The default is "`/`".

- `interval`

  Specifies the interval in seconds at which health checks of instances occur. The default is 30 seconds. Specifying 0 disables the health checker.

- timeout

  Specifies the timeout interval in seconds within which a response must be obtained for a listener to be considered healthy. The default is `10` seconds.

If an application server instance is marked as unhealthy, the health checker polls the unhealthy instances to determine if the instance has become healthy. The health checker uses the specified URL to check all unhealthy application server instances to determine if they have returned to the healthy state.

If the health checker finds that an unhealthy instance has become healthy, that instance is added to the list of healthy instances.

For more information see the documentation for `create-http-health-checker` and `delete-http-health-checker`.

Additional Health Check Properties for Healthy Instances

The health checker created by `create-http-health-checker` only checks unhealthy instances. To periodically check healthy instances set some additional properties in your exported `loadbalancer.xml` file.

NOTE These properties can only be set by manually editing `loadbalancer.xml` *after* you've exported it. There is no equivalent asadmin command to use.

To check healthy instances, set the following properties:

Table 3-2 Health-checker properties

Property	Definition
active-healthcheck-enabled	True/false flag indicating whether to ping healthy server instances to determine whether they are healthy. To ping server instances, set the flag to true.
number-healthcheck-retries	Specifies how many times the load balancer's health checker pings an unresponsive server instance before marking it unhealthy. Valid range is between 1 and 1000. A default value to set is 3.

Set the properties by editing the `loadbalancer.xml` file. For example:

```
<property name="active-healthcheck-enabled" value="true"/>
<property name="number-healthcheck-retries" value="3"/>
```

If you add these properties, then subsequently edit and export the `loadbalancer.xml` file again, you must add these properties to the file again, since the newly exported configuration won't contain them.

Exporting the Load Balancer Configuration File

The load balancing plug-in shipped with Sun Java System Application Server uses a configuration file called `loadbalancer.xml`. Use the `asadmin` tool to create a load balancer configuration in the `domain.xml` file. After configuring the load balancing environment, export it to a file:

1. Export a `loadbalancer.xml` file using the `asadmin` command `export-http-lb-config`.

   Export the `loadbalancer.xml` file for a particular load balancer configuration. You can specify a path and a different file name. If you don't specify a file name, the file is named `loadbalancer.xml.`*load_balancer_config_name*. If you don't specify a path, the file is created in the *application_server_install_dir*`/domains/`*domain_name*`/generated` directory.

   To specify a path on Windows, use quotes around the path. For example, "`c:\sun\AppServer\loadbalancer.xml`".

2. Copy the exported load balancer configuration file to the web server's configuration directory.

   For example, for the Sun Java System Web Server, that location might be *web_server_root*`/config`.

   The load balancer configuration file in the web server's configuration directory must be named `loadbalancer.xml`. If your file has a different name, such as `loadbalancer.xml.`*load_balancer_config_name*, you must rename it.

Changing the HTTP Load Balancer Configuration

If you change an HTTP load balancer configuration by creating or deleting references to servers, deploying new applications, enabling or disabling servers or applications, and so on, export the load balancer configuration file again and copy it to the web server's `config` directory. For more information, see "Exporting the Load Balancer Configuration File" on page 78.

The load balancer plug-in checks for an updated configuration periodically based on the reload interval specified in the load balancer configuration. After the specified amount of time, if the load balancer discovers a new configuration file, it starts using that configuration.

Enabling Dynamic Reconfiguration

When dynamic reconfiguration is enabled, the load balancer plug-in periodically checks for an updated configuration. To enable dynamic reconfiguration:

- To enable dynamic reconfiguration when creating a load balancer configuration, use the `--reloadinterval` option when running `asadmin create-http-lb-config`.

  This option sets the amount of time between checks for changes to the load balancer configuration file `loadbalancer.xml`. A value of 0 disables reloading. By default, dynamic reloading is enabled, and the interval is set to 60 seconds.

- To enable dynamic reloading if you have disabled it, or to change the reload interval, use the `asadmin set` command.

After changing these settings, export the load balancer configuration file again and copy it to the web server's `config` directory.

If you enable dynamic reconfiguration after it has previously been disabled, you also must restart the web server.

NOTE

- If the load balancer encounters a hard disk read error while attempting to reconfigure itself, then it uses the configuration that is currently in memory. The load balancer also ensures that the modified configuration data is compliant with the DTD before over writing the existing configuration.

  If a disk read error is encountered, a warning message is logged to the web server's error log file.

  The error log for Sun Java System Web Server' is at: *web_server_install_dir*`/`*webserver_instance*`/logs/`.

Disabling (Quiescing) a Server Instance or Cluster

Before stopping an application server for any reason, you want the instance to complete serving requests. The process of gracefully disabling a server instance or cluster is called quiescing.

The load balancer uses the following policy for quiescing application server instances:

- If an instance (either stand-alone or part of a cluster) is disabled, and the timeout has not expired, sticky requests continue to be delivered to that instance. New requests, however, are not sent to the disabled instance.
- When the timeout expires, the instance is disabled. All open connections from the load balancer to the instance are closed. The load balancer does not send any requests to this instance, even if all sessions sticking to this instance were not invalidated. Instead, the load balancer fails over sticky requests to another healthy instance.

To disable a server instance or cluster:

1. Run `asadmin disable-http-lb-server`, setting the timeout (in minutes).
2. Export the load balancer configuration file using `asadmin export-http-lb-config`.
3. Copy the exported configuration to the web server `config` directory.
4. Stop the server instance or instances.

Disabling (Quiescing) an Application

Before you undeploy a web application, you want to the application to complete serving requests. The process of gracefully disabling an application is called quiescing.

The load balancer uses the following policy for quiescing applications:

- If an application is disabled, and the timeout has not expired, new requests to the disabled applications are not forwarded by the load balancer. These requests are returned to the web server. Sticky requests continue to be forwarded until the timeout expires.
- When the timeout expires, the application is disabled. The load balancer does not accept any requests for this application, including sticky requests.

When you disable an application from every server instance or cluster the load balancer references, the users of the disabled application experience loss of service until the application is enabled again.

If you disable the application from one server instance or cluster while keeping it enabled in another server instance or cluster, users can still access the application.

To disable an application:

1. Run `asadmin disable-http-lb-application`, specifying the timeout (in minutes) the name of the application to disable, and the target cluster or instance on which to disable it.
2. Export the load balancer configuration file using `asadmin export-http-lb-config`.
3. Copy the exported configuration to the web server `config` directory.

Configuring HTTP and HTTPS Session Failover

The load balancer plug-in fails over HTTP / HTTPS sessions to another application server instance if the original application server instance to which the session was connected becomes unavailable. This section describes how to configure the load balancer plug-in to enable HTTP / HTTPS routing and session failover.

For information about configuring HTTP session persistence, see "Configuring Availability and Session Persistence"

The following topics are discussed in this section:

About HTTPS Routing

All incoming requests, whether HTTP or HTTPS, are routed by the load balancer plug-in to application server instances. However, if HTTPS routing is enabled, a HTTPS request will be forwarded by the load balancer plug-in to the application server using an HTTPS port only. Note that HTTPS routing is performed on both new and sticky requests.

If an HTTPS request is received and no session is in progress, then the load balancer plug-in selects an available application server instance with a configured HTTPS port, and forwards the request to that instance.

In an ongoing HTTP session, if a new HTTPS request for the same session is received, then the session and sticky information saved during the HTTP session is used to route the HTTPS request. The new HTTPS request is routed to the same server where the last HTTP request was served, but on the HTTPS port.

Configuring HTTPS Routing

The `httpsrouting` option of the `create-http-lb-config` command controls whether HTTPS routing is turned on or off for all the application servers that are participating in load balancing. If this option is set to false, all HTTP and HTTPS requests are forwarded as HTTP. Set it to true when creating a new load balancer configuration, or change it later using the `asadmin set` command.

NOTE

- For HTTPS routing to work, one or more HTTPS listeners must be configured.
- If `https-routing` is set to `true`, and a new or a sticky request comes in where there are no healthy HTTPS listeners in the cluster, then that request generates an error.

Known Issues in Load Balancing HTTP/HTTPS Requests

The following list discusses the limitations in Load Balancer with respect to HTTP / HTTPS request processing.

- If a session uses a combination of HTTP and HTTPS requests, then the first request must be an HTTP Request. If the first request is an HTTPS request, it cannot be followed by an HTTP request. This is because the cookie associated with the HTTPS session is not returned by the browser. The browser interprets the two different protocols as two different servers, and initiates a new session.

  This limitation is valid only if `httpsrouting` is set to `true`.

- If a session has a combination of HTTP and HTTPS requests, then the application server instance must be configured with both HTTP and HTTPS listeners.

  This limitation is valid only if `httpsrouting` is set to `true`.

- If a session has a combination of HTTP and HTTPS requests, then the application server instance must be configured with HTTP and HTTPS listeners that use standard port numbers, that is, 80 for HTTP, and 443 for HTTPS.

  This limitation is valid regardless of the value set for `httpsrouting`.

Configuring Idempotent URLs

To enhance the availability of deployed applications, configure the environment to retry failed idempotent HTTP requests on all the application server instances serviced by a load balancer. This option is used for read-only requests, for example, to retry a search request.

An idempotent request is one that does not cause any change or inconsistency in an application when retried. In HTTP, some methods (such as GET) are idempotent, while other methods (such as POST) are not. Retrying an idempotent URL must not cause values to change on the server or in the database. The only difference is a change in the response received by the user.

Examples of idempotent requests include search engine queries and database queries. The underlying principle is that the retry does not cause an update or modification of data.

Configure idempotent URLs in the `sun-web.xml` file. When you export the load balancer configuration, idempotent URL information is automatically added to the `loadbalancer.xml` file.

For more information on configuring idempotent URLs, see the *Developer's Guide*.

Configuring HTML Error Pages

You can specify your own error page or a URL to an error page to be displayed to the end user. Specifying an error page overrides all other mechanisms configured for error reporting.

Configure HTML error pages in the `sun-web.xml` file. When you export the load balancer configuration, HTML error page information is automatically added to the `loadbalancer.xml` file from the `sun-web.xml` file.

For more information on configuring HTML error pages, see the *Developer's Guide*.

Monitoring the HTTP Load Balancer Plug-in

Configuring Log Messages

The load balancer plug-in uses the web server's logging mechanism to write log messages. The default log level on the Application Server is set to the default logging level on Sun Java System Web Server (`INFO`), Apache Web Server (`WARN`) and Microsoft IIS (`INFO`). The application server log levels - `FINE`, `FINER`, and `FINEST` map to the `DEBUG` level on the web server.

These log messages are written to the web server log files, and are in the form of raw data that can be parsed using scripts, or imported into spreadsheets to calculate required metrics.

Types of Log Messages

The load balancer plug-in generates the following three different sets of log messages:

- Load Balancer Configurator Log Messages
- Request Dispatch and Runtime Log Messages
- Configurator Error Messages

Load Balancer Configurator Log Messages

These messages will be logged when you are using idempotent URLs and error page settings.

An output for idempotent URL pattern configuration contains the following information:

- When the log level is set to FINE:

  ```
  CONFxxxx: IdempotentUrlPattern configured <url-pattern>
  <no-of-retries> for web-module : <web-module>
  ```

- When the log level is set to SEVERE:

  ```
  CONFxxxx: Duplicate entry of Idempotent URL element <url-pattern>
  for webModule <web-module> in loadbalancer.xml."
  ```

- When the log level is set to WARN:

  ```
  CONFxxxx: Invalid IdempotentUrlPatternData <url-pattern> for
  web-module <web-module>
  ```

An output for error page URL configuration contains the following information (log level set to WARN):

```
CONFxxxx: Invalid error-url for web-module <web-module>
```

Request Dispatch and Runtime Log Messages

These log messages are generated while a request is being load balanced and dispatched.

- An output for standard logging for each method start contains the following information (log level set to FINE):

  ```
  ROUTxxxx: Executing Router method <method_name>
  ```

- An output for router logs for each method start contains the following information (log level set to INFO):

  ```
  ROUTxxxx: Successfully Selected another ServerInstance for
  idempotent request <Request-URL>
  ```

- An output for runtime logs contains the following information (log level set to INFO):

  ```
  RNTMxxxx: Retrying Idempotent <GET/POST/HEAD> Request <Request-URL>
  ```

Configurator Error Messages

These errors appear if there are configuration problems, for example, if the custom error page referenced is missing.

- Log level set to INFO:

  ROUTxxxx: Non Idempotent Request <Request-URL> cannot be retried

  ```
  Example :: ROUTxxxx: Non Idempotent Request
  http://sun.com/addToDB?x=11&abc=2 cannot be retried
  ```

- Log level set to FINE:

  RNTMxxxx: Invalid / Missing Custom error-url / page: <error-url> for web-module: <web-module>

  ```
  Example :: RNTMxxxx: Invalid / Missing Custom error-url /  page:
  myerror1xyz for web-module: test
  ```

Configuring Monitoring

Perform the following steps to turn on load balancer plug-in log messages:

1. Set the logging options in the web server.
   a. From Sun Java System Web Server's admin console, go to the `Magnus Editor` tab.

      Set the `Log Verbose` option to `On`.

   b. For Apache Web Server, set the log level to `DEBUG`.

   c. For Microsoft IIS, set the log level to `FINE` in the `sun-passthrough.properties` file.
2. Set the load balancer configuration's `monitor` option to true.

   Use the `asadmin create-http-lb-config` command to set monitoring to true when you initially create the load balancer configuration, or use the `asadmin set` command to set it to true later. Monitoring is disabled by default.

The load balancer plug-in logs the following information:

- Request start / stop information for every request.
- Failed-over request information when the request fails over from an unhealthy instance to a healthy instance.
- List of unhealthy instances at the end of every health check cycle.

NOTE When logging is enabled on the load balancer plug-in, and if the web server logging level is set to `DEBUG` or to print verbose messages, the load balancer writes HTTP session IDs in the web server log files. Therefore, if the web server hosting the load balancer plug-in is in the DMZ, do not use the `DEBUG` or similar log level in production environments.

If you must use the `DEBUG` logging level, turn off load balancer logging by setting `require-monitor-data` property to `false` in `loadbalancer.xml`.

Understanding Monitoring Messages

The format of the load balancer plug-in log messages is as follows.

- The start of an HTTP request contains the following information:

  ```
  RequestStart Sticky(New) <req-id> <time-stamp> <URL>
  ```

  The timestamp value is the number of milliseconds from January 1, 1970. For example:

  ```
  RequestStart New 123456 602983
  http://austen.sun.com/Webapps-simple/servlet/Example1
  ```

- The end of an HTTP request contains the `RequestExit` message, as follows:

  ```
  RequestExit Sticky(New) <req-id> <time-stamp> <URL> <listener-id>
  <response-time> Failure-<reason for error>(incase of a failure)
  ```

  For example:

  ```
  RequestExit New 123456 603001
  http://austen.sun.com/Webapps-simple/servlet/Example1
  http://austen:2222 18
  ```

  NOTE In the RequestExit message, <response-time> indicates the total request turn-around time in milliseconds, from the perspective of the load balancer plug-in.

- The list of unhealthy instances, as follows:

  ```
  UnhealthyInstances <cluster-id> <time-stamp> <listener-id>,
  <listener-id>...
  ```

  For example:

  ```
  UnhealthyInstances cluster1 701923 http://austen:2210, http://austen:3010
  ```

- A list of failed-over requests, as follows:

  ```
  FailedoverRequest <req-id> <time-stamp> <URL> <session-id>
  <failed-over-listener-id> <unhealthy-listener-id>
  ```

  For example:

  ```
  FailedoverRequest 239496 705623
  http://austen.sun.com/Apps/servlet/SessionTest 16dfdac3c7e80a40
  http://austen:4044 http://austen:4045
  ```

Upgrading an Application

About Rolling Upgrades

To upgrade an application without loss of service to a user, upgrade the application on one server or cluster at a time. The cluster transparently maintains a mixed-version environment, and users are unaware that the upgrade is taking place. This type of upgrade is called a rolling upgrade.

Rolling upgrades are only possible if old and new versions of the application are compatible and can both run at once. Session information must be compatible. Perform a mixed-mode rolling upgrade in a single stand-alone cluster, or in multiple clusters.

Rolling upgrade in a mixed-mode environment is not possible if the application has major changes, for example, changes to the database schema. In that case, bring down the application while you upgrade. Undeploy the application, then redeploy the upgraded application with the same name.

Upgrading In a Single Stand-alone Cluster

To upgrade an application in a single, stand-alone cluster (that is, a cluster which does not share a configuration with any other cluster):

1. Save an old version of the application or back up the domain.

   To back up the domain use the `asadmin backup-domain` command.

2. Turn off dynamic reconfiguration for the cluster if it is enabled.

   Through the Admin Console:

   a. Expand the Configurations node.

   b. Click the name of the cluster's configuration.

   c. On the Configuration System Properties page, uncheck the Dynamic Reconfiguration Enabled box.

d. Click Save

The `asadmin` equivalent command is `asadmin set`. The syntax is:

`asadmin set --user` *user* `--passwordfile` *password_file* *cluster_name*`-config.dynamic-reconfiguration-enabled=false`

3. Redeploy the upgraded application to the target `domain`. If you redeploy using the Admin Console, the domain is automatically the target. Because dynamic reconfiguration is disabled, the old application continues to run on the cluster.
4. Enable the redeployed application for the instances using `asadmin enable-http-lb-application`.
5. Disable one server instance using `asadmin disable-http-lb-server`.
6. Export the load balancer configuration file using `asadmin export-http-lb-config`.
7. Copy the exported configuration file to the web server instance's configuration directory. For example, for Sun Java System Web Server, the location is *web_server_install_dir*`/https-`*host-name*`/config/loadbalancer.xml`
8. Wait until the timeout has expired. Monitor the load balancer's log file to make sure the instance is offline.
9. Restart the disabled server instance while the other instances in the cluster are still running. The restart causes the server to synchronize with the domain and update the application.
10. Test the application on the restarted server to make sure it runs correctly.
11. Enable the server instance using `asadmin enable-http-lb-server`.
12. Export the load balancer configuration file using `asadmin export-http-lb-config`.
13. Copy the configuration file to the web server's configuration directory.
14. Repeat Step 5 through Step 13 for each instance in the cluster.
15. When all server instances have the new application and are running, enable dynamic reconfiguration for the cluster again.

Upgrading in Two Clusters

1. Save an old version of the application or back up the domain.

   To back up the domain use the `asadmin backup-domain` command.

2. Turn off dynamic reconfiguration for both clusters if it is enabled.

   Through the Admin Console:

   a. Expand the Configurations node.

   b. Click the name of one cluster's configuration.

   c. On the Configuration System Properties page, uncheck the Dynamic Reconfiguration Enabled box.

   d. Click Save

   e. Repeat for the second cluster.

   The `asadmin` equivalent command is `asadmin set`. The syntax is:

   `asadmin set --user` *user* `--passwordfile` *password_file* *cluster_name*`-config.dynamic-reconfiguration-enabled=false`

3. Redeploy the upgraded application to the target `domain`. If you redeploy using the Admin Console, the domain is automatically the target. Because dynamic reconfiguration is disabled, the old application continues to run on the clusters.

4. Enable the redeployed application for the clusters using `asadmin enable-http-lb-application`.

5. Disable one cluster from the load balancer using `asadmin disable-http-lb-server`.

6. Export the load balancer configuration file using `asadmin export-http-lb-config`.

7. Copy the exported configuration file to the web server instance's configuration directory. For example, for Sun Java System Web Server, the location is *web_server_install_dir*`/https-`*host-name*`/config/loadbalancer.xml`

8. Wait until the timeout has expired. Monitor the load balancer's log file to make sure the cluster is offline.

9. Restart the disabled cluster while the other cluster is still running. The restart causes the cluster to synchronize with the domain and update the application.

10. Test the application on the restarted cluster to make sure it runs correctly.

11. Enable the cluster using `asadmin enable-http-lb-server`.

12. Export the load balancer configuration file using `asadmin export-http-lb-config`.

13. Copy the configuration file to the web server's configuration directory.

14. Repeat Step 5 through Step 13 for the other cluster.
15. When all server instances have the new application and are running, enable dynamic reconfiguration for both clusters again.

About RMI-IIOP Load Balancing and Failover

Requirements for RMI-IIOP Load Balancing and Failover

Sun Java™ System Application Server provides high availability of remote EJB references and name service objects over RMI-IIOP. Before using these features, your environment must meet the following requirements:

- Sun Java System Application Server Enterprise Edition is installed.
- A cluster of at least two application server instances exist s.

  For more information on clusters, see "Configuring Clusters"
- J2EE applications are deployed to all application server instances and clusters that will participate in load balancing.
- RMI-IIOP client applications are enabled for load balancing.

  Load balancing is supported for the following RMI-IIOP clients:

  - Java applications executing in the Application Client Container (ACC) accessing EJBs deployed on an Application Server instance.
  - Java applications, not running in the ACC, accessing EJBs deployed on an Application Server instance.

  The configuration settings required to enable RMI-IIOP-based applications depend on the type of client. For more information on configuring RMI-IIOP client applications for load balancing, see the *Sun Java System Application Server Developer's Guide*.

  For additional information on RMI-IIOP failover and load balancing, see the *Sun Java System Application Server High Availability Administration Guide*.

NOTE RMI-IIOP failover over SSL is not supported.

RMI-IIOP Load Balancing and Failover Algorithm

Sun Java System Application Server employs a randomization and round-robin algorithm for load balancing of remote EJB references and name service objects on the RMI-IIOP path.

When an RMI-IIOP client first creates a new `InitialContext` object, the list of available Application Server IIOP endpoints is randomized for that client. For that `InitialContext` object, the load balancer directs lookup requests and other `InitialContext` operations to the first endpoint on the list. If the first endpoint is not available then the second endpoint in the list is used, and so on.

Each time the client subsequently creates a new `InitialContext` object, the endpoint list is rotated so that a different IIOP endpoint is used for `InitialContext` operations.

When you obtain or create beans from references obtained by an `InitialContext` object, those beans are created on the Application Server instance serving the IIOP endpoint assigned to the `InitialContext` object. The references to those beans contain the IIOP endpoint addresses of all Application Server instances in the cluster.

The *primary endpoint* is the bean endpoint corresponding to the `InitialContext` endpoint used to look up or create the bean. The other IIOP endpoints in the cluster are designated as *alternate endpoints*. If the bean's primary endpoint becomes unavailable, further requests on that bean fail over to one of the alternate endpoints.

RMI-IIOP Sample Application

The following directory contains a sample application that demonstrates using RMI-IIOP failover with and without ACC:

install_dir`/samples/ee-samples/sfsbfailover`

See the `index.html` file accompanying this sample for instructions on running the application with and without ACC. The `ee-samples` directory also contains information for setting up your environment to run the samples.

Chapter 4

Configuring Node Agents

This chapter describes the node agents in Application Server. It contains the following sections:

About Node Agents

Node Agents

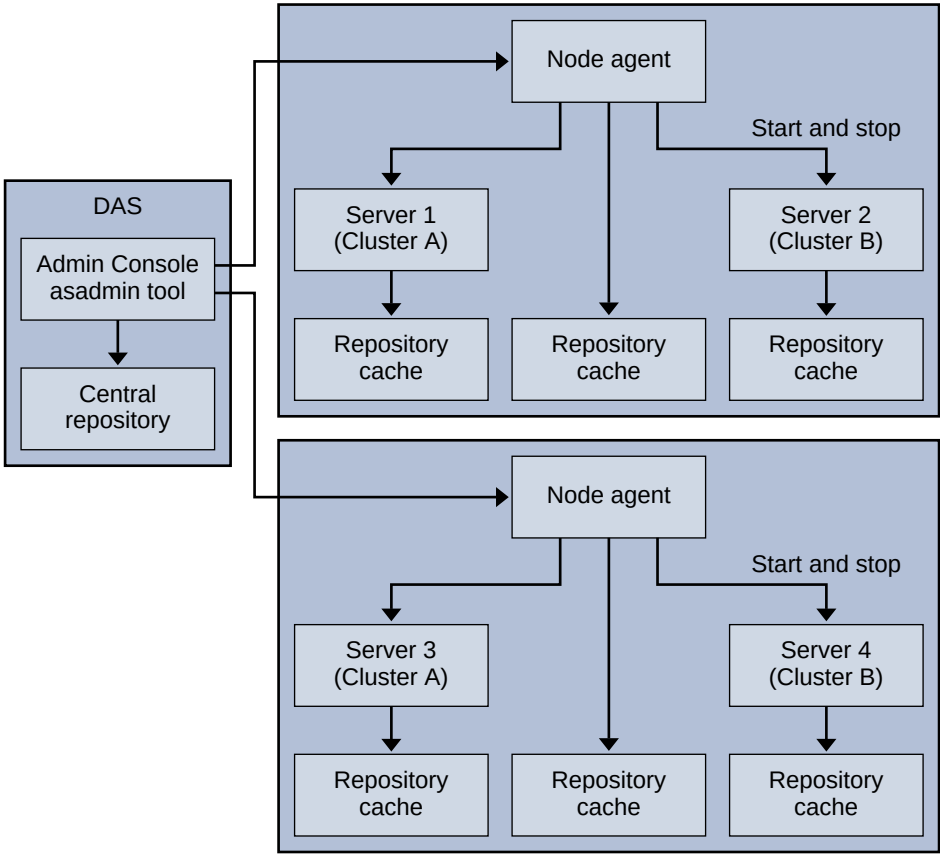
A node agent is a lightweight agent that is required on every machine that hosts server instances, including the machine that hosts the Domain Administration Server (DAS). The node agent:

- Starts, stops, creates and deletes server instances as instructed by the Domain Administration Server.

- Restarts failed server instances.
- Provides a view of the log files of failed servers.
- Synchronizes each server instance's local configuration repository with the Domain Administration Server's central repository. Each local repository contains only the information pertinent to that server instance or node agent.

The following figure illustrates the overall node agent architecture:

Figure 4-1 Node Agent Architecture



Automatically Created Node Agent

When you install the Application Server, a node agent is created by default with the host name of the machine. This node agent must be manually started on the local machine before it runs.

Node Agents and Server Instance Management

You can create and delete server instances even if the node agent is not running. However, the node agent must be running before you use it to start and stop server instances.

If you stop the node agent, the server instances it manages are stopped too.

Additional Node Agents

A node agent services a single domain. If a machine hosts instances running in multiple domains, it must run multiple node agents.

Node Agent Placeholders

You can create and delete server instances without an existing node agent using a node agent placeholder. The placeholder is a node agent configuration created on the Domain Administration Server (DAS) before the node agent itself is created on the node agent's local system.

NOTE Once you've created a placeholder node agent, use it to create instances in the domain. However, before starting the instances you must create and start the actual node agent locally on the machine where the instances will reside using the `asadmin` command. See "Creating a Node Agent" on page 108 and "Starting a Node Agent" on page 109 for more information.

Deploying Node Agents

You configure and deploy your node agents in one of two ways:

- Online deployment, when you know your topology and have the hardware for your domain before configuring it
- Offline deployment, when you configure domains and server instances before setting up the full environment

Before Deploying Node Agents

Before you deploy node agents:

1. Install the Domain Administration Server.
2. Start the Domain Administration Server.

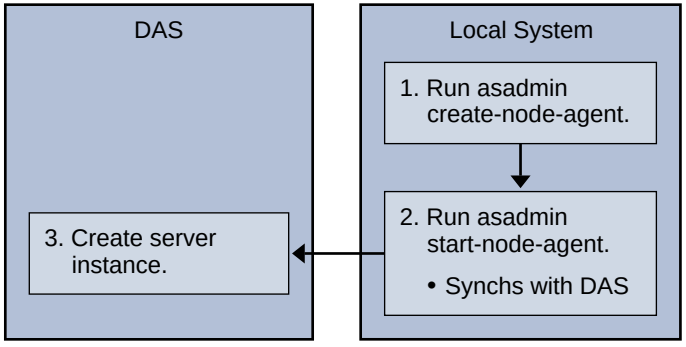
Once the Domain Administration Server is up and running, begin either online or offline deployment.

Online Deployment

To configure the domain if you know the topology and have the hardware for your domain before you start configuring it, use online deployment.

The following figure summarizes the online deployment of node agents:

Figure 4-2 Online Node Agent Deployment



To configure online deployment:

1. Install a node agent on every machine that will host a server instance.

   Use the installer or the `asadmin` command `create-node-agent`. If a machine requires more than one node agent, use the `asadmin` command `create-node-agent` to create them.

   See "Creating a Node Agent" on page 108 for more information.

2. Start the node agents using the `asadmin` command `start-node-agent`.

   When started, a node agent communicates with the Domain Administration Server (DAS). When it reaches the DAS, a configuration for the node agent is created on the DAS. Once the configuration exists, the node agent is viewable in the Admin Console.

   See "Starting a Node Agent" on page 109 for more information.

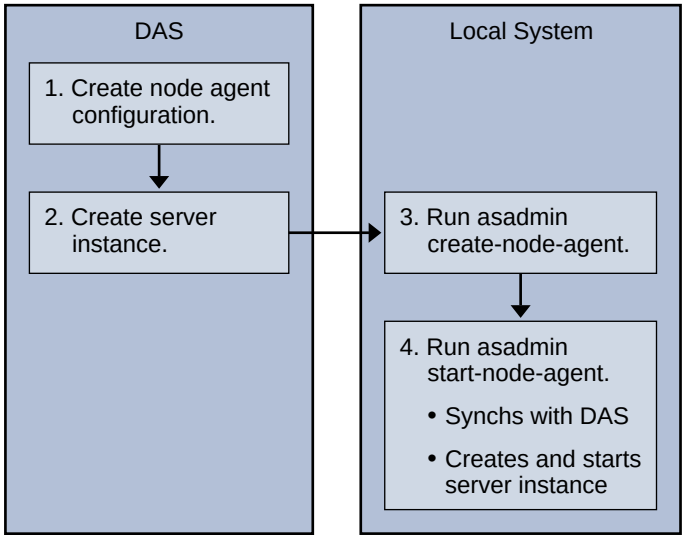
3. Configure the domain, including creating server instances and clusters and deploying applications.

Offline Deployment

The offline approach lets you easily define and rearrange items in the domain before configuring the individual local machines.

The following figure summarizes the offline deployment steps.

Figure 4-3 Offline Node Agent Deployment



To configure the domain and server instances before setting up node agents on local machines (offline configuration):

1. Create placeholder node agents in the Domain Administration Server.

   See "Creating a Node Agent Placeholder" on page 104 for more information.

2. Create server instances and clusters, and deploy applications.

3. Install a node agent on every machine that will host a server instance.

   Use the installer or the `asadmin` command `create-node-agent`. The node agents must have the same names as the placeholder node agents previously created.

   See "Creating a Node Agent" on page 108 for more information.

4. Start the node agents using the `asadmin` command `start-node-agent`.

   When a node agent is started, it binds to the Domain Administration Server and creates any server instances previously associated with the node agent.

   See "Starting a Node Agent" on page 109 for more information.

Node Agent and Domain Administration Server Synchronization

Because configuration data is stored in the Domain Administration Server's repository (the central repository) and also cached on the node agent's local machine, the two must be synchronized.

Node Agent Synchronization

When a node agent is started for the first time, it sends a request to the Domain Administration Server (DAS) for the latest information in the central repository. Once it successfully contacts the DAS and gets configuration information, the node agent is *bound* to that DAS.

If you created a placeholder node agent on the DAS, when the node agent is started for the first time it gets its configuration from the central repository of the DAS.

During its initial start-up, if the node agent is unable to reach the DAS because the DAS is not running, the node agent stops and remains *unbound*.

If changes are made in the domain to the node agent's configuration, they are automatically communicated to the node agent on the local machine while the node agent is running.

If you delete a node agent configuration on the DAS, the next time the node agent synchronizes it stops itself and marks itself as awaiting deletion. Manually delete it using the local `asadmin` command `delete-node-agent`.

Server Instance Synchronization

If you explicitly start a server instance with the Admin Console or `asadmin` tool, the server instance is synchronized with the central repository. If this synchronization fails, the server instance doesn't start.

If a node agent starts a server instance without an explicit request through the Admin Console or the `asadmin` tool, the repository cache for the server instance is not synchronized. The server instance runs with the configuration as stored in its cache.

Synchronizing Large Applications

When your environment contains large applications to synchronize or available memory is constrained, you can adjust the JVM options to limit memory usage. This adjustment reduces the possibility of receiving out of memory errors. The instance synchronization JVM uses default settings, but you can configure JVM options to change them.

Set the JVM options using the `INSTANCE-SYNC-JVM-OPTIONS` property. The command to set the property is:

```
asadmin set domain.node-agent.node_agent_name.property.INSTANCE-SYNC-JVM-OPTIONS="JVM_options"
```

For example:

```
asadmin set domain.node-agent.node0.property.INSTANCE-SYNC-JVM-OPTIONS="-Xmx32m -Xss2m"
```

In this example, the node agent is `node0` and the JVM options are `-Xmx32m -Xss2m`.

For more information on JVM options, see:

http://java.sun.com/docs/hotspot/VMOptions.html

NOTE Restart the node agent after changing the INSTANCE-SYNC-JVM-OPTIONS property, because the node agent is not automatically synchronized when a property is added or changed in its configuration.

Viewing Node Agent Logs

Each node agent has its own log file. If you experience problems with a node agent, see the log file at:

node_agent_dir/*node_agent_name*/agent/logs/server.log.

Sometimes the node agent log instructs you to look at a server's log to get a detailed message about a problem.

The server logs are located at:

node_agent_dir / *node_agent_name* / *server_name* / logs/server.log

The default location for *node_agent_dir* is *install_dir*/nodeagents.

Tasks Available through the Admin Console and asadmin Tool

For node agents, some tasks must be performed locally on the system where the node agent runs, while others can be performed on the Domain Administration Server. Tasks that need to be performed locally are only available through the asadmin tool running on the machine where the node agent resides. Tasks that operate on the Domain Administration Server are available through the Admin Console and through the asadmin tool

The following table summarizes the tasks and where to run them:

Table 4-1 Tasks available through the Admin Console and the asadmin command

Task	Admin Console	asadmin Command
Create node agent placeholder/configuration on Domain Administration Server	Create Node Agent placeholder page	create-node-agent-config
Create node agent	Not available	create-node-agent
Start node agent	Not available	start-node-agent
Stop node agent	Not available	stop-node agent
Delete node agent configuration from Domain Administration Server	Node Agents page	delete-node-agent-config
Delete node agent from local machine	Not available	delete-node-agent

Table 4-1 Tasks available through the Admin Console and the asadmin command

Task	Admin Console	asadmin Command
Edit node agent configuration	Node Agents pages	set
List node agents	Node Agents page	list-node-agents

Admin Console Tasks for Node Agents

Viewing General Node Agent Information

Once a node agent or node agent placeholder has been created, to view its settings:

1. In the tree component, select the Node Agents node.
2. Click the name of a node agent.

   If a node agent already exists but does not appear here, start the node agent on the node agent's host machine using `asadmin start-node-agent`. See "Starting a Node Agent" on page 109 for more information.
3. Check the node agent's host name.

   If the host name is Unknown Host, then the node agent has not made initial contact with the Domain Administration Server (DAS).
4. Check the node agent status.

   Running. The node agent has been properly created and is currently running.

   Not Running. One of the following situations exists:

   - The node agent has been created on the local machine, but has never been started.

- The node agent was started but has been stopped.

Waiting for Rendezvous. The node agent is a placeholder that has never been created on the local machine.

See "Creating a Node Agent" on page 108 and "Starting a Node Agent" on page 109 for more information on creating and starting node agents.

5. Choose whether to start instances on start up.

   Select Yes to start server instances associated with the node agent automatically when the node agent is started. Select No to start the instances manually.

6. Determine whether the node agent has made contact with the Domain Administration Server.

   If the node agent has never made contact with the Domain Administration Server, it has never been successfully started.

7. Manage server instances associated with the node agent.

   If the node agent is running, start or stop an instance by clicking the checkbox next to the instance name and clicking Start or Stop.

Creating a Node Agent Placeholder

Because the node agent must be created locally on the machine hosting the node agent, through the Admin Console you can only create a placeholder for a node agent. This placeholder is a node agent configuration for which a node agent does not yet exist.

After creating a placeholder, use the `asadmin` command `create-node-agent` on the machine hosting the node agent to complete the creation. For more information, see "Creating a Node Agent" on page 108.

1. In the tree component, select the Node Agents node.
2. On the Node Agents page, click New.
3. On the Current Node Agent Placeholder page, enter a name for the new node agent.

   The name must be unique across all node agent names, server instance names, cluster names, and configuration names in the domain.

4. Click OK.

   The placeholder for your new node agent is listed on the Node Agents page.

Equivalent `asadmin` command: `create-node-agent-config`.

Deleting a Node Agent Configuration

Through the Admin Console you can only delete the node agent configuration from the domain. You cannot delete the actual node agent. To delete the node agent itself, run the `asadmin` command `delete-node-agent` on the node agent's local machine. For more information, see "Deleting a Node Agent" on page 109.

Before deleting the node agent configuration, the node agent must be stopped and it must not have any associated instances. To stop a node agent, use the `asadmin` command `stop-node-agent`. See "Stopping a Node Agent" on page 109 for more information.

1. In the tree component, select the Node Agents node.
2. On the Node Agents page, select the checkbox next to the node agent to be deleted.
3. Click delete.

Equivalent `asadmin` command: `delete-node-agent-config`.

Editing a Node Agent Configuration

To edit a node agent configuration:

1. In the tree component, expand the Node Agents node.
2. Select the node agent configuration to edit.
3. On the Node Agents General Information page, choose whether to start the agent's server instances when the agent is started. You can also manually start and stop instances from this page.

If this configuration is for a placeholder node agent, when you create the actual node agent using `asadmin create-node-agent`, it picks up this configuration. For information on creating a node agent, see "Creating a Node Agent" on page 108.

If this configuration is for an existing node agent, the node agent configuration information is synchronized automatically.

Editing a Node Agent Realm

Set an authentication realm for users connecting to the node agent. Only administration users should have access to the node agent.

1. In the tree component, expand the Node Agents node.
2. Select the node agent configuration to edit.
3. Click the Auth Realm tab.
4. On the Node Agents Edit Realm page, enter a realm.

   The default is `admin-realm`, created when you create the node agent. To use a different realm, replace the realms in *all* the components controlled by the domain or the components won't communicate properly.
5. In the Class Name field, specify the Java class that implements the realm.
6. Add any required properties.

   Authentication realms require provider-specific properties, which vary depending on what a particular implementation needs.

Editing the Node Agent's Listener for JMX

The node agent uses JMX to communicate with the Domain Administration Server. Therefore it must have a port to listen on for JMX requests, and other listener information.

1. In the tree component, expand the Node Agents node.
2. Select the node agent configuration to edit.
3. Click the JMX tab.
4. In the Address field, type 0.0.0.0 if the listener listens on all IP addresses for the server, using a unique port value. Otherwise, type a valid IP address for the server.
5. In the Port field, type the port value for the node agent's JMX connector to listen on. If the IP address is 0.0.0.0, the port number must be unique.
6. In the JMX Protocol field, type the protocol that the JMX connector supports.

   The default is rmi_jrmp.

7. Click the checkbox next to Accept All Addresses to allow a connection to all IP addresses.

   The node agent listens on a specific IP address associated to a network card or listens on all IP addresses. Accepting all addresses puts the value `0.0.0.0` in the “listening host address” property.

8. In the Realm Name field, type the name of the realm that handles authentication for the listener.

In the Security section of this page, 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.

   Security is enabled by default.

2. To have clients 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) to enable on the listener. You must check either SSL3 or TLS, or both protocols.

   b. Check the cipher suite used by the protocol(s). To enable all cipher suites, check All Supported Cipher Suites.

5. Click Save.

Tasks for Node Agents in asadmin Tool

Creating a Node Agent

To create a node agent, run the `asadmin` command `create-node-agent` locally on the machine on which the node agent runs.

For example:

```
$ asadmin create-node-agent --host myhost --port 4849 ---user admin
nodeagent1
```

where `myhost` is your Domain Administration Server (DAS) hostname, `4849` is your DAS port number, `admin` is your DAS user, and `nodeagent1` is the name of the node agent you are creating.

The default name for a node agent is the host name on which the node agent is created.

If you've already created a node agent placeholder, use the same name as the node agent placeholder to create the associated node agent. If you have not created a node agent placeholder, and the DAS is up and reachable, the `create-node-agent` command also creates a node agent configuration (placeholder) on the DAS.

For a complete description of the command syntax, see the online help for the command.

NOTE In the following situations, you must specify a DNS-reachable hostname:

1. If domains cross subnet boundaries (that is, the node agent and the Domain Administration Server (DAS) are in different domains, for example, sun.com and java.com)
2. If using a DHCP machine with a host name not registered in the DNS

Specify a DNS-reachable hostname by explicitly specifying the host name for the domain and the node agent when you create them:

`create-domain --domainproperties domain.hostName=`*`DAS-host-name`*

`create-node-agent --host` *`DAS-host-name`* `--agentproperties remoteclientaddress=`*`node-agent-host-name`*

Another solution is to update the `hosts` hostname/IP resolution file specific to the platform so the hostname resolves to the correct IP address. However, when reconnecting using DHCP you might get assigned a different IP address. In that case, you must update the host resolution files on each server.

Starting a Node Agent

Before a node agent can manage server instances, it must be running. Start a node agent by running the `asadmin` command `start-node-agent` locally on the system where the node agent resides.

For example:

```
$ asadmin start-node-agent --user admin nodeagent1
```

where `admin` is your admin user, and `nodeagent1` is the node agent you are starting.

For a complete description of the command syntax, see the online help for the command.

Stopping a Node Agent

Run the `asadmin` command `stop-node-agent` on the system where the node agent resides to stop a running node agent. The `stop-node-agent` command stops all server instances that the node agent manages.

For example:

```
$ asadmin stop-node-agent nodeagent1
```

where `nodeagent1` is your node agent.

For a complete description of the command syntax, see the online help for the command.

Deleting a Node Agent

Run the `asadmin` command `delete-node-agent` on the system where the node agent resides to delete the node agent files.

Before deleting a node agent, the node agent must be stopped. You can also delete a node agent if it has never been started, or never successfully able to contact the Domain Administration Server (that is, if it is still unbound).

For example:

```
$ asadmin delete-node-agent nodeagent1
```

where `nodeagent1` is your node agent.

For a complete description of the command syntax, see the online help for the command.

When deleting a node agent, you must also delete its configuration from the Domain Administration Server using either the Admin Console or the `asadmin delete-node-agent-config` command

Chapter 5

Deploying Applications

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

About Deployment

The Deployment Life Cycle

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

1. Initial Deployment

   Before deploying an application or module, start the domain.

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

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

   You can also deploy to the domain using the `asadmin deploy` command, not the Admin Console. If you deploy the application or module only to the domain, it exists in the domain's central repository, but no server instances or clusters reference it until you add references, as explained in Step 3.

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

2. Enabling or Disabling

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

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

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

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

4. Redeployment

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

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

   For production environments, use rolling upgrade, which upgrades application without interruption in service. For more information, see "About Rolling Upgrades" on page 88.

5. Undeployment

   To uninstall an application or module, undeploy it.

Types of J2EE Archive Files

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

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

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

Naming Conventions

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

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

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

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

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

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

Admin Console Tasks for Deploying Applications

Deploying an Enterprise Application

An enterprise application is packaged in an EAR file, a type of archive file that contains any type of J2EE stand-alone modules, 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 location of the EAR file to deploy.

   The server machine is the host that is running the application server domain administration server. The client machine is the host on which you are viewing the Admin Console through a browser.

a. If the file resides on or is accessible from the client machine, click the radio button to specify a package file to upload to the Application Server.

Click Browse to browse to the file, or type the full path to the file.

b. If the file resides on the server machine, or to deploy an unpackaged application from an exploded directory, click the radio button to specify a package file or a directory path that must be accessible from the server.

Type the full path name to the file or directory. Deploying from an exploded directory is for advanced developers and is not recommended for production environments.

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, either retain the default name, which is the prefix of the file name, or type another name. (The default name appears if you chose to upload a file.) The application name must be unique.

b. By default, an application is available as soon as it is deployed. To disable the application so that is unavailable after deployment, select the Disabled radio button.

c. If the application has already been deployed, select the Redeploy checkbox to redeploy it; otherwise you see an error. You can also choose a different application name and deploy it under a new name.

d. To verify the structure and contents of the file before deployment, select the Verifier checkbox. Verification of large applications can be time-consuming. Verify the file if you suspect it is corrupt or non-portable.

e. To precompile JSP pages, select the JSPs checkbox. If you do not select this checkbox, the JSP pages are compiled at runtime when they are first accessed. Because compilation is often time-consuming, in a production environment select this checkbox.

f. Choose a high availability setting.

To enable high availability for the application, select the Availability checkbox. If availability is enabled for an application, it must also be enabled at all higher levels (named configuration and web container or EJB container) as well.

g. Choose the targets to which to deploy the application.

From the list of available targets, choose the target or targets and click Add. Targets can be clusters or stand-alone server instances. If you do not select a target, the application is deployed to the default server instance `server`.

If you are redeploying, don't select targets. Anything you select here is ignored. Any target clustered or stand-alone server instance that references the deployed application automatically references the new, redeployed application if dynamic reconfiguration is enabled for the cluster or stand-alone instance. For more information about how to redeploy applications without interruption of service, see "Upgrading an Application" on page 88.

h. Choose whether to generate RMI stubs.

If you choose to generate RMI stubs, static RMI-IIOP stubs are generated and put into the `client.jar`.

7. Click OK to deploy the application.

Equivalent `asadmin` command: `deploy`

Deploying a Web Application

A Web application is packaged in a WAR file, a type of archive 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 location of the WAR file to deploy.

The server machine is the host that is running the application server domain administration server. The client machine is the host on which you are viewing the Admin Console through a browser.

a. If the file resides on or is accessible from the client machine, click the radio button to specify a package file to upload to the Application Server.

Click Browse to browse to the file, or type the full path to the file.

b. If the file resides on the server machine, or to deploy an unpackaged application from an exploded directory, click the radio button to specify a package file or a directory path that must be accessible from the server.

Type the full path name to the file or directory. Deploying from an exploded directory is for advanced developers and is not recommended for production environments.

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, either retain the default name, which is the prefix of the file name, or type another name. (The default name appears if you chose to upload a file.) 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. By default, an application is available as soon as it is deployed. To disable the application so that is unavailable after deployment, select the Disabled radio button.

d. If the application has already been deployed, select the Redeploy checkbox to redeploy it; otherwise you see an error. You can also choose a different application name and deploy it under a new name.

e. To verify the structure and contents of the file before deployment, select the Verifier checkbox. Verification of large applications is often time-consuming. Verify the file if you suspect it is corrupt or non-portable.

f. To precompile JSP pages, select the JSPs checkbox. If you do not select this checkbox, the JSP pages are compiled at runtime when they are first accessed. Because compilation is often time-consuming, in a production environment select this checkbox.

g. Choose a high availability setting.

To enable high availability for the application, select the Availability checkbox. If availability is enabled for an application, it must also be enabled at all higher levels (named configuration and web container or EJB container) as well.

h. Choose the targets to which to deploy the application.

   From the list of available targets, choose the target or targets and click Add. Targets can be clusters or stand-alone server instances. If you do not select a target, the application is deployed to the default server instance `server`.

   If you are redeploying, don't select targets. Anything you select here is ignored. Any target clustered or stand-alone server instance that references the deployed application automatically references the new, redeployed application if dynamic reconfiguration is enabled for the cluster or stand-alone instance. For more information about how to redeploy applications without interruption of service, see "About Rolling Upgrades" on page 88.

i. Choose whether to generate RMI stubs.

   If you choose to generate RMI stubs, static RMI-IIOP stubs are generated and put into the `client.jar`.

7. Click OK to deploy the application.

Equivalent `asadmin` command: `deploy`

Launching a Deployed Web Application

After deploying an application, you can launch it from the Admin Console.

1. In the tree component, expand the Applications node.
2. Click Web Applications.
3. Click the Launch link for the web application.
4. Click a link on the Web Application Links page to launch the application.

   The server and HTTP listener must be running for the application to launch.

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 location of the JAR file to deploy.

   The server machine is the host that is running the application server domain administration server. The client machine is the host on which you are viewing the Admin Console through a browser.

   a. If the file resides on or is accessible from the client machine, click the radio button to specify a package file to upload to the Application Server.

      Click Browse to browse to the file, or type the full path to the file.

   b. If the file resides on the server machine, or to deploy an unpackaged application from an exploded directory, click the radio button to specify a package file or a directory path that must be accessible from the server.

      Type the full path name to the file or directory. Deploying from an exploded directory is for advanced developers and is not recommended for production environments.
5. Click Next to display the Deploy EJB Module page.
6. On the Deploy EJB Module page, specify the settings for the module.

   a. In the Application Name field, either retain the default name, which is the prefix of the file name, or type another name. (The default name appears if you chose to upload a file.) The application name must be unique.

   b. By default, an module is available as soon as it is deployed. To disable the module so that is unavailable after deployment, select the Disabled radio button.

   c. If the module has already been deployed, select the Redeploy checkbox to redeploy it; otherwise you see an error. You can also choose a different application name and deploy it under a new name.

   d. To verify the structure and contents of the file before deployment, select the Verifier checkbox. Verification of large applications can be time-consuming. Verify the file if you suspect it is corrupt or non-portable.

   e. Choose a high availability setting.

      To enable high availability for the module, select the Availability checkbox. If availability is enabled for an module, it must also be enabled at all higher levels (named configuration and web container or EJB container) as well.

f. Choose the targets to which to deploy the module.

From the list of available targets, choose the target or targets and click Add. Targets can be clusters or stand-alone server instances. If you do not select a target, the module is deployed to the default server instance `server`.

If you are redeploying, don't select targets. Anything you select here is ignored. Any target clustered or stand-alone server instance that references the deployed module automatically references the new, redeployed module if dynamic reconfiguration is enabled for the cluster or stand-alone instance. For more information about how to redeploy modules without interruption of service, see "About Rolling Upgrades" on page 88.

g. Choose whether to generate RMI stubs.

If you choose to generate RMI stubs, static RMI-IIOP stubs are generated and put into the `client.jar`.

7. Click OK 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 archive 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 Modules page, click Deploy.

4. On the Deployment page, specify the location of the RAR file to deploy.

The server machine is the host that is running the application server domain administration server. The client machine is the host on which you are viewing the Admin Console through a browser.

a. If the file resides on or is accessible from the client machine, click the radio button to specify a package file to upload to the Application Server.

Click Browse to browse to the file, or type the full path to the file.

b. If the file resides on the server machine, or to deploy an unpackaged module from an exploded directory, click the radio button to specify a package file or a directory path that must be accessible from the server.

Type the full path name to the file or directory. Deploying from an exploded directory is for advanced developers and is not recommended for production environments.

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

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

a. In the Application Name field, either retain the default name, which is the prefix of the file name, or type another name. (The default name appears if you chose to upload a file.) The application name must be unique.

b. In the Thread Pool Id field, specify the thread pool for the resource adapter you are deploying.

By default, the Sun Java System Application Server services work requests from all resource adapters from its default thread pool. Use this field to associate a specific user-created thread pool to service work requests from a resource adapter.

c. By default, a module is available as soon as it is deployed. To disable the module so that is unavailable after deployment, select the Disabled radio button.

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 module has already been deployed, select the Redeploy checkbox to redeploy it; otherwise you see an error. You can also choose a different application name and deploy it under a new name.

e. To verify the structure and contents of the file before deployment, select the Verifier checkbox. Verification of large applications is often time-consuming. Verify the file if you suspect it is corrupt or non-portable.

f. If the resource adapter has additional properties specified, they are displayed.

Use the table to modify the default values of these properties.

g. Choose the targets to which to deploy the module.

From the list of available targets, choose the target or targets and click Add. Targets can be clusters or stand-alone server instances. If you do not select a target, the module is deployed to the default server instance `server`.

If you are redeploying, don't select targets. Anything you select here is ignored. Any target clustered or stand-alone server instance that references the deployed module automatically references the new, redeployed module if dynamic reconfiguration is enabled for the cluster or stand-alone instance. For more information about how to redeploy modules without interruption of service, see "About Rolling Upgrades" on page 88.

7. Click OK 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
- Shut down

Lifecycle modules are not part of the J2EE specification, but are an enhancement to the Sun Java System Application Server.

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.

If you don't specify the classpath, you must unpack the classes in *application_server_home*/domains/*domain*/applications/lifecycle-module/*module_name*. If you specify a classpath, nothing else is required.

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 are already deployed. By default, if a load fails, the server continues the start-up operation. To prevent the server from starting up when a load fails, select the On Load Failure checkbox.

f. By default, a module is available as soon as it is deployed. To disable the module so that is unavailable after deployment, select the Disabled radio button.

g. Choose the targets to which to deploy the module.

From the list of available targets, choose the target or targets and click Add. Targets can be clusters or stand-alone server instances. If you do not select a target, the module is deployed to the default server instance `server`.

If you are redeploying, don't select targets. Anything you select here is ignored. Any target clustered or stand-alone server instance that references the deployed module automatically references the new, redeployed module if dynamic reconfiguration is enabled for the cluster or stand-alone instance. For more information about how to redeploy modules without interruption of service, see "About Rolling Upgrades" on page 88.

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 location of the JAR file to deploy.

   The server machine is the host that is running the application server domain administration server. The client machine is the host on which you are viewing the Admin Console through a browser.

   a. If the file resides on or is accessible from the client machine, click the radio button to specify a package file to upload to the Application Server.

      Click Browse to browse to the file, or type the full path to the file.

   b. If the file resides on the server machine, or to deploy an unpackaged module from an exploded directory, click the radio button to specify a package file or a directory path that must be accessible from the server.

      Type the full path name to the file or directory. Deploying from an exploded directory is for advanced developers and is not recommended for production environments.
5. Click Next to display the Deploy Application Client Module page.
6. On the Deploy Application Client Module page, specify the settings for the module.

   a. In the Application Name field, either retain the default name, which is the prefix of the file name, or type another name. (The default name appears if you chose to upload a file.) The application name must be unique.

   b. If the module has already been deployed, select the Redeploy checkbox to redeploy it; otherwise you see an error. You can also choose a different application name and deploy it under a new name.

   c. To verify the structure and contents of the file before deployment, select the Verifier checkbox. Verification of large applications can be time-consuming.Verify the file if you suspect it is corrupt or non-portable.

d. Choose the targets to which to deploy the module.

From the list of available targets, choose the target or targets and click Add. Targets can be clusters or stand-alone server instances. If you do not select a target, the module is deployed to the default server instance `server`.

If you are redeploying, don't select targets. Anything you select here is ignored. Any target clustered or stand-alone server instance that references the deployed module automatically references the new, redeployed module if dynamic reconfiguration is enabled for the cluster or stand-alone instance. For more information about how to redeploy modules without interruption of service, see "About Rolling Upgrades" on page 88.

e. Choose whether to generate RMI stubs.

If you choose to generate RMI stubs, static RMI-IIOP stubs are generated and put into the `client.jar`.

7. Click OK to deploy the module.

For the client-side routines:

- Typically, the application provider ships a JAR file containing the client-side routines.
- The application provider gets the client-side stubs by specifying the `--retrieve` option of the `asadmin deploy` command.

Equivalent `asadmin` command: `deploy`

Admin Console Tasks for Listing, Undeploying, and Enabling Applications

Listing Deployed Applications

To list 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 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, EJB Modules and Connector Modules contain subcomponents. For example, a Web application might contain one or more servlets.

To list the subcomponents of an application or module:

1. In the tree component, expand the Applications node.
2. Expand the node for the type of application or module for which to view descriptors.
3. Select the node for the deployed application or module.
4. On the Application or Module page, note the contents of the Sub Components table.

Equivalent `asadmin` command: `list-sub-components`

Viewing Module Descriptors of Deployed Applications

For Enterprise Applications, Web Applications, EBJ Modules, Connector Modules, and App Client Modules, you can view the module deployment descriptors.

1. In the tree component, expand the Applications node.
2. Select the node for the type of application or module for which to view descriptors.

3. Select the node for the deployed application or module.
4. Select the Descriptor tab.
5. To see the text of the descriptor file, click the file name.

   The page displays the file contents. This information is read-only.

Undeploying an Application

Undeploying an application or module uninstalls it from the domain and removes references to it from all instances.

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 or module 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 is accessible by clients. If it is disabled, it is still deployed but is not accessible by clients. By default, when you deploy an application or module, it is enabled because the Enable on All Targets radio button is selected by default.

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 checkbox next to a deployed application or module.
4. Click Enable or Disable.

   These buttons enable or disable the application on all targets.

To enable an application on a single target

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

2. Expand the node for the application type.
3. Select the node for the application.
4. Click the Targets tab.
5. Select the checkbox next to the deployed application or module.
6. Click Enable or Disable.

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

Managing Application Targets

After deploying an application or module, manage the server instances and clusters that reference it by managing targets.

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.
4. Select the Targets tab.
5. To enable or disable an application on a specific target instance or cluster, click the checkbox next to the target and click Enable or Disable.
6. To add or delete targets for the application, choose Manage Targets.
7. Add or remove targets and click OK.

   The application is now available on the revised list of targets.

Equivalent `asadmin` commands: `create-application-ref`, `delete-application-ref`.

Deploying on Additional Virtual Servers

After deploying an application or module to a target server instances or clusters, you can associate it with additional virtual servers.

1. From the deployed application or module's Target page, click the Manage Virtual Servers link next to the target.
2. Add or remove virtual server targets from the list of available virtual servers.
3. Click OK.

Redeploying to Multiple Targets

If an application is deployed to multiple targets (stand-alone server instances or clusters), you have two ways of redeploying to multiple targets. Use one of the following methods to make sure all server instances that reference an application receive the most recent version.

Development Environment

In a development environment, simply redeploy the application. The application is redeployed to the domain, and all targets that reference it automatically receive the new version if dynamic reconfiguration is enabled for the target server instances. Dynamic reconfiguration is enabled by default. If dynamic reconfiguration is not enabled for a server instance, it continues to use the old version until the server instance is restarted.

Production Environment

In a production environment, follow the steps detailed in "About Rolling Upgrades" on page 88.

Enabling and Disabling Dynamic Reloading

If dynamic reloading is enabled, the server periodically checks for changes in a deployed application and automatically reloads the application with the changes. Changes are signaled by a date change to a file called `.reload` that you create manually. The applications must be installed in *server_root*`/domain/domain1/applications/`*j2ee-module_or_j2ee-apps*`/`*app_or_module_name*

For example:
`AppServer/domain/domain1/applications/j2ee-module/webapps-simple`

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.

NOTE Dynamic reloading is only available for the default server instance.

Dynamic reloading is intended for development environments. It is incompatible with session persistence, a production environment feature. Do not enable session persistence if dynamic deployment is enabled.

To configure dynamic reloading:

1. In the tree component, expand the Stand-Alone Instances node.
2. Click server (Admin Server).
3. Click Advanced.
4. On the Applications Configuration page, configure the following:
    - Reload: Enable or disable dynamic reloading with the Enabled checkbox.
    - Reload Poll Interval: Specify how often the server checks for changes in the deployed applications.
    - Admin Session Timeout: Specify the amount of time before the Admin Session times out and you have to log in again.

After configuring the system to use dynamic reloading, for every application to be reloaded dynamically, create a file called .reload and put it in the application's directory. The file does not have any content. When you change the application, change the date of the file (for example, using the UNIX `touch` command), and the changes are reloaded automatically.

Deployment Methods for Developers

Using Auto Deploy

The auto deploy feature enables you to deploy a pre-packaged application or module by copying it to the *domain_root_dir*/*domain_dir*/`autodeploy` directory.

For example, copy a file named `hello.war` to the *domain_root_dir*/`domain1/autodeploy` directory. To undeploy the application, remove the `hello.war` file from the `autodeploy` directory.

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

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

NOTE Auto deploy is only available for the default server instance.

To configure the auto deploy feature:

1. In the tree component, expand the Stand-Alone Instances node.
2. Click server (Admin Server).
3. Click Advanced.
4. On the Applications Configuration page, configure the following:
   a. Enable or disable auto deploy by selecting or deselecting the Enabled checkbox.
   b. In the Auto Deploy Poll Interval field, specify how often the server checks the auto deploy directory for application or module files. Changing the poll interval does 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.

      The default is a directory called `autodeploy` in the server instance's root directory. By default a variable is used to eliminate the need to manually change the directory for multiple server instances. For more information on these variables, see Setting Domain Attributes.
   d. To run the verifier before deployment, select the Verifier Enabled checkbox. The verifier examines the structure and content of the file. Verification of large applications is often time-consuming.
   e. To precompile JSP pages, select the JSPs checkbox. If you do not select this checkbox, the JSP pages are compiled at runtime when they are first accessed. Because compilation is often time-consuming, in a production environment select this checkbox.

Deploying an Unpackaged Application From a Directory

This feature is for advanced developers.

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

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

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

If your environment is configured to use dynamic reloading, you can also dynamically reload applications deployed from the directory. For more information, see "Enabling and Disabling Dynamic Reloading" on page 130.

To deploy an unpackaged application from a directory:

1. In the Admin Console, begin the deployment process. See "Deploying a Web Application" on page 117.
2. On the Deployment page, specify the following:
   a. Click the radio button to specify a package file or a directory path that must be accessible from the server.
   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 an JAR file that contains only the deployment descriptors that are specific to the Application Server. These deployment descriptors, for example `sun-application.xml`, are described in the *Application Server Developer's Guide*. The deployment plan is part of the implementation of *JSR 88: J2EE Application Deployment*. Use a deployment plan to deploy an application or module that does not contain the deployment descriptors that are specific to the Application Server.

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

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

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

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

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

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

Chapter 6

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

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, the administrator creates a JDBC resource for each database accessed by the applications deployed in a domain. (However, more than one JDBC resource can be created for a database.)

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

JDBC Connection Pools

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

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

The properties of connection pools can 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 database:

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

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

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

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

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

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

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

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

Setting Up Database Access

General Steps for Setting Up Database Access

1. Install a supported database product. For a list of database products supported by the Application Server, 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, do either of the following:

- Make the driver accessible to the common class loader.
  - 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/ext directory.
  - Restart the domain.
- Make the driver accessible to the system class loader.
  - In the Admin Console's tree view, select Configurations.
  - Select the desired configuration (for example, default-config).
  - Select JVM Settings.
  - On the JVM Settings page, click the Path Settings tab.
  - In the Classpath Suffix field, enter the fully-qualified path name for the driver's JAR file.
  - Click Save.
  - Restart the server.

About JDBC Connection Pools

Creating a JDBC Connection Pool

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

Before creating the pool, you must first install and integrate the JDBC driver.

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

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

To create a JDBC connection pool:

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

      Specify this name when creating a JDBC resource.

   b. Select an entry from the Resource Type combo box.
   c. Select an entry from the Database Vendor combo box.
6. Click Next.
7. 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 specified in the previous page, then the value of the DataSource Class Name field is provided.
8. Click Next.
9. 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, retain the default values.

It is most convenient to change these settings at a later time. See Editing a JDBC Connection Pool.

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

10. Click Finish.

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

Editing a JDBC Connection Pool

The Edit JDBC Connection Pool page provides the means 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 Resources node.

2. Under Resources, expand the JDBC node.

3. Under JDBC, expand the Connection Pools node.

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

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

See the following sections for explanations of the settings that might change.

6. Click Save.

General Settings

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

Table 6-1 General Settings for a JDBC Connection Pool

Parameter	Description
DataSource Class Name	The vendor-specific class name that implements the DataSource/ConnectionPoolDataSource/XADataSource APIs. This class is in the JDBC driver.

Table 6-1 General Settings for a JDBC Connection Pool

Parameter	Description
Resource Type	Choices include javax.sql.DataSource (local transactions only), javax.sql.XADataSource (global transactions), and java.sql.ConnectionPoolDataSource (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 6-2 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 delays connection recycling; making it too small will be less efficient.
Idle Timeout	The maximum time in seconds that a connection can remain idle in the pool. After this time expires, the connection is removed from the pool.
Max Wait Time	The amount of time the application requesting a connection will wait before getting a connection timeout. Because the default wait time is long, the application might appear to hang indefinitely.

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 incurs additional overhead and slightly reduces performance.

Table 6-3 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, metadata, and table. • auto-commit and metadata - The application server validates a connection by calling the con.getAutoCommit() and con.getMetaData() methods. However, because many JDBC drivers cache the results of these calls, they do not always provide reliable validations. Check with the driver vendor to determine whether these calls are cached or not. • table - The application queries a database table that is specified. The table must exist and be accessible, but it doesn't require any rows. Do not use an existing table that has a large number of rows or a table that is already frequently accessed.
Table Name	If you selected table from the Validation Method combo box, then specify the name of the database table here.
On Any Failure	If you select the checkbox labelled Close All Connections, if a single connection fails, then the application server closes all connections in the pool and then re-establish them. If you do not select the checkbox, then individual connections are 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 6-4 Transaction Isolation Settings for a JDBC Connection Pool

Parameter	Description
Transaction Isolation	Makes it possible to select the transaction isolation level for the connections of this pool. If left unspecified, the connections operate with default isolation levels provided by the JDBC driver.

Table 6-4 Transaction Isolation Settings for a JDBC Connection Pool

Parameter	Description
Guaranteed Isolation Level	Only applicable if the isolation level has been specified. If you select the Guaranteed checkbox, then all connections taken from the pool have the same isolation level. For example, if the isolation level for the connection is changed programatically (with `con.setTransactionIsolation`) when last used, this mechanism changes the status back to the specified isolation level.

Properties

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

Verifying Connection Pool Settings

To verify the connection pool settings:

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

   The Admin Console attempts to connect to the database. If an error message displays, check to see if the database server was restarted.

Deleting a JDBC Connection Pool

1. In the tree component, expand the Resources node.
2. Under Resources, expand the JDBC node.
3. Under JDBC, select the Connection Pools node.
4. On the Connection Pools page, select the checkbox for the pool to be deleted.
5. Click Delete.

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

About JDBC Resources

Creating a JDBC Resource

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

To create a JDBC resource:

1. In the tree component, expand the Resources node.
2. Under Resources, expand the JDBC node.
3. Under JDBC, select the JDBC Resources node.
4. On the JDBC Resources page, click New.
5. On the Create JDBC Resource page, specify the resource's settings:
    a. In the JNDI Name field, type a unique name. By convention, the name begins 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 to be associated with the new JDBC resource.
    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.
    d. In the Description field, type a short description of the resource.
    e. In the Targets section, specify the targets (clusters and standalone server instances) on which the resource is available. Select the desired target on the left, and click Add to add it to the list of selected targets.
6. Click OK.

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

Editing a JDBC Resource

1. In the tree component, expand the Resources node.
2. Under Resources, expand the JDBC node.

3. Under JDBC, expand the JDBC Resources node.
4. Select the node for the JDBC resource to be edited.
5. On the Edit JDBC Resource page, it is possible to perform these tasks:
   a. From the Pool Name combo box, select a different connection pool.
   b. In the Description field, change the short description of the resource.
   c. Select or deselect the checkbox to enable or disable the resource.
   d. Select the Targets tab to change the targets (clusters and standalone server instances) on which the resource is available.

      Select the checkbox for an existing target in the list, then click Enable to enable the resource for that target or Disable to disable the resource for that target.

      Click Manage Targets to add or remove targets to the list. In the Manage Targets page, select the desired target in the Available list on the left, and click Add to add it to the list of selected targets. Click Remove to remove a target from the Selected list.

      Click OK to save the changes to the available targets.
6. Click Save to apply the edits.

Deleting a JDBC Resource

1. In the tree component, expand the Resources node.
2. Under Resources, expand the JDBC node.
3. Under JDBC, select the JDBC Resources node.
4. On the JDBC Resources page, select the checkbox for the resource to be deleted.
5. Click Delete.

- Enabling and Disabling a JDBC Resource

Enabling and Disabling a JDBC Resource

1. In the tree component, expand the JDBC Resources node or expand the Standalone Instances to select the Server Instance node Resource tab.

2. On the Resources page, select the checkbox for the resource to be enabled or disabled.
3. Click Enable or Disable.

About Persistence Manager Resources

- Creating a Persistence Manager Resource

Creating a Persistence Manager Resource

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

To create a persistence manager resource:

1. In the tree component, expand the Resources node.
2. Under Resources, select the Persistence Managers node.
3. On the Persistence Managers page, click New.
4. On the Create Persistence Manager page, specify these settings:
    a. In the JNDI Name field, type a unique name, for 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 implementation.
    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 persistence manager resource is enabled. To disable it, deselect the Enabled check box.
    e. In the Targets section, specify the targets (clusters and standalone server instances) on which the resource is available. Select the desired target on the left, and click Add to add it to the list of selected targets.
5. Click OK.

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

Editing a Persistence Manager Resource

To edit an existing persistence manager resource property:

1. From the Edit Persistence Manager Properties tab, select the Add Property button.

   A new row is added to the Additional Properties table.

2. Add the desired property and value.

Managing Resource Targets

To manage a resource target:

1. Select the Targets tab to change the targets (clusters and standalone server instances) where the resource resides.
2. Select the checkbox for an existing target in the list, then click Enable to enable the resource for that target or Disable to disable the resource for that target.
3. Click Manage Targets to add or remove targets to the list. In the Manage Targets page, select the desired target in the Available list on the left, and click Add to add it to the list of selected targets. Click Remove to remove a target from the Selected list.
4. Click OK to save the changes to the available targets.
5. Click Save.

Deleting a Persistence Manager Resource

1. In the tree component, expand the Persistence Managers node.
2. Select the Persistence Managers node.
3. On the Persistence Managers page, select the checkbox for the persistence manager that you want to delete.
4. Click Delete.

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

Enabling and Disabling a Persistence Manager Resource

1. In the tree component, expand the Persistence Managers node .
2. Select the checkbox for the resource to be enabled or disabled.
3. Click Enable or Disable.

Chapter 7

Configuring Availability and Session Persistence

This chapter describes how to configure session persistence and availability in the Sun Java™ System Application Server Enterprise Edition environment. It contains the following sections:

- About Availability and Session Persistence
- Admin Console Tasks for Configuring Availability

About Availability and Session Persistence

- Why Session Persistence Is Needed
- Overview of Session Persistence Configuration
- Levels of Availability
- Availability of Single Sign-on in the HTTP Session State
- Sample Applications

Why Session Persistence Is Needed

As an application session proceeds, there is often data that is part of a session that is not stored in a traditional database. An example of such data is the content of a shopping cart. Sun Java System Application Server provides the capability to save, or persist, this session data in a repository so that if an application server instance experiences a failure, the session state can be recovered and the session can continue without information loss.

In J2EE applications, session data is typically stored in HTTP sessions or stateful session bean (SFSB) sessions. Sun Java System Application Server supports persistence of the state of both HTTP sessions and SFSB sessions. Failover of certain J2EE object references that are stored in both HTTP sessions and SFSB sessions is also supported; see the *Developer's Guide*.

The high-availability database (HADB) bundled with the Sun Java System Application Server works as the persistence store to provide high availability for session data.

Overview of Session Persistence Configuration

For successful session persistence configuration, ensure that these steps are followed in the order in which they are presented, because some later steps have one or more previous steps as prerequisites.

1. Create an HADB database for the cluster. See the description of the `configure-ha-cluster` command in the *Reference Manual*.
2. Set up HTTP load balancing for the cluster. See Chapter 3, "Configuring Load Balancing and Failover."
3. Enable availability for the application server instances and web or EJB containers that should support session persistence, and configure the session persistence settings. Choose one of these approaches:
   - See "Admin Console Tasks for Configuring Availability" on page 153.
   - See the description of the `configure-ha-persistence` command in the *Reference Manual*.
4. If you are *not* enabling availability, you can change the file system session store for SFSBs if desired. See "Configuring the SFSB Session Store When Availability Is Disabled" on page 153.
5. Restart each server instance in the cluster.
6. Enable availability for any specific SFSB that requires it, and select methods for which checkpointing the session state is necessary. See the *Developer's Guide*.
7. Make distributable each web module that should be highly available. See the *Developer's Guide*.
8. Enable availability for a J2EE application, web module, or EJB module during deployment. In the Administration Console, check the Availability Enabled box, or use the `deploy` command with the `--availabilityenabled` option set to `true`.

NOTE Session persistence is incompatible with dynamic deployment, dynamic reloading, and auto deployment. These deployment features are intended for development environments, not production environments. For information about how to disable these features, see Chapter 5, “Deploying Applications.”

NOTE If the instance is currently serving requests, quiesce the instance before restarting it so that the instance gets enough time to serve the requests it is handling. For more information, see Disabling (Quiescing) a Server Instance or Cluster.

Levels of Availability

Availability can be enabled at five different levels:

1. The server instance, enabled by default
2. The web or EJB container, enabled by default
3. The application, disabled by default
4. The stand-alone web or EJB module, disabled by default
5. The SFSB, disabled by default

For availability to be enabled at a given level, it must be enabled at all higher levels as well. For example, to enable availability at the application level, you must also enable it at the server instance and container levels.

The default for a given level is the setting at the next level up. For example, if availability is enabled at the container level, it is enabled by default at the application level.

When availability is disabled at the server instance level, enabling it at any other level has no effect. When availability is enabled at the server instance level, it is enabled at all levels unless explicitly disabled.

Availability of Single Sign-on in the HTTP Session State

In a single application server instance, once a user is authenticated by an application, the user is not required to reauthenticate individually to other applications running on the same instance. This is called *single sign-on*. For more information on single sign-on, see "Verifying Single Sign-On" on page 227.

For this feature to continue to work even when an HTTP session fails over to another instance in a cluster, single sign-on information must be persisted to the HADB. First enable availability for the server instance and the web container, then enable single-sign-on state persistence. See "Configuring Availability at the Server Instance Level" on page 154.

Applications that can be accessed through a single name and password combination constitute a *single sign-on group*.

For HTTP sessions corresponding to applications that are part of a single sign-on group, if one of the sessions times out, other sessions are not invalidated and continue to be available. This is because time out of one session should not affect the availability of other sessions.

As a corollary of this behavior, if a session times out and you try to access the corresponding application from the same browser window that was running the session, you are not required to authenticate again. However, a new session is created.

Take the example of a shopping cart application that is a part of a single sign-on group with two other applications. Assume that the session time out value for the other two applications is higher than the session time out value for the shopping cart application. If your session for the shopping cart application times out and you try to run the shopping cart application from the same browser window that was running the session, you are not required to authenticate again. However, the previous shopping cart is lost, and you have to create a new shopping cart. The other two applications continue to run as usual even though the session running the shopping cart application has timed out.

Similarly, suppose a session corresponding to any of the other two applications times out. You are not required to authenticate again while connecting to the application from the same browser window in which you were running the session.

NOTE This behavior applies only to cases where the session times out. If single sign-on is enabled and you invalidate one of the sessions using `HttpSession.invalidate()`, the sessions for all applications belonging to the single sign-on group are invalidated. If you try to access any application belonging to the single sign-on group, you are required to authenticate again, and a new session is created for the client accessing the application.

Sample Applications

The following directories contain sample applications that demonstrate HTTP and SFSB session persistence:

install_dir`/samples/ee-samples/highavailability`
install_dir`/samples/ee-samples/failover`

Admin Console Tasks for Configuring Availability

Configuring the SFSB Session Store When Availability Is Disabled

If availability is disabled, the local file system is used for SFSB state passivation, but not persistence. To change where the SFSB state is stored, change the Session Store Location setting in the EJB container. See "Configuring the General EJB Settings" on page 213.

Configuring Availability at the Server Instance Level

To enable or disable availability at the server instance level using the Administration Console:

1. In the tree component, expand the Configurations node.
2. Expand the node for the configuration you want to edit.
3. Select the Availability Service node.
4. Go to the Availability Service page.
5. Enable instance level availability by checking the Availability Service box. To disable it, uncheck the box.

   You can change the Store Pool Name if you changed the JDBC resource used for connections to the HADB for session persistence. For details, see the description of the `configure-ha-cluster` command in the *Reference Manual*.
6. Click on the Save button.
7. Expand the Instances node.
8. Select the server instance.
9. Go to the server instance page.
10. Restart the server.

Configuring Availability at the Web Container Level

To enable availability or override availability settings for an individual web application, use settings in the `sun-web.xml` file. For details, see the *Developer's Guide*.

To enable or disable web container availability using the Administration Console:

1. Select the Web Container Availability tab, then check the Availability Service box. To disable it, uncheck the box. You can also change these optional settings:

- Persistence Type: Specifies the session persistence mechanism for web applications that have availability enabled. Allowed values are `memory` (no persistence) `file` (the file system) and `ha` (the HADB). If availability is enabled, the default is `ha`. If availability is disabled, the default is `memory`. For production environments that require session persistence, use `ha`.

  If the Persistence Type is set to `memory`, you can use the `sessionFilename` property to specify a file system location where the HTTP session state is stored if the server instance is gracefully shut down. This is useful for internal testing but is not supported for production environments.

  If the Persistence Type is set to `file`, you can use the directory property to specify the file system location where the HTTP session state is stored. Persisting to the file system is useful for internal testing but is not supported for production environments.

- Persistence Frequency: Specifies how often the session state is stored. Applicable only if the Persistence Type is `ha`. Allowed values are as follows:

  - `web-method` - The session state is stored at the end of each web request prior to sending a response back to the client. This mode provides the best guarantee that the session state is fully updated in case of failure. This is the default.
  - `time-based` - The session state is stored in the background at the frequency set by the `reapIntervalSeconds` store property. This mode provides less of a guarantee that the session state is fully updated. However, it can provide a significant performance improvement because the state is not stored after each request. To set this property, see "Configuring the Store Properties" on page 212.

- Persistence Scope: Specifies how much of the session state is stored. Applicable only if the Persistence Type is `ha`. Allowed values are as follows:

  - `session` - The entire session state is stored every time. This mode provides the best guarantee that your session data is correctly stored for any distributable web application. This is the default.
  - `modified-session` - The entire session state is stored if it has been modified. A session is considered to have been modified if `HttpSession.setAttribute()` or `HttpSession.removeAttribute()` was called. You must guarantee that `setAttribute()` is called every time an attribute is changed. This is not a J2EE specification requirement, but it is required for this mode to work properly.

- `modified-attribute` - Only modified session attributes are stored. For this mode to work properly, you must follow a few guidelines.

  Call `setAttribute()` every time the session state is modified.

  Make sure there are no cross-references between attributes. The object graph under each distinct attribute key is serialized and stored separately. If there are any object cross references between the objects under each separate key, they are not serialized and deserialized correctly.

  Distribute the session state across multiple attributes, or at least between a read-only attribute and a modifiable attribute.

- Single Sign-On State: Check this box to enable persistence of the single sign-on state. To disable it, uncheck the box.
- HTTP Session Store: You can change the HTTP Session Store if you changed the JDBC resource used for connections to the HADB for session persistence. For details, see the description of the `configure-ha-cluster` command in the *Reference Manual*.

2. Click on the Save button.
3. To change additional optional settings that affect session persistence, see "Configuring Web Container Sessions" on page 211.
4. Expand the Instances node.
5. Select the server instance.
6. Go to the server instance page.
7. Restart the server.

Configuring Availability at the EJB Container Level

To enable availability and select methods to be checkpointed for an individual stateful session bean (SFSB), use settings in the `sun-ejb-jar.xml` file. For details, see the *Developer's Guide*.

To enable or disable EJB container availability using the Administration Console:

1. Select the EJB Container Availability tab, then check the Availability Service box. To disable it, uncheck the box. You can also change these optional settings:

- HA Persistence Type: Specifies the session persistence and passivation mechanism for SFSBs that have availability enabled. Allowed values are `file` (the file system) and `ha` (the HADB). For production environments that require session persistence, use `ha`, the default.
- SFSB Persistence Type: Specifies the passivation mechanism for SFSBs that *do not* have availability enabled. Allowed values are `file` (the default) and `ha`.

  If either Persistence Type is set to `file`, the EJB container specifies the file system location where the passivated session bean state is stored. See "Configuring the General EJB Settings" on page 213. Checkpointing to the file system is useful for internal testing but is not supported for production environments.
- SFSB Store Pool Name: You can change the SFSB Store Pool Name if you changed the JDBC resource used for connections to the HADB for session persistence. For details, see the description of the `configure-ha-cluster` command in the *Reference Manual*.

2. Click on the Save button.
3. Expand the Instances node.
4. Select the server instance.
5. Go to the server instance page.
6. Restart the server.

Chapter 8

Configuring Java Message Service Resources

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

About JMS Resources

The JMS Provider in the Application Server

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

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

JMS Resources

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

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

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

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

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

There are three types of connection factories:

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

There are two kinds of destinations:

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

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

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

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

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

The Relationship Between JMS Resources and Connector Resources

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

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

It is possible to create connector resources for the JMS system resource adapter by using the Connectors node of the Admin Console instead of the JMS Resources node. See Chapter 11, "Connector Resources," for details.

Admin Console Tasks for JMS Connection Factories

Creating a JMS Connection Factory Resource

To create a JMS connection factory resource, follow these steps:

1. In the tree component, expand the Resources node, then expand the JMS Resources 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/ConnectionFactory1`

   It is a recommended practice to use the naming subcontext prefix `jms/` for JMS resources.
5. From the Type drop-down list, choose either `javax.jms.ConnectionFactory`, `javax.jms.QueueConnectionFactory`, or `javax.jms.TopicConnectionFactory`.
6. Select the Enabled checkbox to enable the resource at run time.
7. In the Advanced area, change values as needed for the connection factory attributes. For details about these attributes, see the table entitled “Pool Settings for a Connector Connection Pool” in “Editing a Connector Connection Pool” on page 196. The Application Server applies these attributes to the connector connection pool created for the connection factory.

   For a JMS connection factory resource, specify the Transaction Support value as follows:

- Specify `XATransaction` (the default value) for a resource that can be used for transactions that involve the use of more than one resource within a transaction scope (for example, this resource plus a JDBC resource, a connector resource, or another JMS connection factory resource). This value offers the most flexibility. A resource that is configured as `XATransaction` will participate in two-phase commit operations.
- Specify `LocalTransaction` for a resource that can be used either for transactions that involve only one resource within the transaction scope or as the last agent in a distributed transaction that involves more than one XA resource. This value offers significantly better performance. A resource that is configured as `LocalTransaction` will not be used in two-phase commit operations.
- Specify `NoTransaction` for a resource that can never participate in transactions; this setting is of limited use in JMS applications.

8. In the Additional Properties area, provide values for properties required by applications. The following table lists the available properties.

Table 8-1 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.
`AddressList`	Specifies the names (and, optionally, port numbers) of a message broker instance or instances with which applications will communicate. Each address in the list specifies the host name (and, optionally, host port and connection service) for the connection. For example, the value might be `earth` or `earth:7677`. Specify the port number if the message broker is running on a port other than the default (7676). If the property setting specifies multiple hosts and ports in a clustered environment, the first available host on the list is used unless the `AddressListBehavior` property is set to `RANDOM`. For details, see the *Sun Java System Message Queue Developer's Guide for Java Clients*. Default: The local host and default port number (7676). The client will attempt a connection to a broker on port 7676 of the local host.
`MessageServiceAddressList`	Same as `AddressList`. This property name is deprecated. Use `AddressList` instead.
`UserName`	The user name for the connection factory. Default: `guest`

Table 8-1 Additional Properties for JMS Connection Factories *(Continued)*

Property Name	Description
Password	The password for the connection factory. Default: guest
ReconnectEnabled	If enabled (value = true), specifies that the client runtime attempts to reconnect to a message server (or the list of addresses in the AddressList) when a connection is lost. Default: true
ReconnectAttempts	Specifies the number of attempts to connect (or reconnect) for each address in the AddressList before the client runtime tries the next address in the list. A value of -1 indicates that the number of reconnect attempts is unlimited (the client runtime attempts to connect to the first address until it succeeds). Default: 3
ReconnectInterval	Specifies the interval in milliseconds between reconnect attempts. This applies for attempts on each address in the AddressList and for successive addresses in the list. If the interval is too short, the broker does not have time to recover. If it is too long, the reconnect might represent an unacceptable delay. Default: 30000
AddressListBehavior	Specifies whether connection attempts are in the order of addresses in the AddressList attribute (PRIORITY) or in a random order (RANDOM). RANDOM means that the reconnect chooses a random address from the AddressList. If many clients are likely to attempt a connection using the same connection factory, this value prevents them from all being connected to the same address. PRIORITY means that the reconnect always tries to connect to the first server address in the AddressList and uses another address only if the first broker is not available. Default: RANDOM
AddressListIterations	Specifies the number of times the client runtime iterates through the AddressList in an effort to establish (or re-establish) a connection). A value of -1 indicates that the number of attempts is unlimited. Default: 3

9. In the Targets area, do the following:

a. From the Available column, select the target or targets to which applications that use the resource will be deployed. The available targets include the available clusters and server instances as well as the default server instance, `server`.

b. Click Add to move the target to the Selected column.

10. Click OK to save the connection factory.

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

Editing a JMS Connection Factory Resource

To edit a JMS connection factory resource, follow these steps:

1. In the tree component, expand the Resources node, then expand the JMS Resources 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.
   - Change the values of the attributes in the Advanced area.
   - Add, remove, or modify properties.
5. Optionally, click the Targets tab to go to the JMS Connection Factory Resource Targets page. On this page, do the following:

   a. Click Manage Targets to open the Manage Resource Targets page.

      On this page, move targets between the Available column and the Selected column. Make sure to place in the Selected column the target or targets to which applications that use the resource will be deployed. The available targets include the available clusters and server instances, as well as the default server instance, `server`. Click OK to save the changes.

   b. Select the checkbox for a target, then click Enable or Disable to enable or disable the resource for that target.
6. Click Save to save the changes.

Deleting a JMS Connection Factory Resource

To delete a JMS connection factory resource, follow these steps:

1. In the tree component, expand the Resources node, then expand the JMS Resources 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 Destination Resources

Creating a JMS Destination Resource

To create a JMS destination resource, follow these steps:

1. In the tree component, expand the Resources node, then expand the JMS Resources 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 drop-down list, choose either `javax.jms.Topic` or `javax.jms.Queue`.
6. Select the Enabled checkbox to enable the resource at run time.
7. In the Additional Properties area, provide values for properties. The following table lists the available properties.

Table 8-2 Additional Properties for JMS Destination Resources

Property Name	Description
`Name`	(Required) The name of the physical destination to which the resource refers.
`Description`	A description of the physical destination.

8. In the Targets area, do the following:
   a. From the Available column, select the target or targets to which applications that use the resource will be deployed. The available targets include the available clusters and server instances, as well as the default server instance, `server`.
   b. Click Add to move the target to the Selected column.
9. Click OK.

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

Editing a JMS Destination Resource

To edit a JMS destination resource, follow these steps:

1. In the tree component, expand the Resources node, then expand the JMS Resources node.
2. Expand the Destination Resources node.
3. Select the destination resource to be edited.
4. On the Edit JMS Destination Resource page, you can perform these 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 or Description property.

5. Click Save to save the changes.
6. Optionally, click the Targets tab to go to the JMS Destination Resource Targets page. On this page, do the following:

   a. Click Manage Targets to open the Manage Resource Targets page.

      On this page, move targets between the Available column and the Selected column. Make sure to place in the Selected column the target or targets to which applications that use the resource will be deployed. The available targets include the available clusters and server instances, as well as the default server instance, `server`. Click OK to save the changes.

   b. Select the checkbox for a target, then click Enable or Disable to enable or disable the resource for that target.

Deleting a JMS Destination Resource

To delete a JMS destination resource, follow these steps:

1. In the tree component, expand the Resources node, then expand the JMS Resources 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 JMS Physical Destinations

Creating a JMS Physical Destination

For production purposes, always create physical destinations. During the development and testing phase, however, this step is not required. The first time an application accesses a destination resource, Message Queue automatically creates the physical destination specified by the Name property of the destination resource. The physical destination is temporary and expires after a period specified by a Message Queue configuration property.

To create a JMS physical destination, follow these steps:

1. In the tree component, expand the Configurations node, then expand the Java Message Service node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.
3. Select the Physical Destinations node.
4. On the Physical Destinations page, click New. The Create Physical Destination page appears.
5. In the Physical Destination Name field, type the name of the destination (for example, `PhysicalQueue`).
6. From the Type drop-down list, choose either `topic` or `queue`.
7. In the Additional Properties area, click Add Property to add a property. The following table lists the one property currently available.

Table 8-3 Additional Property for JMS Physical Destinations

Property Name	Description
`maxNumActiveConsumers`	The maximum number of consumers that can be active in load-balanced delivery from a queue destination. A value of -1 means an unlimited number. The default is 1 if the destination is created for a standalone server instance and -1 if it is created for a cluster.

To modify the value of this property or to specify other physical destination properties, use the *install_dir*`/imq/bin/imqcmd` command. See the *Sun Java System Message Queue Administration Guide* for more information.

8. Click OK.

The Physical Destinations page shows the system destination, a queue named `mq.sys.dmq`, to which expired and undeliverable messages are redirected. You can create destination resources, consumers, and browsers for this destination. You cannot delete it or send messages to it.

Equivalent `asadmin` command: `create-jmsdest`

Deleting a JMS Physical Destination

To delete a JMS physical destination, follow these steps:

1. In the tree component, expand the Configurations node, then expand the Java Message Service node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.
3. Select the Physical Destinations node.
4. On the Physical Destinations page, select the checkbox next to the name of the destination to be deleted.
5. Click Delete.

If you try to delete the system destination `mq.sys.dmq`, an error message appears.

Equivalent `asadmin` command: `delete-jmsdest`

Admin Console Tasks for the JMS Provider

Configuring General Properties for the JMS Provider

Use the JMS Service page to configure properties to be used by all JMS connections. Follow these steps:

1. In the tree component, select the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.
3. Select the Java Message Service node to open the JMS Service page.
4. Edit the value in the Startup Timeout field to change the time the Application Server waits for the JMS service to start before aborting the startup. On a slow or overloaded system, increase the value from the default (60).
5. From the Type drop-down list:
   - Choose `LOCAL` (the default for the `server-config` configuration) to access the JMS service on the local host. The JMS service is started and managed by the Application Server.
   - Choose `REMOTE` (the default for the `default-config` configuration) to access the JMS service on another system or on a cluster. If you choose `REMOTE`, the JMS service is not started by the Application Server the next time the server starts. Instead, the JMS service is started and managed via Message Queue, so you must start the Message Queue broker separately. For information about starting the broker, see the *Sun Java System Message Queue Administration Guide*. If you choose this value and are using a remote host, follow the instructions in "Editing a JMS Host" on page 176 to specify the name of the remote host.
6. In the Start Arguments field, type arguments to customize the JMS service startup. Use any arguments available through the *install_dir*`/imq/bin/imqbrokerd` command.
7. Use the Reconnect checkbox to specify whether the JMS service attempts to reconnect to a message server (or the list of addresses in the AddressList) when a connection is lost.

   By default, reconnection is enabled.

8. In the Reconnect Interval field, type the number of seconds between reconnect attempts. This applies for attempts on each address in the AddressList and for successive addresses in the list. If it is too short, this time interval does not give a broker time to recover. If it is too long, the reconnect might represent an unacceptable delay.

   The default value is 60 seconds.

9. In the Reconnect Attempts field, type the number of attempts to connect (or reconnect) for each address in the AddressList before the client runtime tries the next address in the list. A value of -1 indicates that the number of reconnect attempts is unlimited (the client runtime attempts to connect to the first address until it succeeds).

   The default value is 3.

10. Choose a host from the Default JMS Host drop-down list. The default is `default_JMS_host`.

11. In the Address List Behavior drop-down list, choose whether connection attempts are in the order of addresses in the AddressList (`priority`) or in a random order (`random`).

    `priority` means that the reconnect always tries to connect to the first server address in the AddressList and uses another one only if the first broker is not available.

    If there are many clients attempting a connection using the same connection factory, specify `random` to prevent them from all being connected to the same address.

    The default is `random`.

12. In the Address List Iterations field, type the number of times the JMS service iterates through the AddressList in an effort to establish (or re-establish) a connection). A value of -1 indicates that the number of attempts is unlimited.

    The default value is 3.

13. In the MQ Scheme and MQ Service fields, type the Message Queue address scheme name and the MQ connection service name if a nondefault scheme or service is to be used. The full syntax for a message service address is

scheme://*address_syntax*

where the *scheme* and *address_syntax* are described in the table below.

The MQ Scheme and MQ Service are the values shown in the first two columns of the following table.

Table 8-4 Message Server Address Schemes and Syntax

Scheme Name	Connection Service	Description	Address Syntax
mq	jms and ssljms	MQ client runtime makes a connection to the MQ Port Mapper at the specified host and port. The Port Mapper returns a list of the dynamically established connection service ports, and the MQ client runtime then makes a connection to the port hosting the specified connection service.	[*hostName*][:*port*][/*serviceName*] Defaults: *hostName* = localhost *port* = 7676 *serviceName* = jms Defaults only apply to the jms connection service. For the ssljms connection service, all variables need to be specified Example: mq:MyHost:7677/ssljms
mqtcp	jms	MQ client runtime makes a TCP connection to the specified host and port (bypassing the MQ Port Mapper) to establish a connection.	*hostName*:*port*/jms Example: mqtcp:localhost:7676/jms
mqssl	ssljms	MQ client runtime makes a secure SSL connection to the specified host and port (bypassing the MQ Port Mapper) to establish a connection.	*hostName*:*port*/ssljms Example: mqssl:localhost:7676/ssljms
http	httpjms	MQ client runtime makes an HTTP connection to an MQ tunnel servlet at the specified URL. (The broker must be configured to access the HTTP tunnel servlet, as described in the MQ *Administrator's Guide*.)	*hostName*:*port*/*contextRoot*/tunnel If multiple broker instances are using the same tunnel servlet, then the syntax for connecting to a specific broker instance (rather than a randomly selected one) is: http://*hostName*:*port*/*contextRoot*/tunnel?*serverName*=*hostName*:*instanceName*

Table 8-4 Message Server Address Schemes and Syntax *(Continued)*

Scheme Name	Connection Service	Description	Address Syntax
`https`	`httpsjms`	MQ client runtime makes a secure HTTPS connection to the specified MQ tunnel servlet URL. (The broker must be configured to access the HTTPS tunnel servlet, as described in the MQ *Administrator's Guide*.)	*hostName*`:`*port*`/`*contextRoot*`/tunnel` If multiple broker instances are using the same tunnel servlet, then the syntax for connecting to a specific broker instance (rather than a randomly selected one) is: `http://`*hostName*`:`*port*`/`*contextRoot*`/tunnel?`*serverName*`=`*hostName*`:`*instanceName*

14. In the Additional Properties area, click Add Property to add a property. The following table lists the available Message Queue broker configuration properties.

Table 8-5 Additional Properties for JMS Providers

Property Name	Description
`instance-name`	Specifies the full Sun Java System Message Queue broker instance name. The default is `imqbroker`.
`instance-name-suffix`	Specifies a suffix to add to the full Sun Java System Message Queue broker instance name. The suffix is separated from the instance name by an underscore character (_). For example, if the instance name is `imqbroker`, appending the suffix `xyz` changes the instance name to `imqbroker_xyz`.
`append-version`	If `true`, appends the major and minor version numbers, preceded by underscore characters (_), to the full Sun Java System Message Queue broker instance name. For example, if the instance name is `imqbroker`, appending the version numbers changes the instance name to `imqbroker_8_0`. The default is `false`.

15. Click Save to save the changes, or click Load Defaults to restore the default values for the service.

Click Ping to see if the JMS service is up and running. If it is, the message "Ping succeeded: JMS service is running" appears.

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

For more information about configuring the JMS service, see the *Sun Java System Application Server Developer's Guide*.

Equivalent `asadmin` command: `jms-ping`

Creating a JMS Host

To create a JMS host, follow these steps:

1. In the tree component, expand the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.
3. Expand the Java Message Service node.
4. Select the JMS Hosts node.
5. On the JMS Hosts page, click New. The Create JMS Host page appears.
6. In the Name field, type the name of the host. For example:

   ```
   NewJmsHost
   ```

7. In the Host field, type the name or Internet Protocol (IP) address of the system where the JMS host will run (`localhost` or the name of the local or remote system).
8. In the Port field, type the port number of the JMS service. Change this field only if the JMS service to be used is running on a nondefault port. (The default port is 7676.)
9. 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 the MQ broker values have been changed using the *install_dir*`/imq/bin/imqusermgr` command. The default values are `admin` and `admin`.

10. Click OK.

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

Editing a JMS Host

To edit a JMS host, follow these steps:

1. In the tree component, expand the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.
3. Expand the Java Message Service node.
4. Select the JMS Hosts node.
5. On the JMS Hosts page, select the host to be edited.
6. On the Edit JMS Host page, it is possible to perform these tasks:
   - Change the host name or Internet Protocol (IP) address in the Host field.
   - Change the port number of the JMS service in the Port field.
   - Change the values in the Admin Username and Admin Password fields.
7. Click Save to save the changes, or click Load Defaults to restore the default values for the host.

Deleting a JMS Host

To delete a JMS host, follow these steps:

1. In the tree component, expand the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.

3. Expand the Java Message Service node.
4. Select the JMS Hosts node.
5. On the JMS Hosts page, select the checkbox next to the name of the host to be deleted.
6. Click Delete.

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

Chapter 9

Configuring JavaMail Resources

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

About JavaMail

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 Application Server includes the JavaMail API along with JavaMail service providers that allow 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

To create a JavaMail session, follow these steps:

1. In the tree component, expand the Resources node, then select the JavaMail Sessions node.
2. On the JavaMail Sessions page, click New. The Create JavaMail 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 host name of the default mail server. The connect methods of the Store and Transport objects use this value if a protocol-specific host property is not supplied. The name must be resolvable to an actual host name.
5. In the Default User field, type the user name to provide when connecting to a mail server. The connect methods of the Store and Transport objects use this value if a protocol-specific username property is not supplied.
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 the Application Server's mail provider has been reconfigured 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 is generated and is included in the system log file. See "Configuring Log Levels" on page 339 for information about setting the log level.

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

10. In the Targets area, do the following:

    a. From the Available column, select the target or targets to which applications that use the resource will be deployed. The available targets include the available clusters and server instances as well as the default server instance, `server`.

    b. Click Add to move the target to the Selected column.

11. Click OK to save the session.

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

Editing a JavaMail Session

To edit a JavaMail session, follow these steps:

1. In the tree component, expand the Resources node, then select the JavaMail Sessions node.
2. On the JavaMail Sessions page, select the session to be edited.
3. On the Edit JavaMail Session page, you can perform these tasks:
   - Modify the values in the Mail Host, Default User, Default Return Address, and Description fields.
   - Select or deselect the Enabled checkbox to enable or disable the resource.
   - Modify the values in the Advanced fields.

- Add, remove, or modify properties.

4. Click the Targets tab to go to the JavaMail Session Targets page. On this page, do the following:

   a. Click Manage Targets to open the Manage Resource Targets page.

      On this page, move targets between the Available column and the Selected column. Make sure to place in the Selected column the target or targets to which applications that use the resource will be deployed. The available targets include the available clusters and server instances as well as the default server instance, `server`. Click OK to save the changes.

   b. Select the checkbox for a target, then click Enable or Disable to enable or disable the resource for that target.

5. Click Save to save the changes, or click Load Defaults to restore the default values for a mail session.

Deleting a JavaMail Session

To delete a JavaMail session, follow these steps:

1. In the tree component, expand the Resources node, then select the JavaMail Sessions node.
2. On the JavaMail 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 10

JNDI Resources

This chapter describes how to use the Admin Console to configure JNDI resources. It contains the following sections:

- About Java Naming and Directory Interface (JNDI)
- About Custom Resources
- About External JNDI Repositories and Resources

About Java Naming and Directory Interface (JNDI)

This section discusses the Java Naming and Directory Interface (JNDI). JNDI is an application programming interface (API) for accessing different kinds of naming and directory services. J2EE components locate objects by invoking the JNDI lookup method.

This section covers the following topics:

- JNDI Names and Resources
- J2EE Naming Services
- Naming References and Binding Information

JNDI Names and Resources

JNDI is the acronym for the Java Naming and Directory Interface API. By making calls to this API, applications locate resources and other program objects. A resource is a program object that provides connections to 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, called the JNDI name. A resource object and its JNDI name are bound together by the naming and directory service, which is included with the Application Server. To create a new resource, a new name-object binding is entered into the JNDI.

J2EE Naming Services

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

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

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

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

- The application component's business methods access the environment using the JNDI interfaces. The application component provider declares in the deployment descriptor all the environment entries that the application component expects to be provided in its environment at runtime.
- The container provides an implementation of the JNDI naming context that stores the application component environment. The container also provides the tools that allow the deployer to create and manage the environment of each application component.

- A deployer uses the tools provided by the container to initialize the environment entries that are declared in the application component's deployment descriptor. The deployer sets and modifies the values of the environment entries.
- The container makes the environment naming context available to the application component instances at runtime. The application component's instances use the JNDI interfaces to obtain the values of the environment entries.

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

Naming References and Binding Information

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

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

Table 10-1 lists JNDI lookups and their associated references for the J2EE resources used by Sun Java System Application Server.

Table 10-1 JNDI Lookups and Their Associated References

JNDI Lookup Name	Associated Reference
`java:comp/env`	Application environment entries
`java:comp/env/jdbc`	JDBC DataSource resource manager connection factories
`java:comp/env/ejb`	EJB References
`java:comp/UserTransaction`	UserTransaction references
`java:comp/env/mail`	JavaMail Session Connection Factories
`java:comp/env/url`	URL Connection Factories

Table 10-1 JNDI Lookups and Their Associated References

JNDI Lookup Name	Associated Reference
`java:comp/env/jms`	JMS Connection Factories and Destinations
`java:comp/ORB`	ORB instance shared across application components

About Custom Resources

Using Custom Resources

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

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

Creating Custom Resources

To create a custom resource:

1. In the left pane of the Admin Console, open the Sun Java System Application Server instance for the JNDI configuration to be modified.
2. Open the JNDI tab and click Custom Resources. If any custom resources have been created already, they are listed in the right pane. To create a new custom resource, click New. Open the JNDI tab and click New. A page for adding a new custom resource appears.

3. In the JNDI Name field, enter the name to use to access the resource. This name will be registered in the JNDI naming service.
4. In the Resource Type field, enter a fully qualified type definition, as shown in the example above. The Resource Type definition follows the format, `xxx.xxx`.
5. In the Factory Class field, enter a factory class name for the custom resource to be created. The Factory Class is the user-specified name for the factory class. This class implements the `javax.naming.spi.ObjectFactory` interface.
6. In the Description field, enter a description for the resource to be creating. This description is a string value and can include a maximum of 250 characters.
7. In the Additional Properties section, add the property name and value.
8. Mark the Custom Resource Enabled checkbox, to enable the custom resource.
9. Click OK to save your custom resource.

If the custom resource is deployed on a cluster or a stand-alone instance, you can manage targets using the Targets tab. The tab appears after the custom resource has been created. Set the target by entering the target name and clicking OK.

asadmin command equivalent: create-custom-resource.

Editing Custom Resources

To edit a custom resource:

1. In the left pane of the Admin Console, open the Sun Java System Application Server instance for the JNDI configuration to be modified.
2. Open JNDI and select Custom Resources. If any custom resources have been created already, they are listed in the right pane. To edit a custom resource, click on the file name in the right pane.
3. Edit the Resource Type field, the Factory Class field, or the Description field.
4. Mark the Custom Resource Enabled checkbox, to enable the custom resource.
5. Click Save to save the changes to the custom resource.

Deleting Custom Resources

To delete a custom resource:

1. In the left pane of the Admin Console, open the JNDI tab.

2. Click Custom Resources. If any custom resources have been created already, they are listed in the right pane. To delete a custom resource, click in the box next to the name of the resource to be deleted.
3. Click Delete. The custom resource is deleted.

asadmin command equivalent: delete-custom-resource.

Listing Custom Resources

To list the custom resources, type the `asadmin ist-custom-resources` command. For example, to list custom resources on the the host, plum, type the following:

```
$asadmin list-custom-resource --host plum target6
```

For the full context, type `asadmin help list-custom-resources`.

About External JNDI Repositories and Resources

Using External JNDI Repositories and Resources

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

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

```
<resources>
<!-- external-jndi-resource element specifies how to access J2EE resources
-- stored in an external JNDI repository. The following example
-- illustrates how to access a java object stored in LDAP.
-- factory-class element specifies the JNDI InitialContext factory that
-- needs to be used to access the resource factory. property element
-- corresponds to the environment applicable to the external JNDI context
-- and jndi-lookup-name refers to the JNDI name to lookup to fetch the
-- designated (in this case the java) object.
-->
<external-jndi-resource jndi-name="test/myBean"
jndi-lookup-name="cn=myBean"
res-type="test.myBean"
factory-class="com.sun.jndi.ldap.LdapCtxFactory">

<property name="PROVIDER-URL" value="ldap://ldapserver:389/o=myObjects" />
<property name="SECURITY_AUTHENTICATION" value="simple" />
<property name="SECURITY_PRINCIPAL", value="cn=joeSmith, o=Engineering" />
<property name="SECURITY_CREDENTIALS" value="changeit" />
</external-jndi-resource>
</resources>
```

Creating External Resources

To create an external resource:

1. In the left pane of the Admin Console, open the Sun Java System Application Server instance for the JNDI configuration to be modified.
2. Open JNDI and select External Resources. If any external resources have been created already, they are listed in the right pane. To create a new external resource, click New.
3. In the JNDI Name field, enter the name that is to be used to access the resource. This name is registered in the JNDI naming service.
4. In the Resource Type field, enter a fully qualified type definition, as shown in the example above. The Resource Type definition follows the format, `xxx.xxx`.
5. In the JNDI Lookup field, enter the JNDI value to look up in the external repository. For example, when creating an external resource to connect to an external repository, to test a bean class, theJNDI Lookup can look like this; `cn=`*testmybean*.

6. In the Factory Class field, enter a JNDI factory class external repository, for example, `com.sun.jndi.ldap`. This class implements the javax.naming.spi.`ObjectFactory` interface.
7. In the Description field, enter a description for the resource to be created. This description is a string value and can include a maximum of 250 characters.
8. In the Additional Properties section, add the property name and value.
9. Mark the External Resource Enabled checkbox, to enable the external resource.
10. Click OK to save the external resource.

If the external resource is deployed on a cluster or a stand-alone instance, you can manage targets using the Targets tab. The tab appears after the external resource has been created. Set the target by entering the target name and clicking OK.

asadmin command equivalent: `create-jndi-resource` .

Editing External Resources

To edit an external resource:

1. In the left pane of the Admin Console, open the Sun Java System Application Server instance for the JNDI configuration to be modified.
2. Open JNDI and select External Resources. If any external resources have been created already, they are listed in the right pane. To edit an external resource, click on the file name in the right pane.
3. Edit the Resource Type field, the JNDI Lookup field, the Factory Class field, or the Description field.
4. Mark the External Resource Enabled checkbox, to enable the external resource.
5. Click Save to save the changes to the external resource.

Deleting External Resources

To delete an external resource:

1. In the left pane of the Admin Console, open the JNDI tab.
2. Click External Resources. If any external resources have been created already, they are listed in the right pane. To delete an external resource, click the box next to the name of the resource to be deleted.

3. Click Delete. The external resource is deleted.

asadmin command equivalent delete-jndi-resource.

Listing External Resources

To list external resources, type the `asadmin list-jndi-resources` command and specify the jndi name. For example, to list an external resource, type the following:

```
$asadmin list-jndi-resources -- target plum jndi_name_test
```

For the full context, type `asadmin help list-jndi-resources`.

Chapter 11

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

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. To create a connector connection pool, specify the connector module (resource adapter) that is associated with the pool.

A connector resource is a program object that provides an application with a connection to an EIS. To create a connector resource, specify its JNDI name and its associated connection pool. Multiple connector resources can specify a single connection pool. The application locates the resource by looking up its JNDI name. (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 Relationship Between JMS Resources and Connector Resources.

Connector Connection Pool Tasks

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 creating the pool, deploy the connector module (resource adapter) associated with the pool. The values that are specified for the new pool depend on the connector module that is deployed.

To create a connector connection pool:

1. In the tree component, expand the Resource node and then 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.

      Specify this name when creating 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, a type of `ConnectionFactory` is specified, a factory instance to get a connection to the EIS.
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, the default values can be retained.

      These settings can be changed 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`

- Connector Modules, Connection Pools, and Resources
- Deploying a Connector Module

Editing a Connector Connection Pool

The Edit Connector Connection Pool page provides the means to change the pool settings and the additional properties.

To access the Edit Connector Connection Pool page:

1. In the tree component, expand the Resources node and then 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 11-1.

Table 11-1 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.

Table 11-1 Pool Settings for a Connector Connection Pool

Parameter	Description
Transaction Support	Use the Transaction Support list to select the type of transaction support for the connection pool. The chossen transaction support overrides the transaction support attribute in the resource adapter associated with this connection pool in a downward compatible way. In other words, it can support a lower transaction level than that specified in the resource adapter or the same transaction level as that specified in resource adapter, but it cannot specify a higher level. The transaction support options include the following. The None selection from the Transaction Support menu indicates that the resource adapter does not support resource manager local or JTA transactions and does not implement XAResource or LocalTransaction interfaces. Local transaction support means that the resource adapter supports local transactions by implementing the LocalTransaction interface. Local transactions are managed internal to a resource manager and involve no external transaction managers. XA transaction support means that the resource adapter supports resource manager local and JTA transactions by implementing the LocalTransaction and XAResource interfaces. XA transactions are controlled and coordinated by a transaction manager external to a resource manager. Local transactions are managed internal to a resource manager and involve no external transaction managers.

5. In the Additional Properties table, specify name-value pairs. The properties specified depend on the resource adapter used by this pool. The name-value pairs specified by the deployer using this table can be used to override the default values for the properties defined by the resource-adapter vendor.
6. On the Security Maps tabbed pane, create or modify a security map for the connection pool. See "About Security Maps" for information on how to create a security map.
7. Click Save.

Deleting a Connector Connection Pool

1. In the tree component, expand the Resources node and then the Connectors node.
2. Select the Connector Connection Pools node.

3. On the Connector Connector Connection Pools page, select the checkbox for the pool to be deleted.
4. Click Delete.

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

Connector Resources Tasks

Creating a Connector Resource

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

To create a connector resource:

1. In the tree component, expand the Resources node and then 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 to which the the new connector resource belongs.
    c. By default, the resource is available (enabled) as soon as it is created. To change the resource to be unavailable, select the Disable on All Targets radio button.

d. In the Targets section of the page, select the domain, cluster, or server instances where the connector resource will reside, from the Available field and click Add. If you do not want to deploy the connector resource to one of the domains, clusters, or server instances listed in the Selected field, select it from the field and click Remove.

5. Click OK.

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

Editing a Connector Resource

1. In the tree component, expand the Resources node and then 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 select a different connection pool from the Pool Name menu.
5. On the Targets tabbed pane, you can edit the targets on which the connector resource is deployed by clicking Manage Targets. See Creating a Connector Resource for more information on targets.
6. Click Save to apply the edits.

Deleting a Connector Resource

1. In the tree component, expand the Resources node and then the Connectors node.
2. Select the Connector Resources node.
3. On the Connector Resources page, select the checkbox for the resource to be deleted.
4. Click Delete.

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

•

Configuring a Connector Service

Use the Connector Service screen to configure the connector container for all resource adapters deployed to this cluster or server instance.

To configure the connector container:

1. Select Configurations from the tree.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, server, select the server-config node.
   b. To configure the default settings for future instances that use a copy of default-config, select the default-config node.
3. Select the Connector Service node.
4. Specify the shutdown timeout in seconds in the Shutdown Timeout field. Enter an integer representing the number of seconds that the application server waits to allow the `ResourceAdapter.stop` method of the connector module's instance to complete. Resource adapters that take longer than the specified shutdown timeout are ignored by the application server and the shutdown procedure continues. The default shutdown timeout is 30 seconds. Click Load Defaults to select the default shutdown timeout for the resource adapters deployed to this cluster or server instance.

Administered Object Resources Tasks

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, add name-value property pairs in the Create or Edit Admin Ojbect Resource pages. When creating an administered object resource, associate the administered object to a JNDI name.

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

To create an administered object resource:

1. In the tree component, expand the Resources node and then the Connectors node.
2. Expand the Admin Object Resources node.
3. On the Admin Object Resources page, click New.
4. On the Admin Object Resources page, specify the following settings:
    a. In the JNDI Name field, type a uniqe name that identifies 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.
    e. Click Next.
5. On the second Create Admin Object Resource page, the following tasks can be performed.
    a. To configure the administered object with name-value property pairs, click Add Property.
    b. In the Targets section of the page, select the domain, cluster, or server instances where the administered object will reside, from the Available field and click Add. To undeploy the administered object to one of the domains, clusters, or server instances listed in the Selected field, select it from the field and click Remove.
6. Click Finish.

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

Editing an Administered Object Resource

1. In the tree component, expand the Resource node and then the Connectors node.
2. Expand the Administered Object Resources node.
3. Select the node for the administered object resource to be edited.
4. On the Edit Administered Object Resources page, modify values specified in Creating an Administered Object Resource.
5. On the Targets tabbed pane, edit the targets on which the administered object is deployed by clicking Manage Targets. See Creating an Administered Object Resource for more information on targets.
6. Click Save to apply the edits.

Deleting an Administered Object Resource

1. In the tree component, expand the Resources node and then the Connectors node.
2. Select the Administered Object Resources node.
3. On the Administered Object Resources page, select the checkbox for the resource to be deleted.
4. Click Delete.

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

Chapter 12

Managing Named Configurations

This chapter describes adding, changing, and using named server configurations in Application Server. It contains the following sections:

About Named Configurations

Named Configurations

A named configuration is a set of server configuration information. This information includes configuration settings for things such as http listeners, orb/iiop listeners, JMS brokers, the EJB container, security, logging, and monitoring. Applications and resources are not defined in a named configurations.

Configurations are created in the administration domain. Multiple server instances or clusters in the domain can reference the same configuration, or they can have separate configurations.

For clusters, all server instances in the cluster inherit the cluster's configuration so that a homogenous environment is assured in a cluster's instances.

Because a named configuration contains so many required configuration settings, create a new configuration by copying an existing named configuration. The newly-created configuration is identical to the configuration you copied until you change its configuration settings.

There are three types of clusters and instances, depending on the way the clusters or instances use configurations:

- **Stand-alone**. A stand-alone server instance or cluster doesn't share its configuration with another server instance; that is, no other server instance or cluster references the named configuration.
- **Shared.** A shared server instance or cluster shares a configuration with another server instance or cluster; that is, multiple instances or clusters reference the same named configuration.
- **Clustered.** A clustered server instance inherits the cluster's configuration.

The default-config Configuration

The `default-config` configuration is a special configuration that acts as a template for creating stand-alone server instance or stand-alone cluster configurations. No unclustered server instances or clusters are allowed to refer to the `default-config` configuration; it can only be copied to create new configurations. Edit the default configuration to ensure that new configurations copied from it have the correct initial settings.

Configurations Created when Creating Instances or Clusters

When creating a new server instance or a new cluster, either:

- Reference an existing configuration. No new configuration is added.
- Make a copy of an existing configuration. A new configuration is added when the server instance or cluster is added.

By default, new clusters or instances are created with configurations copied from the `default-config` configuration. To copy from a different configuration, specify it when creating a new instance or cluster.

For a server instance, the new configuration is named *instance_name*`-config`. For a cluster, the new configuration is named *cluster-name*`-config`.

Unique Port Numbers and Configurations

If multiple instances on the same host machine reference the same configuration, each instance must listen on a unique port number. For example, if two server instances reference a named configuration with an HTTP listener on port 80, a port conflict prevents one of the server instances from starting. Change the properties that define the port numbers on which individual server instances listen so that unique ports are used.

The following principles apply to port numbers:

- Port numbers for individual server instances are initially inherited from the configuration.
- If the port is already in use when you create a server instance, override the inherited default value at the instance level to prevent port conflicts.
- Assume an instance is sharing a configuration. The configuration has port number n. If you create a new instance on the machine using the same configuration, the new instance is assigned port number n+1, if it is available. If it is not available, the next available port after n+1 is chosen.
- If you change the port number of the configuration, a server instance inheriting that port number automatically inherits the changed port number.
- If you change an instance's port number and you subsequently change the configuration's port number, the instance's port number remains unchanged.

Admin Console Tasks for Named Configurations

Creating a Named Configuration

To create a named configuration:

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

2. On the Configurations page, click New.
3. On the Create Configurations page, enter a unique name for the configuration.
4. Select a configuration to copy.

   The configuration `default-config` is the default configuration used when creating stand-alone server instance or stand-alone cluster.

Equivalent `asadmin` command: `copy-config`.

Editing a Named Configuration's Properties

To edit a named configuration's properties:

1. In the tree component, expand the Configurations node.
2. Select the node for a named configuration.
3. On the Configuration System Properties page, choose whether to enable dynamic reconfiguration.

   If enabled, changes to the configuration are applied to the server instances without requiring a server restart.
4. Add properties, change current values for properties, or delete properties.

   The properties that are already defined are ports. If more than one server instance exists on a system, the port numbers must be unique.

Table 12-1 contains a list of predefined properties and their descriptions.

Table 12-1 Properties for Named Configurations

Property Name	Description
HTTP_LISTENER_PORT	This property specifies the port number for http-listener-1. Valid values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges.
HTTP_SSL_LISTENER_PORT	This property specifies the port number for http-listener-2. Valid values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges.
IIOP_SSL_LISTENER_PORT	This property specifies which ORB listener port for IIOP connections the IIOP listener called SSL listens on.

Table 12-1 Properties for Named Configurations

Property Name	Description
IIOP_LISTENER_PORT	This property specifies which ORB listener port for IIOP connections orb-listener-1 listens on.
JMX_SYSTEM_CONNECTOR_PORT	This property specifies the port number on which the JMX connector listens. Valid values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges.
IIOP_SSL_MUTUALAUTH_PORT	This property specifies which ORB listener port for IIOP connections the IIOP listener called SSL_MUTUALAUTH listens on.

5. To edit the current values of a property for all instances associated with the configuration, click Instance Values.

Equivalent `asadmin` command: `set`.

Editing Port Numbers for Instances Referencing a Configuration

Each instance referencing a named configuration initially inherits its port numbers from that configuration. Since port numbers must be unique on the system, you might need to override the inherited port numbers.

1. In the tree component, expand the Configurations node.
2. Select the node for a named configuration.
3. On the Configuring System Properties page, click Instance Values next to the port number you want to edit.

   For example, if you click Instance Values next to the SSL-port property, you see the value of SSL-port for every server instance that references that configuration.
4. Change the port values and click Save.

Equivalent `asadmin` command: `set`.

Viewing a Named Configuration's Targets

To view a named configuration's targets:

1. In the tree component, expand the Configurations node.
2. Select a node for the named configuration.

   The Configuration System Properties page displays a list of all targets using the configuration. For a cluster configuration, those targets are clusters. For an instance configuration, those targets are instances.

Deleting a Named Configuration

To delete a named configuration:

1. In the tree component, select the Configurations node.
2. On the Configurations page, select the checkbox for the named configuration to delete.

   The `default-config` configuration cannot be deleted.
3. Click Delete.

Equivalent `asadmin` command: `delete-config`.

Chapter 13

J2EE Containers

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

About the J2EE Containers

This section describes the J2EE containers included with the Application Server.

Types of J2EE Containers

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

The Web Container

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

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

Configuring Web Container Sessions

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

To set the session timeout value:

1. In the tree component, select the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for all instances, select the `default-config` node.
3. Select the Web Container node.
4. Click the Session Properties tab.
5. In the Session Timeout field enter the number of seconds that a session is valid.
6. Click Save.

Configuring the Manager Properties

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

To change the session manager settings:

1. In the tree component select the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for all instances, select the `default-config` node.
3. Select the Web Container node.
4. Click the Manager Properties tab.
5. Set the Reap Interval value.

   The Reap Interval field is the number of seconds before the inactive session data is deleted from the store.

6. Set the Max Sessions value.

   The Max Sessions field is the maximum number of sessions allowed.

7. Set the Session Filename value.

   The Session Filename field is the file that contains the session data.

8. Set the Session ID Generator Classname value.

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

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

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

public interface UuidGenerator {

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

}
```

   The class must be in the Application Server classpath.

9. Click Save.

Configuring the Store Properties

1. In the tree component select the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for all instances, select the `default-config` node.
3. Select the Web Container node.
4. Click the Store Properties tab.

5. Set the Reap Interval.

   The Reap Interval field is the number of seconds before the inactive session data is deleted from the store.

6. Click Save.

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

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

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

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

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 select the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for all instances, select the `default-config` node.
3. Select the EJB Container node.
4. Under Pool Settings in the Initial and Minimum Pool Size field enter the minimum number of beans the container creates in the pool.
5. In the Maximum Pool Size field enter the maximum number of beans the container maintains 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. Click Save.
9. 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 select the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for all instances, select the `default-config` node.
3. Select the EJB Container node.
4. 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 the cache is increased, the server consumes more memory and resources. Be sure your operating environment is sufficient for your cache settings.
5. Adjust the cache resize quantity in the Cache Resize Quantity field.

   When the maximum number of cached beans is reached, the container removes a number of passivated beans from the backup store, set to 32 by default.
6. 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 is passivated. That is, the bean's state is written to a backup store.
7. Adjust the time, in seconds, after which stateful session beans are removed from the cache or passivated store in the Removal Timeout field.
8. 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.

9. Click Save.
10. 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 select the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for all instances, select the `default-config` node.
3. Select the EJB Container node.
4. Click the MDB Settings tab.
5. Under Pool Settings in the Initial and Minimum Pool Size field, enter the minimum number of message beans the container creates in the pool.
6. In the Maximum Pool Size field, enter the maximum number of beans the container maintains in the pool, at any time.
7. In the Pool Resize Quantity field enter the number of beans that are removed from the pool if they are idle for more than the time specified in Pool Idle Timeout.
8. In the Pool Idle Timeout field, enter the time, in seconds, that a bean in the pool can remain idle before it is removed from the pool.
9. Click Save.

10. 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 select the Configurations node.
2. Select the instance to configure:
    a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
    b. To configure the default settings for all instances, select the `default-config` node.
3. Select the EJB Container node.
4. Click the EJB Timer Service tab.
5. 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 server overload.
6. Set the maximum number of attempts the timer service makes to deliver the notification in the Maximum Redeliveries field.
7. Set the interval, in milliseconds, between redelivery attempts in the Redelivery Interval field.
8. Click Save.
9. 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" on page 144.
2. Enter the JNDI name of the resource in the Timer Datasource field.
3. Click Save.
4. Restart the Application Server.

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

Chapter 14

Configuring Security

This chapter describes some core application server security concepts, and describes how to configure security for the Sun Java System Application Server 8.1 2005Q1. This chapter contains the following topics:

About Application Server Security

- Managing Security 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. The following topics are discussed:

- Understanding Application and System Security
- Tools for Managing Security
- Managing Security of Passwords
- Assigning Security Responsibilities

Understanding Application and System Security

Broadly, there are two kinds of application security:

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

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

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

Tools for Managing Security

The Application Server provides the following tools for managing security:

- Admin Console, a browser-based tool used to configure security for the entire server, to manage users, groups, and realms, and to perform other system-wide security tasks. For a general introduction to Admin Console, see "Tools for Administration". For an overview of the security tasks you can perform with Admin Console, see "Managing Security 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 from either a command prompt or from a script, 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 does not describe its use in detail. For instructions on using `deploytool`, see the tool's online help and *The J2EE 1.4 Tutorial* at `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`.

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

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

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

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

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

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

Managing Security of Passwords

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

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

1. From the directory where the domain.xml file resides (which is *install_dir*/domains/*domain_dir*/config by default), run the following asadmin command:

   ```
   asadmin create-password-alias <alias-name>
   ```

   For example,

   ```
   asadmin create-password-alias jms-password
   ```

   A password prompt appears (admin in this case). Refer to the manpages for the create-password-alias, list-password-aliases, delete-password-alias commands for more information.

2. Remove and replace the password in domain.xml. This is accomplished using the asadmin set command. An example of using the set command for this purpose is as follows:

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

3. Restart the Application Server for the relevant domain.

Protecting files with encoded passwords

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

- *install_dir*/domains/*domain_dir*/master-password

This file contains the encoded master password and should be protected with file system permissions 600.

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

Changing the Master Password

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

To change the master password, the following procedure must be followed:

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

   ```
   asadmin change-master-password>
   Please enter the master password>
   Please enter the new master password>
   Please enter the the new master password again>
   ```

2. Restart the Application Server.

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

3. Stop each node agent and its related servers one at a time. Run the `asadmin change-master-password` command again, and then restart the node agent and its related servers.

4. Continue with the next node agent until all node agents have been addressed. In this way, a rolling change may be accomplished.

Changing the Admin Password

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

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

It is also possible to change the admin password using Admin Console. To change the admin password using the Admin Console, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance you want to configure:
    a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
    b. To configure the default settings for all instances, expand the `default-config` node.
3. Expand the Security node.
4. Expand the Realms node.
5. Select the `admin-realm` node.
6. Click the Manage Users button from the Edit Realm page.
7. Select the user named `admin`.
8. Enter the new password and confirm the password.
9. Click Save to save or click Close to close without saving.

Assigning 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 can use tools such as `deploytool` to edit application deployment descriptors. These security tasks are discussed in more detail in the *Security* chapter of *The J2EE 1.4 Tutorial*, which can be viewed at the following URL:

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

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 can use tools such as `deploytool` to edit application deployment descriptors. These security tasks are discussed in more detail in the *Security* chapter of *The J2EE 1.4 Tutorial*, which can be viewed at the following URL:

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

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 uses the Admin Console to manage server security settings and `certutil` to manage certificates. This document is intended primarily for system administrators.

About Authentication and Authorization

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

- Authenticating Entities
- Authorizing Users
- Specifying JACC Providers
- Auditing Authentication and Authorization Decisions

- Configuring Message Security

Authenticating Entities

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

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

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

The Application Server supports four types of authentication, as outlined in Table 14-1. An application specifies the type of authentication it uses within its deployment descriptors. For more information on using `deploytool` to configure the authentication method for an application, see *The J2EE 1.4 Tutorial* at the following URL:

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

Table 14-1 Application Server Authentication Methods

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

Verifying Single Sign-On

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

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

Single sign-on is enabled by default for virtual servers defined for the Application Server. For information on disabling single sign-on, see "Configuring Single Sign-On (SSO)".

Authorizing Users

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

Specifying JACC Providers

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

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

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

Auditing Authentication and Authorization Decisions

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

Configuring Message Security

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

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

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

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

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

Understanding Users, Groups, Roles, and Realms

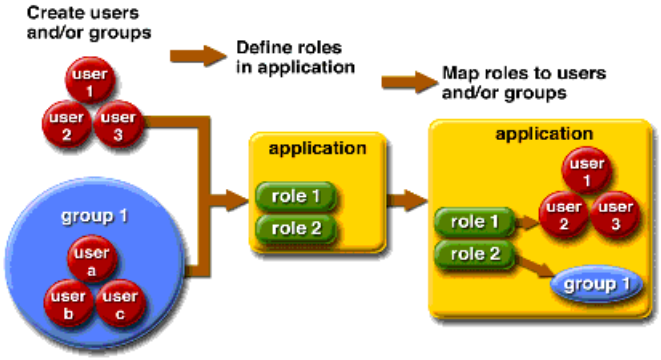
The Application Server enforces its authentication and authorization policies upon the following entities:

- Users: An individual identity *defined in the Application Server*. In general, a user is a person, a software component such as an enterprise bean, or even a service. A user who has been authenticated is sometimes called a *principal*. Users are sometimes referred to as *subjects*.
- Groups: A set of users *defined in the Application Server*, classified by common traits.
- Roles: A named authorization level *defined by an application*. A role can be compared to a key that opens a lock. Many people might have a copy of the key. The lock doesn't care who seeks access, only that the right key is used.

- Realms: A repository containing user and group information and their associated security credentials. A realm is also called a *security policy domain*.

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

Role Mapping



Users

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

Groups

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

Roles

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

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

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

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

Realms

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

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

In the `file` realm, the server stores user credentials locally in a file named `keyfile`. You can use the 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".

The `admin-realm` is also a `FileRealm` and stores administrator user credentials locally in a file named `admin-keyfile`. Use the Admin Console to manage users in this realm in the same way you manage users in the `file` realm. For more information, see "Managing file Realm Users".

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

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

A custom realm is any other repository of user credentials, such as a relational database or third-party component. For more information, see "Creating a Custom Realm" or he *Developer's Guide* chapter titled *Securing Applications*.

Introduction to Certificates and SSL

The following topics are discussed in this section:

About Digital Certificates

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

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

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

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

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

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

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

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

About 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 imports the response, the self-signed certificate is replaced by a chain of certificates. At the bottom of the chain is the certificate (reply) issued by the CA authenticating the subject's public key. The next certificate in the chain is one that authenticates the CA's public key. Usually, this is a self-signed certificate (that is, a certificate from the CA authenticating its own public key) and the last certificate in the chain.

In other cases, the CA can return a chain of certificates. In this case, the bottom certificate in the chain is the same (a certificate signed by the CA, authenticating the public key of the key entry), but the second certificate in the chain is a certificate signed by a different CA, authenticating the public key of the CA to which you sent

the CSR. Then, the next certificate in the chain is a certificate authenticating the second CA's key, and so on, until a self-signed *root* certificate is reached. Each certificate in the chain (after the first) thus authenticates the public key of the signer of the previous certificate in the chain.

About Secure Sockets Layer

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

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

- The browser sends a message over the network requesting a secure session (typically, by requesting a URL that begins with `https` instead of `http`).
- The server responds by sending its certificate (including its public key).
- The browser verifies that the server's certificate is valid and is signed by a CA whose certificate is in the browser's database (and who is trusted). It also verifies that the CA certificate has not expired.
- If the certificate is valid, the browser generates a one-time, unique *session key* and encrypts it with the server's public key. The browser then sends the encrypted session key to the server so that they both have a copy.
- The server decrypts the message using its private key and recovers the session key.

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

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

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

About Ciphers

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

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

Using Name-based Virtual Hosts

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

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

About Firewalls

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

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

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

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

Managing Security With the Admin Console

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

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

Server Security Settings

On the Security Settings page, set properties for the entire server, including specifying the default realm, the anonymous role, and the default principal user name and password. For more information, see “Configuring Security Settings”.

Realms and file Realm Users

The concept of realms was introduced in “Understanding Users, Groups, Roles, and Realms”. Use the Admin Console to perform the following tasks:

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

- Set the default realm

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

JACC Providers

JACC providers were introduced in "Specifying JACC Providers". Use the Admin Console to perform the following tasks:

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

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

Use the Admin Console to perform the following tasks:

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

See "Admin Console Tasks for Audit Modules" for details on these tasks.

Message Security

The concept of message security was introduced in "Configuring Message Security". Use the Admin Console to perform the following tasks:

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

See "Configuring Message Security" 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, perform the following tasks:

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

See "Admin Console Tasks for Listeners and JMX Connectors" for details on these tasks.

Admin Service Security

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

With the Admin Console, perform the following tasks:

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

See "Configuring Security for the Admin Service's JMX Connector" for details on these tasks.

Security Maps

The concept of security maps for connector connection pools is introduced in "About Security Maps". Use the Admin Console to perform the following tasks:

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

See "Admin Console Tasks for Connector Connection Pools" for details on these tasks.

Admin Console Tasks for Security

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, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Select the Security node.

   The Security page displays.
4. Modify the values as necessary. The general security options are discussed in Table 14-2.

Table 14-2 General Security Settings

Setting	Description
Audit Logging	Select to enable audit logging. If enabled, the server will load and run all the audit modules specified in the Audit Modules setting. If disabled, the server does not access audit modules. Disabled by default.
Default Realm	The active (default) realm the server uses for authentication. Applications 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.

Table 14-2 General Security Settings *(Continued)*

Setting	Description
Default Principal	Specifies the default user name. The server uses this when no principal is provided. If you enter a value in this field, enter a corresponding value in the Default Principal Password field. This attribute is not required for normal server operation.
Default Principal Password	Password of the default principal specified in the Default Principal field. This attribute is not required for normal server operation.
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 already be 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".

5. Enter additional properties to pass to the Java Virtual Machine (JVM) in the Additional Properties section.

   Valid properties are dependent upon the type of realm selected ini the Default Realm field. Valid properties are discussed in the following sections:

   - Editing the file and admin-realm Realms
   - Editing the certificate Realm
   - Creating the solaris Realm
   - Creating an ldap Realm
   - Creating a Custom Realm

6. Select Save to save the changes or Load Defaults to restore the default values

Controlling Access to Administration Tools

Only users in the `asadmin` group are able to access Admin Console and the `asadmin` command line utility.

To give a user access to these administration tools, add them to the `asadmin` group in the `admin-realm`. To do this, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.

2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Expand the Security node.
4. Expand the Realms node.
5. Select the `admin-realm` node.
6. Click the Manage Users button from the Edit Realm page.

   Initially after installation, the administrator user name and password entered during installation are listed in a file named `admin-keyfile`. 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.

   If you add users to the `admin-realm` realm, but assign the user to a group other than `asadmin`, the user information will still be written to the file named `admin-keyfile`, but the user will have no access to administrative tools or to applications in the `file` realm.
7. Click New to add a new user to the `admin-realm` realm.
8. 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.
9. Click OK to add this user to the `admin-realm` realm or click Cancel to quit without saving.

Admin Console Tasks for Realms

Creating a Realm

The Application Server comes preconfigured with three realms: `file`, `certificate`, and `admin-realm`. It is also possible to create `ldap`, `solaris`, and custom realms. Generally, you will have one realm of each type on a server, but on the Application Server there are two file realms: `file` and `admin-realm`. These are two realms of the same type used for two different purposes. It is also possible to have a different certificate database for each virtual server on your system.

To create a security realm, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance you want to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Expand the Security node.
4. Select the Realms node.
5. On the Realms page, click New.

   The Create Realm page is displayed.
6. Enter a name for the realm in the Name field.

7. Specify the class name for the realm being created. Valid choices are shown in Table 14-3:

Table 14-3 Valid values for realm class name

Realm Name	Class Name
`file`	`com.sun.enterprise.security.auth.realm.file.FileRealm`
`certificate`	`com.sun.enterprise.security.auth.realm.certificate.CertificateRealm`
`ldap`	`com.sun.enterprise.security.auth.realm.ldap.LDAPRealm`
`solaris`	`com.sun.enterprise.security.auth.realm.solaris.SolarisRealm`
custom	Name of login realm class

8. Add the required properties and any desired optional properties for the realm.

   To add a property:

   a. Click Add Property.

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

   - For a description of `file` realm properties, see "Editing the file and admin-realm Realms".
   - For a description of `certificate` realm properties, see "Editing the certificate Realm".
   - For a description of `ldap` realm properties, see "Creating an 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".

   c. In the Value field, enter the value of the property.

9. Click OK.

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

Creating an ldap Realm

The `ldap` realm performs authentication using information from an LDAP server. User information includes user name, password, and the groups to which the user belongs. To use an LDAP realm, the users and groups must already be defined in your LDAP directory.

To create an LDAP realm, follow the steps in "Creating a Realm" for adding a new realm, then add the properties as shown in Table 14-4.

Table 14-4 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 Distinguished Name (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`.
jaas-context	Type of login module to use for this realm.	Must be `ldapRealm`.

Optional properties for the `ldap` realm are shown in Table 14-5:

Table 14-5 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 group 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 the 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
```

Using the example code, when creating or editing the `ldap` realm, you can enter the values as shown in Table 14-6.

Table 14-6 Example `ldap` realm values

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`. Can 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 the 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 Application Server *Developer's Guide* chapter titled Securing Applications.

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

To configure the Application Server to use a custom realm:

1. Follow the procedure outline in "Creating a Realm", entering the name of the custom realm and the name of the `LoginModule` class. Any unique name can be used for the custom realm, for example `myCustomRealm`.
2. Add the properties shown in Table 14-7:

Table 14-7 Valid properties for a custom realm

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,

```
myCustomRealm {
    com.foo.bar.security.customrealm.simpleCustomLoginModule required;
};
```

5. Copy the `LoginModule` class and all dependent classes into the directory *install_dir*`/domains/`*domain_name*`/lib/classes`.
6. Restart the Server if Restart Required displays in the console.

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

To edit a realm, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, server, expand the server-config node.
   b. To configure the default settings for all instances, expand the default-config node.
3. Expand the Security node.
4. Expand the Realms node.
5. Select the name of an existing realm.

   The Edit Realm page displays.
6. Edit existing properties and their values as desired.

   For information on file realm properties, see "Editing the file and admin-realm Realms". To manage users in the file realm, click the Manage Users button; see "Managing file Realm Users" for more information.

   For information on certificate realm properties, see "Editing the certificate Realm".
7. To add additional properties, click the Add Properties button. The page displays a new row. Enter a valid property name and property value. See the following tables for a description of the optional properties that can be configured:
   - Table 14-4, Required properties for ldap realm
   - Table 14-5, Optional properties for ldap realm
   - Table 14-7, Valid properties for a custom realm
   - Table 14-8, Required properties for file realms

- Table 14-9, Optional properties for `certificate` realm

8. Click Save to save the changes.

Editing the file and admin-realm Realms

The server maintains all user, group, and password information in a file named `keyfile` for the `file` realm and `admin-keyfile` for the `admin-realm`. For both, the `file` property specifies the location of the keyfile. Table 14-8 shows required properties for a `file` realm.

Table 14-8 Required properties for file realms

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

The `keyfile` is initially empty, so users must be added before the `file` realm is used. For instructions, see "Managing file Realm Users".

The `admin-keyfile` initially contains the admin user name, the admin password in an encrypted format, and the group to which this user belongs, which is `asadmin` by default. For more information on adding users to the `admin-realm`, read "Controlling Access to Administration Tools".

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

Managing Users with Network Security Services (NSS)

In the **Enterprise Edition only**, you can manage users using the Admin Console as discussed in "Managing file Realm Users" or you can manage users using NSS tools. Network Security Services (NSS) is a set of libraries designed to support cross-platform development of security-enabled client and server applications. Applications built with NSS can support SSL v2 and v3, TLS, PKCS #5, PKCS #7, PKCS #11, PKCS #12, S/MIME, X.509 v3 certificates, and other security standards. For detailed information, link to the following URLs:

- Network Security Services (NSS) at `http://www.mozilla.org/projects/security/pki/nss/`

- NSS Security Tools at
  http://www.mozilla.org/projects/security/pki/nss/tools/
- Overview of NSS at
  http://www.mozilla.org/projects/security/pki/nss/overview.html

Managing file Realm Users

Manage `file` realm users with the Admin Console. Users and groups in the `file` realm are listed in the keyfile, whose location is specified by the `file` property.

Note: It is also possible to use these steps to add users to any file realm, including the `admin-realm`. Simply substitute the name of the target realm in place of the `file` realm referenced in this section.

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`, in the realm `admin-realm`, which gives rights to modify the Application Server. Any users assigned to this group will have administrator privileges, that is, they will have access to the `asadmin` tool and the Admin Console.

To manage `file` realm users, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Expand the Security node.
4. Expand the Realms node.
5. Select the `file` node.
6. Click the Manage Users button from the Edit Realm page.

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

   - Adding a User

- Editing a User
- Deleting a User

Adding a User

In the File Users page, add a new user by following these steps:

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, change a user's information by following these steps:

1. In the User ID column, click the name of the user to be modified.

   The Edit File Realm User page displays.
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 Close to quit without saving.

Deleting a User

In the File Users page, delete a user by following these steps:

1. Select the checkbox to the left of the name of the user(s) to be deleted.
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 "About Certificate Files"). Add users to these files using `certutil`. 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 certutil documentation at "About the CertUtil Utility".

Table 14-9 lists the optional properties for the `certificate` realm.

Table 14-9 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`.

Configuring Mutual Authentication

- Enabling Mutual Authentication for all Applications
- Enabling Mutual SSL Authentication in an Application

In mutual authentication, both server and client-side authentication are enabled. To test mutual authentication, a client with a valid certificate must exist. For information on mutual authentication, see the *Security* chapter of *The J2EE 1.4 Tutorial* at:

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

Enabling Mutual Authentication for all Applications

The Application Server uses the `certificate` realm for HTTPS authentication.

To specify mutual authentication for all the applications that use this realm, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.

2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Expand the Security node.
4. Expand the Realms node.
5. Select the `certificate` realm.
6. Click the Add Property button.
   - In the Name field, enter `clientAuth`.
   - In the Value field, enter `true`.
7. Click Save.
8. Restart the Application Server if Restart Required displays in the console.

   After restarting the server, client authentication is required for all applications that use the `certificate` realm.

Enabling Mutual SSL Authentication in an Application

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 1.4 Tutorial* at:

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

Deleting a Realm

To delete a realm, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Expand the Security node.

4. Select the Realms node.
5. Click in the box beside the realm to be deleted.
6. Click Delete.

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

Setting the Default Realm

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

To set the default realm, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Select the Security node.

   The Security page displays.
4. In the Default Realm field, pick the desired realm from the drop-down list.
5. Click Save to save the changes or Load Defaults to delete changes and restore the Application Server default values.
6. Restart the server if Restart Required displays in the console.

Admin Console Tasks for JACC Providers

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 the administrator 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.

To create a JACC provider, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Expand the Security node.
4. Select the JACC Providers node.
5. On the JACC Providers page, click New.
6. On the Create JACC Provider page, enter the following:
   - **Name** – The name to use to identify this provider.
   - **Policy Configuration** – The name of the class that implements the policy configuration factory. The default provider uses `com.sun.enterprise.security.provider.PolicyConfigurationFactoryImpl`.
   - **Policy Provider** – The name of the class that implements the policy factory. The default provider uses `com.sun.enterprise.security.provider.PolicyWrapper`.
7. Add 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*`/domains/`*domain_dir*`/generated/policy`.
8. Click OK to save this configuration, or click Cancel to quit without saving.

Editing a JACC Provider

To edit a JACC provider, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Expand the Security node.
4. Expand the JACC Providers node.
5. Select the node of the JACC provider to be edited.
6. 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.
7. To add 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 `${com.sun.aas.instanceRoot}/generated/policy`.
8. To delete an existing property, click in the checkbox to the left of the property, then click Delete Properties.
9. Click Save to save or click the browser's back button to cancel without saving.

Deleting a JACC Provider

To delete a JACC provider, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.

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

3. Expand the Security node.
4. Select the JACC Providers node.
5. Click in the checkbox to the left of the JACC provider to be deleted.
6. Click Delete.

Setting the Active JACC Provider

To specify the JACC provider, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Select the Security node.

   The Security page displays.
4. In the JACC field, enter the name of the JACC provider to be used by the server.

   If you don't know which JACC providers are available, expand the JACC Provider component in the tree to view all configured JACC providers.
5. Select Save to save the changes or Load Defaults to return to the default values.
6. Restart the Application Server if Restart Required displays in the console.

Admin Console Tasks for Audit Modules

- 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, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Expand the Security node.
4. Select the Audit Modules node.
5. On the Audit Modules page, click New.
6. On the Create Audit Module page, enter the following information:
   - **Name** – The name used to identify this audit module.
   - **Classname** – The fully-qualified name of the class that implements this module. The class name for the default audit module is `com.sun.enterprise.security.Audit`.
7. To add JVM properties to this module, click Add Property. Specify a name and value for each property. Valid properties include:
   - `auditOn` - Specifies whether or not to enable this implementation class. Valid values are `true` and `false`.
8. Click OK to save entries, or click Cancel to quit without saving.

Editing an Audit Module

Audit modules are not turned on by default. For more information on how to activate audit modules, read "Enabling and Disabling Audit Logging".

To edit an audit module, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Expand the Security node.
4. Expand the Audit Modules node.
5. Click the node of the audit module to be edited.
6. On the Edit Audit Module page, modify the class name, if needed.
7. Enter any additional properties for the module by selecting the Add button and entering the name and value of the property. Valid properties include:
   - `auditOn` - Specifies whether or not to use this audit module. Valid values are `true` and `false`.
8. Modify any existing properties by selecting the name or value to be modified, and entering the changes directly into the text field.
9. Delete a property by selecting the checkbox to the left of the property and clicking Delete Properties.
10. Click Save to save or click the Back button on the browser to cancel without saving.

Deleting an Audit Module

To delete an audit module, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.

3. Expand the Security node.
4. Select the Audit Modules node.
5. Click in the checkbox to the left of the audit module to be deleted.
6. Click Delete.

Enabling and Disabling Audit Logging

To specify the audit module that the server uses, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Select the Security node.

   The Security page displays.
4. To enable logging, select the Audit Logging check box. To disable it, deselect it. Selecting this option causes the loading of the audit modules and ensures they are called by the Application Server's audit library at audit points.
5. If you are enabling audit logging, specify a default audit module as described in "Setting the Active Audit Module".
6. Select Save to save the changes.
7. Restart the Application Server if Restart Required displays in the console.

Setting the Active Audit Module

To specify the audit module that the server uses, enable audit logging as described in "Enabling and Disabling Audit Logging" and then follow these steps:

1. In the Audit Modules field, enter the name of the audit module to be used by the server. (The preconfigured audit module is called `default`.) Make sure that this audit module has auditOn set to true as described in "Enabling and Disabling the Default Audit Module".

2. Select Save to save the changes, Load Defaults to cancel.
3. Restart the Application Server if Restart Required displays in the console.

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 the Default Audit Module

In addition to enabling logging, set any properties required by the specific audit modules required. In the case of the default audit module, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Expand the Security node.
4. Expand the Audit Modules node.

5. Click the `default` node.
6. Set the value of the `auditOn` property to `true`.
7. Select Save to save the changes.
8. Restart the Application Server if Restart Required displays in the console.

Admin Console Tasks for Listeners and JMX Connectors

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, create new HTTP listeners and edit the security settings of existing HTTP listeners.

To edit security settings for an existing HTTP listener, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Expand the HTTP Service node.
4. Select the HTTP Listeners node.
5. Select an HTTP listener to edit an existing listener or click New and follow the procedure in "Creating an HTTP Listener" to create a new listener.
6. Follow the procedure in "Setting Listener Security Properties" to set security properties.

7. Click Save to save the changes, or click the browser's Back button to cancel without saving.

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, create new IIOP listeners and edit the settings of existing IIOP listeners.

To edit security properties for an IIOP listener, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Expand the ORB node.
4. Select the IIOP Listeners node.
5. Select an IIOP listener to edit that listener or click New and follow the procedure in "Creating an IIOP Listener" to create a new listener.
6. Follow the procedure in "Setting Listener Security Properties" to set security properties.
7. Click Save to save the changes, or click Load Defaults to restore the properties to their default values.

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

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

Configuring Security for the Admin Service's JMX Connector

To edit security properties for a JMX Connector in the Admin Service, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Expand the Admin Service node.
4. Select the admin service to be modified.
5. Follow the procedure in "Setting Listener Security Properties" to set security properties.
6. Click Save to save the changes, or click Load Defaults to restore the properties to their default values.

Setting Listener Security Properties

Follow this common procedure for setting HTTP listener, IIOP listener, and JMX Connector security properties:

1. In the Edit HTTP Listener, Edit IIOP Listener, or Edit JMX Connector page, go to the section labeled SSL.
2. Check the Enabled box in the Security field to enable security for this listener. When this option is selected, you must select SSL3 or TLS to specify which type of security is enabled, and you must enter a certificate nickname.
3. Check the Enabled box in the Client Authentication field if clients are to authenticate themselves to the Application Server when using this listener.

4. Enter the keystore alias in the Certificate Nickname field if the Enabled box is checked. The keystore alias is a single value that identifies an existing server keypair and certificate. The certificate nickname for the default keystore is `s1as`.

   To find the Certificate Nickname, use the `certutil` utility, as described in "About the CertUtil Utility".

5. Select SSL3 and / or TLS if the Enabled box is checked. By default, both SSL3 and TLS are enabled.

6. Enable individual cipher suites, if needed. By default, all supported cipher suites are enabled. Ciphers are discusses in "About Ciphers".

7. Select Save to save the changes or Load Defaults to cancel.

Admin Console Security Tasks for Virtual Servers

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 be used only 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 uses 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 log in 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 these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Expand the instance to configure:
   a. To configure a particular instance, expand the instance's config node. For example, the default instance, `server`, expand the `server-config` node.
   b. To configure the default settings for all instances, expand the `default-config` node.
3. Expand the HTTP Service node.
4. Expand the Virtual Servers node, and select the virtual server to be configured for single sign-on support.
5. Click Add Property.

   A blank property entry is added to the bottom of the list.
6. Enter `sso-enable` in the Name field.
7. Enter `false` in the Value field to disable, enter `true` to enable SSO. SSO is enabled by default.
8. Add or change any other single sign-on properties by clicking Add Property and configuring any applicable SSO properties. Valid SSO properties are discussed in Table 14-10.

Table 14-10 Virtual Servers SSO 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 the server.
sso-reap-interval-seconds	Interval (in seconds) between purges of expired single sign-on records.	Default is 60.

9. Click Save.
10. Restart the Application Server if Restart Required displays in the console.

Admin Console Tasks for Connector Connection Pools

About Connector Connection Pools

A *connector module* (also called a resource adapter) enables J2EE applications to interact with enterprise information systems (EIS). 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 enables the creation of a mapping between J2EE users and groups and EIS users and groups. Use the Admin Console to create, update, list, and delete security maps for connector connection pools.

Note: In this context, users are referred to as principals. The enterprise information system (EIS) is any system that holds the information. It can be a mainframe, a messaging system, a database system, or an application.

About Security Maps

Use security maps to map the caller identity of the application (principal or user group) to a suitable EIS principal in container-managed transaction-based scenarios. 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. Security maps are used when an application user needs to execute EIS operations that require to be executed as a specific identity in the EIS.

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.

To create a security map for a given connector connection pool, follow these steps.

1. Expand the Resources node.
2. Expand the Connectors node.
3. Select the Connector Connection Pools node.
4. Select a Connector Connection Pool by selecting its name from the list of current pools or create a new connector connection pool by selecting New from the list of current pools and following the instructions in "Creating a Connector Connection Pool".
5. Select the Security Maps page.
6. Click New to create a new Security Map.
7. On the Create Security Map page, enter the following properties.
    - **Name** – Enter a name to be used to reference this particular security map.
    - **User Groups** – The caller identity of the application to be mapped to a suitable EIS principal. Enter a comma-separated list of application-specific user groups, or enter the wild card asterisk (*) to indicate all users or all user groups. Specify either the Principals or User Groups options, but not both.
    - **Principals** – The caller identity of the application to be mapped to a suitable EIS principal. Enter a comma-separated list of application-specific principals, or enter the wild card asterisk (*) to indicate all principals. Specify either the Principals or User Groups options, but not both.
8. In the Backend Principal section, enter the following properties.

- **Username** – Enter the EIS user name. The enterprise information system (EIS) is any system that holds the information. It can be a mainframe, a messaging system, a database system, or an application.
- **Password** – Enter the password for the EIS user.

9. Click OK to create the security map or Cancel to quit without saving.

Equivalent `asadmin` command: `create-connector-security-map`

Editing a Security Map

To modify a security map for a given connector connection pool, follow these steps.

1. Expand the Resources node.
2. Expand the Connectors node.
3. Select the Connector Connection Pools node.
4. Select a Connector Connection Pool by selecting its name from the list of current pools.
5. Select the Security Maps page.
6. On the Security Maps page, select a security map from the list of current security maps.
7. On the Edit Security Map page, modify the following properties where needed.
   - **User Groups** – The caller identity of the application to be mapped to a suitable EIS principal. Enter a comma-separated list of application-specific user groups, or enter the wild card asterisk (*) to indicate all users or all user groups. Specify either the Principals or User Groups options, but not both.
   - **Principals** – The caller identity of the application to be mapped to a suitable EIS principal. Enter a comma-separated list of application-specific principals, or enter the wild card asterisk (*) to indicate all principals. Specify either the Principals or User Groups options, but not both.
8. In the Backend Principal section, enter the following properties.
   - **Username** – Enter the EIS user name. The enterprise information system (EIS) is any system that holds the information. It can be a mainframe, a messaging system, a database system, or an application.
   - **Password** – Enter the password for the EIS user.

9. Click Save to save the changes to the security map.

Helpful `asadmin` command: `list-connector-security-maps`, `update-connector-security-maps`

Deleting a Security Map

To delete a security map for a given connector connection pool, follow these steps.

1. Expand the Resources node.
2. Expand the Connectors node.
3. Select the Connector Connection Pools node.
4. Select a Connector Connection Pool by selecting its name from the list of current pools.
5. Select the Security Maps page.
6. On the Security Maps page, click the checkbox to the left of the name of the security map to be deleted.
7. Click Delete.

Equivalent `asadmin` command: `delete-connector-security-map`

Working with Certificates and SSL

About Certificate Files

Installation of the Application Server generates a digital certificate in NSS format 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 `key3.db`, contains the Application Server's digital certificate, including its private key. The keystore file is protected with a password. Change the password using the `asadmin change-master-password` command. For more information about `certutil`, `read` "About the CertUtil Utility".

  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 `cert8.db`, contains the Application Server's trusted certificates, including public keys for other entities. For a trusted certificate, the server has confirmed that the public key in the certificate belongs to the certificate's owner. Trusted certificates generally include those of certification authorities (CAs).

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

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

Changing the Location of Certificate Files

The keystore and trust-store files provided for development are stored in the *install_dir*`/domains/`*domain_name*`/config` directory. To change the name and/or location of the keystore and trust-store files, follow these steps.

1. In the Admin Console tree, expand Configurations.
2. Expand the server-config (Admin Config) node.
3. Select the JVM Settings node.
4. Click the JVM Options tab.
5. On the JVM Options page, add or modify the following values in the Value field to reflect the new location of the certificate files:

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

where *ks_name* is the keystore file name and *ts_name* is the trust-store file name.

6. Click Save.
7. Restart the Application Server if Restart Required displays in the console.

About the Keytool Utility

Use `keytool` to set up and work with JSSE digital certificates in the Platform Edition. The J2SE SDK ships with `keytool`, thus allowing the administrator to administer public / private key pairs and associated certificates. It also enables users to cache the public keys (in the form of certificates) of their communicating peers.

To run `keytool`, the shell environment must be configured so that the J2SE `/bin` directory is in the path, or the full path to the tool must be present on the command line. For more information on `keytool`, see the `keytool` documentation at:

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

About the CertUtil Utility

Use `certutil` to set up and work with NSS digital certificates in the **Enterprise Edition** only. The Certificate Database Tool, `certutil`, is a command-line utility that can create and modify the Netscape Communicator `cert8.db` and `key3.db` database files. It can also list, generate, modify, or delete certificates within the `cert8.db` file and create or change the password, generate new public and private key pairs, display the contents of the key database, or delete key pairs within the `key3.db` file.

The key and certificate management process generally begins with creating keys in the key database, then generating and managing certificates in the certificate database. The document listed below discusses certificate and key database management with NSS, including the syntax for the `certutil` utility:

`http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html`

The command-line utility used to import and export keys and certificates between the certificate / key databases and files in PKCS12 format is `pk12util`. More description of the `pk12util` utility can be read at:

`http://www.mozilla.org/projects/security/pki/nss/tools/pk12util.html`

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

The tools are located in the *install_dir*`/lib/` directory.

Generating a Server Certificate

Use certutil to generate, import, and export certificates. Read "About the CertUtil Utility" for more information on how to do this.

Signing a Digital Certificate

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

Using a Certificate From a CA

To use a digital certificate signed by a CA:

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

   Save the certificate in the directory containing the server keystore and trust-store files, by default *install_dir*`/domains/`*domain-dir*`/config` directory. See "Changing the Location of Certificate Files" for instructions on changing this location.
3. In your shell, change to the directory containing the certificate.
4. Use `certutil` to import the certificate into the local keystore and, if necessary, the local trust-store.
5. Restart the Application Server.

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

`http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html`

Deleting a Certificate

To delete an existing certificate, use the `certutil` utility.For more information on the `certutil` utility, see "About the CertUtil Utility".

Further Information

- The Java 2 Standard Edition discussion of security can be viewed from: `http://java.sun.com/j2se/1.4.2/docs/guide/security/index.html`
- The *J2EE 1.4 Tutorial* chapter titled *Security* can be viewed from: `http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html`
- The *Administration Guide* chapter titled "Configuring Message Security".
- The *Developer's Guide* chapter titled *Securing ApplicationsSecuring Applications.*

Chapter 15

Configuring Message Security

This chapter describes the configuration of message layer security for web services in the Sun Java System Application Server 8.1 2005Q1. This chapter contains the following topics:

- About Message Security
- Admin Console Tasks for Message Security

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

About Message Security

- Overview of Message Security
- Understanding Message Security in the Application Server
- Securing a Web Service
- Securing the Sample Application
- Configuring the Application Server for Message Security

Overview of Message Security

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

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

The WS-Security specification can be viewed at the following URL:

`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf`

Understanding Message Security in the Application Server

The Sun Java System Application Server 8.1 2005Q1 offers integrated support for the WS-Security standard in its web services client and server-side containers. This functionality is integrated such that web services security is enforced by the containers of the Application Server on behalf of applications, and such that it can be applied to protect any web service application without requiring changes to the implementation of the application. The Application Server achieves this effect by providing facilities to bind SOAP layer message security providers and message protection policies to containers and to applications deployed in containers.

Assigning Message Security Responsibilities

In Sun Java System Application Server 8.1 2005Q1, the System Administrator and Application Deployer roles are expected to take primary responsibility for configuring message security. In some situations, the Application Developer may also contribute, although in the typical case either of the other roles may secure an existing application without changing its implementation without involving the developer. The responsibilities of the various roles are defined in the following sections:

- System Administrator
- Application Deployer
- Application Developer

System Administrator

The system administrator is responsible for:

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

A system administrator uses the Admin Console to manage server security settings and uses a command line tool to manage certificate databases. In PE, certificates and private keys are stored in keystores and are managed with `keytool`. SE and EE store certificates and private keys in an NSS database, where they are managed using `certutil`. This document is intended primarily for system administrators. For an overview of message security tasks, see “Configuring the Application Server for Message Security”.

Application Deployer

The application deployer is responsible for:

- Specifying (at application assembly) any required application-specific message protection policies if such policies have not already been specified by upstream roles (the developer or assembler).

- Modifying Sun-specific deployment descriptors to specify application-specific message protection policies information (i.e. message-security-binding elements) to web service endpoint and service references.

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

Application Developer

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

The application developer or assembler is responsible for the following:

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

About Security Tokens and Security Mechanisms

The WS-Security specification provides an extensible mechanism for using security tokens to authenticate and encrypt SOAP web services messages. The SOAP layer message security providers installed with the Application Server may be used to employ username/password and X509 certificate security tokens to authenticate and encrypt SOAP web services messages. Additional providers that employ other security tokens including SAML assertions will be installed with subsequent releases of the Application Server.

About Username Tokens

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

When using a Username token, a valid user database must be configured on the Application Server. For more information on this topic, read "Editing a Realm".

About Digital Signatures

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

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

About Encryption

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

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

About Message Protection Policies

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

Request and response message protection policies are defined when a provider is configured into a container. Application-specific message protection policies (at the granularity of the web service port or operation) may also be configured within the Sun-specific deployment descriptors of the application or application client. In any case, where message protection policies are defined, the request and response message protection policies of the client must match (i.e. be equivalent to) the

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

Glossary of Message Security Terminology

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

- Authentication Layer

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

- Authentication Provider

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

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

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

- Default Server Provider

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

- Default Client Provider

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

- Request Policy

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

- Response Policy

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

Securing a Web Service

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

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

The administrative interfaces of the Application Server can be employed to bind the existing providers for use by the server-side containers of the Application Server, to modify the message protection policies enforced by the providers, or to create new provider configurations with alternative message protection policies.

These operations are defined in "Admin Console Tasks for Message Security". Analogous administrative operations can be performed on the SOAP message layer security configuration of the application client container as defined in Enabling Message Security for Client Applications.

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

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

Configuring Application-Specific Web Services Security

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

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

Securing the Sample Application

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

The `xms` sample application is provided to demonstrate the use of the Application Server's WS-Security functionality to secure an existing web services application. The instructions which accompany the sample describe how to enable the WS-Security functionality of the Application Server such that it is used to secure

the xms application. The sample also demonstrates the binding of WS-Security functionality directly to the application (as described in Configuring Application-Specific Web Services Security) such that it applies specifically to the application.

The xms sample application is installed in the directory:
install_dir\samples\webservices\security\ejb\apps\xms\

For information on compiling, packaging, and running the xms sample application, refer to the *Securing Applications* chapter of the *Developers' Guide*. There is a link to this chapter in "Further Information".

Configuring the Application Server for Message Security

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

To configure these other facilities follow these steps:

1. If using a version of the Java SDK prior to version 1.5.0, and using encryption technology, configure a JCE provider.

   Configuring a JCE provider is discussed in "Configuring a JCE Provider".

2. If using a username token, configure a user database, if necessary. When using a username/password token, an appropriate realm must be configured and an appropriate user database must be configured for the realm.

   Configuring a user database is discussed in "Editing a Realm".

3. Manage certificates and private keys, if necessary.

   Managing certificates and private keys is discussed in "About Certificate Files".

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

Configuring a JCE Provider

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

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

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

1. Download and install a JCE provider JAR (Java ARchive) file. The following URL provides a list of JCE providers that support RSA encryption:

   `http://java.sun.com/products/jce/jce14_providers.html`

2. Copy the JCE provider JAR file to <*JAVA_HOME*>`/jre/lib/ext/`.

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

4. Edit the <*JAVA_HOME*>`/jre/lib/security/java.security` properties file in any text editor. Add the JCE provider you've just downloaded to this file. The `java.security` file contains detailed instructions for adding this provider. Basically, you need to add a line of the following format in a location with similar properties:

   ```
   security.provider.<n>=<provider class name>
   ```

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

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

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

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

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

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

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

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

5. Save and close the file.
6. Restart the Application Server.

Admin Console Tasks for Message Security

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

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

- Enabling Providers for Message Security
- Configuring a Message Security Provider
- Creating a Message Security Provider
- Deleting a Message Security Configuration
- Deleting a Message Security Provider

- Enabling Message Security for Client Applications

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

Enabling Providers for Message Security

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

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

To enable the default providers for the Application Server, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for all instances, select the `default-config` node.
3. Expand the Security node.
4. Expand the Message Security node.
5. Select the SOAP node.
6. Select the Message Security tab.
7. On the Edit Message Security Configuration page, specify a provider to be used on the server side and a provider to be used on the client side for all applications for which a specific provider has not been bound. This is accomplished by modifying the following optional properties:

- **Default Provider** – The identity of the server provider to be invoked for any application for which a specific server provider has not been bound.

  By default, no provider configuration is selected for the Application Server. To identify a server-side provider, select `ServerProvider`. Selecting the null option means that no message security provider will be invoked (by default) on the server side.

  You would generally select `ServerProvider` for this field.

- **Default Client Provider** – The identity of the client provider to be invoked for any application for which a specific client provider has not been bound.

  By default, no provider configuration is selected for the Application Server. To identify a client-side provider, select `ClientProvider`. Selecting the null option means that no message security provider will be invoked (by default) on the client side.

  You would generally select null for this field. You would select ClientProvider if you wanted to enable a default provider and message protection policy to apply to the web services invocations originating from web services endpoints deployed on the Application Server.

8. Click Save.
9. If you enabled a client or server provider, and you want to modify the message protection policies of the enabled providers, then refer to "Configuring a Message Security Provider" for information on modifying the configuration of the message security providers enabled in this step.

Equivalent asadmin commands:

- To specify the default server provider:

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

- To specify the default client provider:

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

Configuring a Message Security Provider

Typically, a provider would be reconfigured to modify its message protection policies, although the provider type, implementation class, and provider-specific configuration properties may also be modified. Follow the steps listed below to reconfigure a message security provider.

1. In the Admin Console tree component, expand the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for all instances, select the `default-config` node.
3. Expand the Security node.
4. Expand the Message Security node.
5. Select the SOAP node.
6. Select the Providers tab.
7. Select the message security provider to edit. `ClientProvider` and `ServerProvider` ship with the Application Server.
8. In the Provider Config section of the Edit Provider Config page, the following properties are available for modification:
   - **Provider Type** – Select `client`, `server`, or `client-server` to establish whether the provider is to be used as a client authentication provider, a server authentication provider, or both (a client-server provider).
   - **Class Name** - Enter the Java implementation class of the provider. Client authentication providers must implement the `com.sun.xml.wss.provider.ClientSecurityAuthModule` interface. Server-side providers must implement the `com.sun.xml.wss.provider.ServerSecurityAuthModule` interface. A provider may implement both interfaces, but it must implement the interface corresponding to its provider type.
9. In the Request Policy section of the Create a Provider Configuration page, enter the following **optional** values, if needed. These properties are optional, but if not specified, no authentication is applied to request messages.

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

- **Authentication Source** – Select `sender`, `content`, or null (the blank option) to define a requirement for message-layer sender authentication (for example, username password), content authentication (for example, digital signature), or no authentication be applied to request messages. When null is specified, source authentication of the request is not required.
- **Authentication Recipient** – Select `beforeContent` or `afterContent` to define a requirement for message-layer authentication of the receiver of the request message to its sender (e.g. by XML encryption). When the value is not specified it defaults to `afterContent`.

For a description of the actions performed by the SOAP message security providers as a result of the following message protection policies see "Actions of Request and Response Policy Configurations".

10. In the Response Policy section of the Create a Provider Configuration page, enter the following **optional** properties, if needed. These properties are optional, but if not specified, no authentication is applied to response messages.

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

    - **Authentication Source** – Select `sender`, `content`, or null (the blank option) to define a requirement for message-layer sender authentication (for example, username password) or content authentication (for example, digital signature) to be applied to response messages. When null is specified, source authentication of the response is not required.
    - **Authentication Recipient** – Select `beforeContent` or `afterContent` to define a requirement for message-layer authentication of the receiver of the response message to its sender (e.g. by XML encryption). When the value is not specified it defaults to `afterContent`.

    For a description of the actions performed by the SOAP message security providers as a result of the following message protection policies see "Actions of Request and Response Policy Configurations".

11. Add additional properties by clicking the Add Property button. The provider that is shipped with the Application Server supports the property listed below. If other providers are used, refer to their documentation for more information on properties and valid values.
    - `server.config` - The directory and file name of an XML file that contains the server configuration information. For example, *install_dir*`/domains/`*domain_dir*`/config/wss-server-config.xml`.
12. Click Save.

Equivalent `asadmin` commands are listed below. To set the response policy, replace the word `request` in the following commands with `response`.

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

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

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

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

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

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

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

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

Creating a Message Security Provider

To create a new message security provider, follow these steps. To configure an existing provider, follow the steps in "Configuring a Message Security Provider".

1. In the Admin Console tree component, expand the Configurations node.

2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for all instances, select the `default-config` node.
3. Expand the Security node.
4. Expand the Message Security node.
5. Select the SOAP node.
6. Select the Providers tab.
7. On the Provider Configuration page, click New.
8. In the Provider Config section of the Create a Provider Configuration page, enter the following:
   - **Default Provider** – Check the box beside this field to make the new message security provider the provider to be invoked for any application for which a specific provider has not been bound. Whether the provider becomes the default client provider, the default server provider, or both will be based on the value selected for Provider Type.
   - **Provider Type** – Select `client`, `server`, or `client-server` to establish whether the provider is to be used as a client authentication provider, a server authentication provider, or both (a client-server provider).
   - **Provider ID** - Enter an identifier for this provider configuration. This name will appear in the Current Provider Configurations list.
   - **Class Name** - Enter the Java implementation class of the provider. Client authentication providers must implement the `com.sun.xml.wss.provider.ClientSecurityAuthModule` interface. Server-side providers must implement the `com.sun.xml.wss.provider.ServerSecurityAuthModule` interface. A provider may implement both interfaces, but it must implement the interface corresponding to its provider type.
9. In the Request Policy section of the Create a Provider Configuration page, enter the following **optional** values, if needed. These properties are optional, but if not specified, no authentication is applied to request messages.

- **Authentication Source** – Select `sender`, `content`, or null (the blank option) to define a requirement for message-layer sender authentication (for example, username password), content authentication (for example, digital signature), or no authentication be applied to request messages. When null is specified, source authentication of the request is not required.
- **Authentication Recipient** – Select `beforeContent` or `afterContent` to define a requirement for message-layer authentication of the receiver of the request message to its sender (e.g. by XML encryption). When the value is not specified it defaults to `afterContent`.

For a description of the actions performed by the SOAP message security providers as a result of the following message protection policies see "Actions of Request and Response Policy Configurations".

10. In the Response Policy section of the Create a Provider Configuration page, enter the following **optional** properties, if needed. These properties are optional, but if not specified, no authentication is applied to response messages.
    - **Authentication Source** – Select `sender`, `content`, or null (the blank option) to define a requirement for message-layer sender authentication (for example, username password) or content authentication (for example, digital signature) to be applied to response messages. When null is specified, source authentication of the response is not required.
    - **Authentication Recipient** – Select `beforeContent` or `afterContent` to define a requirement for message-layer authentication of the receiver of the response message to its sender (e.g. by XML encryption). When the value is not specified it defaults to `afterContent`.

    For a description of the actions performed by the SOAP message security providers as a result of the following message protection policies see "Actions of Request and Response Policy Configurations".
11. Add additional properties by clicking the Add Property button. The provider that is shipped with the Application Server supports the property listed below. If other providers are used, refer to their documentation for more information on properties and valid values.
    - `server.config` - The directory and file name of an XML file that contains the server configuration information. For example, *install_dir*`/domains/`*domain_dir*`/config/wss-server-config.xml`.
12. Click OK to save this configuration, or click Cancel to quit without saving.

Equivalent `asadmin` command: `create-message-security-provider`

Actions of Request and Response Policy Configurations

Table 15-1 shows message protection policy configurations and the resulting message security operations performed by the WS-Security SOAP message security providers for that configuration.

Table 15-1 Message protection policy to WS-Security SOAP message security operation mapping

Message Protection Policy	Resulting WS-Security SOAP message protection operations
auth-source="sender"	The message contains a `wsse:Security` header that contains a `wsse:UsernameToken` (with password).
auth-source="content"	The content of the SOAP message Body is signed. The message contains a `wsse:Security` header that contains the message Body signature represented as a `ds:Signature`.
auth-source=”sender” auth-recipient="before-content" OR auth-recipient="after-content"	The content of the SOAP message Body is encrypted and replaced with the resulting `xend:EncryptedData`. The message contains a `wsse:Security` header that contains a `wsse:UsernameToken (with password)` and an `xenc:EncryptedKey`. The `xenc:EncryptedKey` contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.
auth-source="content" auth-recipient="before-content"	The content of the SOAP message Body is encrypted and replaced with the resulting xend:EncryptedData. The `xenc:EncryptedData` is signed. The message contains a `wsse:Security` header that contains an `xenc:EncryptedKey` and a `ds:Signature`. The `xenc:EncryptedKey` contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.
auth-source="content" auth-recipient="after-content"	The content of the SOAP message Body is signed, then encrypted, and then replaced with the resulting xend:EncryptedData. The message contains a `wsse:Security` header that contains an `xenc:EncryptedKey` and a `ds:Signature`. The `xenc:EncryptedKey` contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.
auth-recipient="before-content" OR auth-recipient="after-content"	The content of the SOAP message Body is encrypted and replaced with the resulting xend:EncryptedData. The message contains a `wsse:Security` header that contains an `xenc:EncryptedKey`. The `xenc:EncryptedKey` contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.
No policy specified.	No security operations are performed by the modules.

Deleting a Message Security Configuration

To delete a message security configuration, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for all instances, select the `default-config` node.
3. Expand the Security node.
4. Select the Message Security node.
5. Click in the checkbox to the left of the Message Security Configuration to be deleted.
6. Click Delete.

Deleting a Message Security Provider

To delete a message security provider, follow these steps.

1. In the Admin Console tree component, expand the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for all instances, select the `default-config` node.
3. Expand the Security node.
4. Expand the Message Security node.
5. Select the SOAP node.
6. Select the Providers page.
7. Click in the checkbox to the left of the Provider Configuration to be deleted.
8. Click Delete.

Equivalent `asadmin` command: `delete-message-security-provider`

Enabling Message Security for Client Applications

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

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

To enable a default client provider in the application client, follow these steps:

1. Stop any client applications that depend on the client container descriptor.
2. In a text editor, open the Sun application client container descriptor, located in *install_dir*/domains/*domain_dir*/config/sun-acc.xml.
3. Add the text in **bold** to the file to enable the default client provider in the application client. The other code is provided to show where the code to enable message security for client applications should be located. The code that is not in bold may differ slightly in your installation, do not change the text that is not in bold.

```
<client-container>
  <target-server name="<your_host>" address="<your_host>" port="<your_port>"/>
  <log-service file="" level="WARNING"/>
  <message-security-config auth-layer="SOAP"
    default-client-provider="ClientProvider">
    <provider-config
      class-name="com.sun.xml.wss.provider.ClientSecurityAuthModule"
      provider-id="ClientProvider" provider-type="client">
      <request-policy auth-source="sender"/>
      <response-policy/>
       <property name="security.config"
      value="C:/Sun/AppServer/lib/appclient/wss-client-config.xml"/>
    </provider-config>
  </message-security-config>
</client-container>
```

The message security provider configured in the client container will also require access to private keys and trusted certificates. This is accomplished by defining appropriate values for the following system properties in the application client startup script.

-Djavax.net.ssl.keyStore

-Djavax.net.ssl.trustStore

Setting the Request and Response Policy for the Application Client Configuration

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

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

To set the request policy for the application client configuration, modify the Sun Java System Application Server-specific configuration for the application client container as described in "Enabling Message Security for Client Applications". In the application client configuration file, add the text in **bold** to set the request policy. The other code is provided for reference. The code that is not in bold may differ slightly in your installation, do not change the text that is not in bold.

```
<client-container>
  <target-server name="<your_host>" address="<your_host>" port="<your_port>"/>
  <log-service file="" level="WARNING"/>
  <message-security-config auth-layer="SOAP"
    default-client-provider="ClientProvider">
    <provider-config
      class-name="com.sun.xml.wss.provider.ClientSecurityAuthModule"
      provider-id="ClientProvider" provider-type="client">
      <request-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
      <response-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
       <property name="security.config"
      value="install_dir/lib/appclient/wss-client-config.xml"/>
    </provider-config>
  </message-security-config>
</client-container>
```

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

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

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

Further Information

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

Chapter 16

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

What is a Transaction?

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

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

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

Transactions in J2EE Technology

Transaction processing in J2EE technology involves the following five participants:

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

Each of these entities contribute to reliable transaction processing by implementing the different APIs and functionalities, discussed below:

- The Transaction Manager provides the services and management functions required to support transaction demarcation, transactional resource management, synchronization, and transaction context propagation.
- The Application Server provides the infrastructure required to support the application 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

The Application Server handles transactions based on the settings in the Admin Console.

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. Application Server is designed to recover from these failures and complete the transactions upon server startup.

While performing the recovery, if some of the resources are unreachable the server restart may be delayed as it tries to recover the transactions.

When the transaction spans across servers, the server that started the transaction can contact the other servers to get the outcome of the transactions. If the other servers are unreachable, the transaction uses the Heuristic Decision field to determine the outcome.

To configure how the Application Server recovers from transactions:

1. In the tree component select the Configurations node.
2. Select the instance to configure:
    a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.

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

3. Select the Transaction Service node.
4. To enable the recovery of incomplete transactions, check the Recover in the On Restart field.
5. Set the amount of time, in seconds, the Application Server tries to connect to the unreachable server in the Retry Timeout field. The default value is 10 minutes (600 seconds).
6. Set the policy for unreachable servers in a transaction in the Heuristic Decision field.

   Unless there is 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.
7. Click Save.
8. Restart the Application Server.

Transaction Timeouts

By default, the server does not timeout a transaction. That is, the server waits indefinitely for a transaction to complete. If you set a timeout value for transactions, if a transaction isn't completed within the configured time, the Application Server rolls back the transaction.

To set a timeout value:

1. In the tree component, select the Configurations node.
2. Select the instance to configure:

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

   b. To configure the default settings for all instances, select the `default-config` node.
3. Select the Transaction Service node.
4. Enter the number of seconds before the transaction times out, in the Transaction Timeout field.

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

5. Click Save.
6. 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 human readable.

To set the location of the transaction logs:

1. In the tree component, select the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for all instances, select the `default-config` node.
3. Select the Transaction Service node.
4. Enter the location of the transaction logs in the Transaction Log Location field.

   A `tx` subdirectory is created and transaction logs are kept under that directory.

   The default value is `${com.sun.aas.instanceRoot}/logs`. The `${com.sun.aas.instanceRoot}` variable is the name of the instance, and is set when you start an Application Server instance. To see the value of `${com.sun.aas.instanceRoot}`, click Actual Values.
5. Click Save.
6. 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, 2048) results in larger transaction log files, but fewer keypoint operations, and potentially better performance. A smaller keypoint interval (for example, 256) results in smaller log files but slightly reduced performance due to the greater frequency of keypoint operations.

To set the keypoint interval:

1. In the tree component select the Configurations node.

2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for all instances, select the `default-config` node.
3. Select the Transaction Service node.
4. Enter the number of transactions between keypoint operations in the Keypoint Interval field.

   The default value is 2048.
5. Click Save.
6. Restart the Application Server.

Chapter 17

Configuring the HTTP Service

This chapter describes how to configure virtual servers and HTTP listeners for the HTTP service component of the Application Server.

About the HTTP Service

What Is the HTTP Service?

The HTTP service is the component of the Application Server that provides facilities for deploying web applications and for making deployed web applications accessible by HTTP clients. (See “Deploying a Web Application” on page 117.) 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 the same physical server to host multiple Internet domain names. All virtual servers hosted on the same physical server share the Internet Protocol (IP) address of that physical server. A virtual server associates a domain name for a server (such as `www.aaa.com`) with the particular server on which the Application Server is running.

Note: Do not confuse an Internet domain with the administrative domain of the Application Server.

For instance, assume you want to host these domains on your physical server:

```
www.aaa.com
www.bbb.com
www.ccc.com
```

Assume also that `www.aaa.com`, `www.bbb.com`, and `www.ccc.com` have web modules `web1`, `web2`, and `web3`, respectively, associated with them.

This means that all of these URLs are handled by your physical server:

```
http://www.aaa.com:8080/web1
http://www.bbb.com:8080/web2
http://www.ccc.com:8080/web3
```

The first URL is mapped to virtual host `www.aaa.com`, the second URL is mapped to virtual host `www.bbb.com`, and the third is mapped to virtual host `www.ccc.com`.

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

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

For this mapping to work, make sure that `www.aaa.com`, `www.bbb.com`, and `www.ccc.com` all resolve to your physical server's IP address. They need to be registered with the DNS server for your network. In addition, on a UNIX system, add these domains to your `/etc/hosts` file (if the setting for `hosts` in your `/etc/nsswitch.conf` file includes `files`).

When the Application Server is started, it starts the following virtual servers automatically:

- A virtual server named `server`, which hosts all user-defined web modules
- A virtual server named `__asadmin`, which hosts all administration-related web modules (specifically, the Admin Console). This server is restricted; you cannot deploy web modules to this virtual server.

For development, testing, and deployment of web services in a non-production environment, `server` is often the only virtual server required. In a production environment, additional virtual servers provide hosting facilities for users and customers so that each appears to have its own web server, even though there is only one physical server.

HTTP Listeners

Each virtual server provides connections between the server and clients through one or more HTTP listeners. Each HTTP listener is a listen socket that has an IP address, a port number, a server name, and a default virtual server.

HTTP listeners must have a unique combination of port number and IP address. For example, an HTTP listener can listen on all configured IP addresses on a given port for a machine by specifying the IP address 0.0.0.0. Alternatively, the HTTP listener can specify a unique IP address for each listener, but use the same port.

Since an HTTP listener is a combination of IP address and port number, you can have multiple HTTP listeners with the same IP address and different port numbers (for example, 1.1.1.1:8081 and 1.1.1.1:8082), or with different IP addresses and the same port number (for example, 1.1.1.1:8081 and 1.2.3.4:8081, if your machine was configured to respond to both these addresses).

However, if an HTTP listener uses the 0.0.0.0 IP address, which listens on all IP addresses on a port, you cannot create HTTP listeners for additional IP addresses that listen on the same port for a specific IP address. For example, if an HTTP listener uses 0.0.0.0:8080 (all IP addresses on port 8080), another HTTP listener cannot use 1.2.3.4:8080.

Because the system running the Application Server typically has access to only one IP address, HTTP listeners typically use the 0.0.0.0 IP address and different port numbers, with each port number serving a different purpose. If the system does have access to more than one IP address, each address can serve a different purpose.

By default, when the Application Server starts, it has the following HTTP listeners:

- Two HTTP listeners named `http-listener-1` and `http-listener-2`, associated with the virtual server named `server`. The listener named `http-listener-1` does not have security enabled; `http-listener-2` has security enabled.
- An HTTP listener named `admin-listener`, associated with the virtual server named `__asadmin`. This listener has security enabled.

All these listeners use the IP address 0.0.0.0 and the port numbers specified as the HTTP server port numbers during installation of the Application Server. If the Application Server uses the default port number values, `http-listener-1` uses port 8080, `http-listener-2` uses port 8181, and `admin-listener` uses port 4849.

Each HTTP listener has a default virtual server. The default virtual server is the server to which the HTTP listener routes all request URLs whose host component does not match any of the virtual servers that are associated with the HTTP listener (a virtual server is associated with an HTTP listener by listing the HTTP listener in its `http-listeners` attribute).

In addition, specify the number of acceptor threads in the HTTP listener. Acceptor threads are threads that wait for connections. The threads accept connections and put them in a queue, called the connection queue, where they are then picked up by worker threads. Configure enough acceptor threads so that there is always one available when a new request comes in, but few enough so that they do not provide too much of a burden on the system. The connection queue includes both the new connections just accepted by acceptor threads and persistent connections managed by the Keep-Alive connection management subsystem.

A set of request processing threads retrieves incoming HTTP requests from the connection queue and processes the requests. These threads parse the HTTP headers, select the appropriate virtual server, and run through the request processing engine to service the request. When there are no more requests to process, but the connection can be kept persistent (either by using HTTP/1.1 or sending a `Connection: keep-alive` header), the request processing thread assumes the connection to be idle and passes the connection to the Keep-Alive connection management subsystem.

The Keep-Alive subsystem periodically polls such idle connections and queues those connections with activity into the connection queue for future processing. From there, a request processing thread again retrieves the connection and processes its request. The Keep-Alive subsystem is multi-threaded, as it manages potentially tens of thousands of connections. Efficient polling techniques are used, by dividing the number of connections into smaller subsets, to determine which connections are ready with requests and which of those connections have idled for sufficient time to deem them closed (beyond a maximum permissible Keep-Alive timeout).

The HTTP listener's server name is the host name that appears in the URLs the server sends to the client as part of a redirect. This attribute affects URLs the server automatically generates; it does not affect the URLs for directories and files stored in the server. This name is normally the alias name if the server uses an alias. If a client sends a `Host:` header, that host name supersedes the HTTP listener's server name value in redirects.

Specify a redirect port to use a different port number from that specified in the original request. A *redirect* occurs in one of these situations:

- If a client tries to access a resource that no longer exists at the specified URL (that is, the resource has moved to another location), the server redirects the client to the new location (instead of returning a 404), by returning a designated response code and including the new location in the response's Location header.
- If a client tries to access a resource that is protected (for example, SSL) on the regular HTTP port, the server redirects the request to the SSL-enabled port. In this case, the server returns a new URL in the Location response header, in which the original nonsecure port has been replaced with the SSL-enabled port. The client then connects to this new URL.

Specify also whether security is enabled for an HTTP listener and what kind of security is used (for example, which SSL protocol and which ciphers).

To access a web application deployed on the Application Server, use the URL `http://localhost:8080/` (or `https://localhost:8181/` if it is a secure application), along with the context root specified for the web application. To access the Admin Console, use the URL `https://localhost:4849/` or `https://localhost:4849/asadmin/` (its default context root).

Because a virtual server must specify an existing HTTP listener, and because it cannot specify an HTTP listener that is already being used by another virtual server, create at least one HTTP listener before creating a new virtual server.

Admin Console Tasks for the HTTP Service

Configuring the HTTP Service

To configure the HTTP service, follow these steps:

1. In the tree component, expand the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.

b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.

3. Select the HTTP Service node.

4. On the HTTP Service page, you can set properties that apply to all of the service's HTTP listeners.

The following table lists these properties.

Table 17-1 HTTP Service Properties

Property Name	Description	Default Value
`traceEnabled`	If set to true, enables the TRACE operation. Set this property to false to make the Application Server less susceptible to cross-site scripting attacks.	false
`monitoringCacheEnabled`	If set to true, the Application Server caches local values of the statistics of the HTTP Service to answer statistics queries. This value improves performance. If set to false, the Application Server queries the HTTP Service for each statistics value.	true
`monitoringCacheRefreshInMillis`	Specifies the time interval, in milliseconds, after which the monitoring cache is refreshed.	5000
`sslCacheEntries`	Specifies the number of SSL sessions that can be cached. There is no upper limit.	10000
`sslSessionTimeout`	Specifies the number of seconds before an SSL2 session times out.	100
`ssl3SessionTimeout`	Specifies the number of seconds before an SSL3 session times out.	86400
`sslClientAuthDataLimit`	Specifies the maximum amount of application data, in bytes, that is buffered during the client certificate handshake phase.	1048576
`sslClientAuthTimeout`	Specifies the number of seconds before the client certificate handshake phase times out.	60
`keepAliveQueryMeanTime`	Specifies the desired keep-alive latency (in milliseconds).	100
`keepAliveQueryMaxSleepTime`	Specifies the upper limit to the time slept (in milliseconds) after polling keep-alive connections for further requests.	100

Table 17-1 HTTP Service Properties *(Continued)*

Property Name	Description	Default Value
`stackSize`	Specifies the maximum stack size of the native thread.	OS/ Machine dependent
`statsProfilingEnabled`	If set to false, disables the recording of monitoring statistics by the HTTP Service, which improves performance. If t this property is set to false, enabling monitoring for the HTTP service has no effect.	true
`chunkedRequestBufferSize`	Specifies the default buffer size, in bytes, for un-chunking request data.	8192
`chunkedRequestTimeoutSeconds`	Specifies the default timeout (in seconds) for un-chunking request data.	60
`dnsCacheEnabled`	If set to true, allows the user to monitor statistics related to DNS caching. This property takes effect only if the DNS Lookup box on the HTTP Protocol tab is selected. Otherwise, the property setting is ignored.	false

5. Click the Access Log tab to configure access log rotation. Click the other tabs to configure request processing, the Keep-Alive subsystem, the connection pool, the HTTP protocol, and the HTTP file cache.
6. Click Save.

Configuring the HTTP Service Access Log

Use this page to enable and configure rotation for the access logs for the virtual servers. These logs are in the *domain_root_dir*`/`*domain_dir*`/logs/access` directory and are named as follows:

virtual_server_name`_access_log.`*yyyy*`-`*mm*`-`*dd*`.txt`

Click Load Defaults to load the default values. To change the rotation properties for these logs, do the following:

- Check the File Rotation box to turn on file rotation. By default, file rotation is enabled.

- From the Rotation Policy drop-down list, choose a policy. (The only policy available is `time`.)
- In the Rotation Interval field, type a numeric value to specify the number of minutes between rotations of the access log. This field is valid only if the Rotation Policy is `time`. The default is 1440 minutes.
- In the Rotation Suffix field, type a string value to specify the suffix to be added to the log file name after rotation. The default is `%YYYY;%MM;%DD;-%hh;h%mm;m%ss;s`.
- In the Format field, enter a string value to specify the format of the access log. Use the formats shown in the following table. The default format is `%client.name% %auth-user-name% %datetime% %request% %status% %response.length%`.

Table 17-2 Token Values for Access Log Format

Data	Token
Client Host Name	`%client.name%`
Client DNS	`%client.dns%`
System Date	`%datetime%`
Full HTTP Request line	`%request%`
Status	`%status%`
Response Content Length	`%response.length%`
Referer Header	`%header.referer%`
User-agent	`%header.user-agent%`
HTTP Method	`%http-method%`
HTTP URI	`%http-uri%`
HTTP Query String	`%query-str%`
HTTP Protocol Version	`%http-version%`
Accept Header	`%header.accept%`
Date Header	`%header.date%`
If-Modified-Since Header	`%header.if-mod-since%`
Authorization Header	`%header.auth%`
Any valid HTTP header value defined in RFC 2616 (any is also a valid header value; it is specified as a variable here)	`%header.`*`any`*`%`
Name of Authorized User	`%auth-user-name%`

Table 17-2 Token Values for Access Log Format *(Continued)*

Data	Token
Value of a Cookie	`%cookie.value%`
Virtual Server ID	`%vs.id%`

- Click Save to save the changes, or Load Defaults to return to the default settings.

Configuring HTTP Service Request Processing Threads

Use this page to configure request processing threads for the HTTP service:

- Click Load Defaults to load the default values.
- In the Thread Count field, type a numeric value to specify the maximum number of request processing threads. The default is 128.
- In the Initial Thread Count field, type the number of request processing threads that will be available when the server starts. The default is 48.
- In the Thread Increment field, type the number of request processing threads to be added when the number of requests exceeds the initial thread count. The default is 10.
- In the Request Timeout field, type the number of seconds after which requests will time out. The default is 30 seconds.
- In the Buffer Length field, type the size (in bytes) of the buffer used by the request processing threads to read the request data. The default is 4096 bytes.
- Click Save to save the changes, or Load Defaults to return to the default settings.

Configuring the HTTP Service Keep-Alive Subsystem

Use this page to configure the Keep-Alive subsystem of the HTTP service.

- Click Load Defaults to load the default values.

- In the Thread Count field, type the number of keep-alive threads to be used. The default is 1.
- In the Max Connections field, type the maximum number of persistent connections to be maintained. The default is 256.
- In the Time Out field, type the maximum number of seconds for which a keep-alive connection should be kept open. The default is 30 seconds.
- Click Save to save the changes, or Load Defaults to return to the default settings.

See Also:

- HTTP Listeners

Configuring the HTTP Service Connection Pool

Use this page to configure the connection pool for the HTTP service connection queue:

- Click Load Defaults to load the default values.
- In the Max Pending Count field, type the maximum number of pending connections to be allowed for an HTTP listener. The default is 4096.
- In the Queue Size field, type the maximum size of the connection queue, in bytes. This value also specifies the maximum number of outstanding connections the server can have. The default is 4096.
- In the Receive Buffer Size field, type the size of the receive buffer for an HTTP listener. The default is 4096.
- In the Send Buffer Size field, type the size of the send buffer for an HTTP listener. The default is 8192.
- Click Save to save the changes, or Load Defaults to return to the default settings.

Configuring the HTTP Protocol for the HTTP Service

Use this page to configure the HTTP protocol for the HTTP service:

- Click Load Defaults to load the default values.

- In the Version field, type the version of the HTTP protocol to be used (`HTTP/1.0` or `HTTP/1.1`). The default is `HTTP/1.1`.
- Select the DNS Lookup box to enable lookup of the DNS entry for the client. The default is false.
- Remove the check from the SSL box to disable security in the server globally. Leave this value set to true to be able to use SSL for any listener that has security enabled. The default is true.
- In the Forced Response Type field, type the response type to be used if there is no MIME mapping available matching the extension. The default value is `text/html; charset=iso-8859-1`.
- In the Default Response Type field, type the default response type. The default value is `text/html; charset=iso-8859-1`. The value is a semicolon-delimited string consisting of content-type, encoding, language, and charset.
- Click Save to save the changes, or Load Defaults to return to the default settings.

Configuring the HTTP File Cache for the HTTP Service

Use this page to configure the HTTP file cache for the HTTP service.

The file cache stores static content so that the server handles requests for such content quickly.

- Click Load Defaults to load the default values.
- Check the Globally box to enable file caching. The default is true.
- Check the File Transmission box to enable the use of the `TransmitFileSystem` method on Windows. The default is false.
- In the Max Age field, type the maximum age, in seconds, of a valid cache entry. The default is 30 seconds.
- In the Max Files Count field, type the maximum number of files in the file cache. The default is 1024.
- In the Hash Init Size field, type the initial number of hash buckets. The default is zero.
- In the Medium File Size Limit field, type the maximum size, in bytes, of a file that can be cached as a memory mapped file. The default is 537,600 bytes.

- In the Medium File Size field, type the total size, in bytes, of all files that are cached as memory mapped files. The default is 10,485,760 bytes.
- In the Small File Size Limit field, type the maximum size, in bytes, of a file that can be read into memory. The default is 2048 bytes.
- In the Small File Size field, type the total size, in bytes, of all files that are read into memory. The default is 1,048,576 bytes.
- Choose `ON` or `OFF` from the File Caching Enabled drop-down list to enable or disable caching of file content if the size of the file is less than the medium file size limit. The default is `ON`.
- Click Save to save the changes, or Load Defaults to return to the default settings.

Admin Console Tasks for Virtual Servers

Creating a Virtual Server

To create a virtual server, follow these steps:

1. In the tree component, expand the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.
3. Expand the HTTP Service node.
4. Select the Virtual Servers node.
5. On the Virtual Servers page, click New. The Create Virtual Server page appears.

6. In the ID field, type a unique name for the virtual server. 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.
7. In the Hosts field, type the host name or names for the machine on which the server is running. 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).
8. In the area opposite State, select either On, Off, or Disabled. The default is On.
9. Leave the HTTP Listeners field empty. It is filled in automatically when you create an HTTP listener and associate it with this server.

   (Use of this field requires that you specify an existing HTTP listener. You must not, however, specify a listener that is used by another virtual server; if you do, an error appears in the server log. Since a listener must be associated with an existing virtual server when it is created, all existing listeners are used by another virtual server.)
10. From the Default Web Module drop-down list, choose the deployed web module (if any) that is to respond to all requests that cannot be mapped to other web modules deployed to the virtual server.

    If a Default Web Module is not specified, 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.
11. 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 to send logging messages to the default server log, *domain_root_dir*/*domain_dir*/`logs/server.log`.
12. 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 has the default properties `docroot` and `accesslog`, set to default values.
13. Click OK to save the virtual server.

The following table lists the available properties.

Table 17-3 Virtual Server Properties

Property Name	Description
`docroot`	Absolute path to root document directory for server. Default is *domain_root_dir*/*domain_dir*/`docroot`.

Table 17-3 Virtual Server Properties ***(Continued)***

Property Name	Description
`accesslog`	Absolute path to server access logs. Default is *domain_root_dir*`/`*domain_dir*`/logs/access`.
`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.

`allowLinking`	If true, resources that are symbolic links will be served for all web applications deployed on this virtual server. Individual web applications may override this setting by using the `sun-web-app` property `allowLinking` in the `sun-web.xml` file:
	`<sun-web-app>`
	`  <property name="allowLinking" value="{true|false}"/>`
	`</sun-web-app>`
	Default is true.

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

Editing a Virtual Server

To edit a virtual server, follow these steps:

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

2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.
3. Expand the HTTP Service node.
4. Select the Virtual Servers node.
5. Select the virtual server to be edited.
6. 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.
7. Click Save to save the changes.

Deleting a Virtual Server

To delete a virtual server, follow these steps:

1. In the tree component, expand the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.
3. Expand the HTTP Service node.
4. Select the Virtual Servers node.
5. On the Virtual Servers page, check the box next to the name of the virtual server to be deleted.

6. Click Delete.

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

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

Admin Console Tasks for HTTP Listeners

Creating an HTTP Listener

To create an HTTP listener, follow these steps:

1. In the tree component, expand the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.
3. Expand the HTTP Service node.
4. Select the HTTP Listeners node.
5. On the HTTP Listeners page, click New. The Create HTTP Listener page appears.
6. In the Name field, type a name for the listener.
7. In the Listener field, remove the check from the Enabled box if you do not want to enable the listener when the server restarts.
8. 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.

9. 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.
10. Choose a virtual server from the Default Virtual Server drop-down list.
11. In the Server Name field, type the host name to be used in the URLs the server sends to the client. This name is the alias name if your server uses an alias. If your server does not use an alias, leave this field empty.
12. In the Advanced area, perform any of the following tasks:
    - To redirect requests to another port, type a value in the Redirect Port field. The Application Server automatically redirects the request if these two conditions exist:
      - This listener is supporting non-SSL requests.
      - 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.
    - Remove the check from the Powered By box 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 *Application Server Developer’s Guide*. See “Further Information” on page 51 for a link to this document.

      Production environments might decide to omit the generation of `X-Powered-By` headers to hide their underlying technology.
13. To create a listener that is not secure, click OK.

In the SSL 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. To force 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) to be enabled on the listener. Check either SSL3 or TLS, or both.
    b. Check the cipher suite used by the protocol(s). To enable all cipher suites, check All Supported Cipher Suites. You can also enable individual cipher suites.

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

Equivalent `asadmin` commands: `create-http-listener`, `create-ssl`

Editing an HTTP Listener

To edit an HTTP listener, follow these steps:

1. In the tree component, expand the Configurations node.
2. Select the instance to configure:
    a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.
    b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.
3. Expand the HTTP Service node.
4. Select the HTTP Listeners node.
5. Select the HTTP listener to be edited.
6. On the Edit HTTP Listener page, modify any of the settings.

7. Click Save to save the changes.

Deleting an HTTP Listener

To delete an HTTP listener, follow these steps:

1. In the tree component, expand the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.
3. Expand the HTTP Service node.
4. Select the HTTP Listeners node.
5. On the HTTP Listeners page, check the box next to the name of the HTTP listener to be deleted.
6. Click Delete.

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, first copy the `http-listener` elements of the Application Server's `domain.xml` file to a safe place so that the settings can be restored if needed.

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

Chapter 18

Configuring 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

CORBA

The Application Server supports a standard set of protocols and formats that ensure interoperability. Among these protocols are those defined by CORBA.

The CORBA (Common Object Request Broker Architecture) model is based on clients requesting services from distributed objects or servers through a well-defined interface by issuing requests to the objects in the form of remote method requests. A remote method request carries information about the operation that needs to be performed, including the object name (called an object reference)

of the service provider and parameters, if any, for the invoked method. CORBA automatically handles network programming tasks such as object registration, object location, object activation, request de-multiplexing, error-handling, marshalling, and operation dispatching.

What is the ORB?

The Object Request Broker (ORB) is the central component of CORBA. The ORB provides the required infrastructure to identify and locate objects, handle connection management, deliver data, and request communication.

A CORBA object never talks directly with another. Instead, the object makes requests through a remote stub to the ORB running on the local machine. The local ORB then passes the request to an ORB on the other machine using the Internet Inter-Orb Protocol (IIOP for short). The remote ORB then locates the appropriate object, processes the request, and returns the results.

IIOP can be used as a Remote Method Invocation (RMI) protocol by applications or objects using RMI-IIOP. Remote clients of enterprise beans (EJB modules) communicate with the Application Server via RMI-IIOP.

IIOP Listeners

An IIOP listener is a listen socket that accepts incoming connections from the remote clients of enterprise beans and from other CORBA-based clients. Multiple IIOP listeners can be configured for the Application Server. For each listener, specify a port number, a network address, and optionally, security attributes. For more information, see “Creating an IIOP Listener” on page 327.

Admin Console Tasks for the ORB

Configuring the ORB

1. In the tree component, expand the Configurations node.
2. Select the instance to configure:

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

b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.

3. Select the ORB node.
4. Choose the thread pool the ORB uses from the Thread Pool ID drop-down list.

   The ORB uses thread pools to respond to requests from remote clients of enterprise beans and other clients that communicate via RMI-IIOP. For more information, see "Thread Pools in the Application Server" on page 331 and "Creating Thread Pools" on page 332.
5. In the Max Message Fragment Size field, set the maximum fragment size for IIOP messages.

   Messages larger than this size are fragmented.
6. In the Total Connections field, set the maximum number of incoming connections for all IIOP listeners.
7. Select the Required checkbox if IIOP client authentication is required.
8. Click Save to save the changes, or Load Defaults to load the default values.
9. Restart the server.

Admin Console Tasks for IIOP Listeners

Creating an IIOP Listener

1. In the tree component, expand the Configurations node.
2. Select the instance to configure:

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

   b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.

3. Expand the ORB node.
4. Select IIOP Listeners.
5. Click New.
6. Enter a name to identify the listener in the Name field.
7. Enter the network address of the listener in the Network Address field.

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

8. In the Listener Port field, enter the port number upon which the listener is to listen.
9. Check the Enabled box in the Listener field to enable the listener.
10. In the Additional Properties area, provide values for properties required by applications.
11. To create a listener that is not 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. To force 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) to enable on the listener. Check either SSL3 or TLS, or enable both protocols.

   b. Check the cipher suite used by the protocol(s). To enable all cipher suites, check All Supported Cipher Suites. You can also enable individual cipher suites.

5. Click OK.

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

Equivalent `asadmin` commands: `create-iiop-listener`, `create-ssl`

Editing an IIOP Listener

1. In the tree component, expand the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.
3. Expand the ORB node.
4. Select the IIOP Listeners node.
5. Select the listener to be modified in the Current Listeners table.
6. Modify the listener's settings. See "Creating an IIOP Listener" on page 327 for descriptions of the fields that are modifiable.
7. If you changed the port number of the listener, restart the server.

Deleting an IIOP Listener

1. In the tree component, expand the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.
3. Expand the ORB node.
4. Select the IIOP Listeners node.
5. Check the listener(s) to be deleted in the Current Listeners table.
6. Click Delete.

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

Chapter 19

Thread Pools

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

About Thread Pools

This section describes thread pools and how the work 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. It is possible to assign specific thread pools to connector modules and to the ORB.

One thread pool can serve multiple connector modules and enterprise beans. Request threads handle user requests for application components. When the server receives a request, it assigns the request to a free thread from the thread pool. The thread executes the client's requests and returns results. For example, if the request needs to use a system resource that is currently busy, the thread waits until that resource is free before allowing the request to use that resource.

Specify the minimum and maximum number of threads that are reserved for requests from applications. The thread pool is dynamically adjusted between these two values. The minimum thread pool size that is specified signals the server to allocate at least that many threads in reserve for application requests. That number is increased up to the maximum thread pool size that is specified.

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

Avoid thread starvation, where one resource adapter or application occupies all threads in the Application Server, by dividing the Application Server threads into different thread-pools.

Admin Console Tasks for Thread Pools

Creating Thread Pools

1. In the tree component select the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for all instances, select the `default-config` node.
3. Select the Thread Pools node.
4. Under Current Pools click New.
5. Enter the name of the thread pool in the Thread Pool ID field.
6. 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.
7. 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.
8. Enter the number, in seconds, after which idle threads are removed from pool in the Idle Timeout field.

9. Enter the total number of work queues that are serviced by this thread pool in the Number of Work Queues field.
10. Click OK.
11. Restart the Application Server.

Equivalent `asadmin` command: `create-threadpool`

Editing Thread Pools

1. In the tree component select the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for all instances, select the `default-config` node.
3. Select the Thread Pools node.
4. Under Current Pools select the name of the thread pool to be changed.
5. 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.
6. 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.
7. Enter the number, in seconds, after which idle threads are removed from pool in the Idle Timeout field.
8. Enter the total number of work queues that are serviced by this thread pool in the Number of Work Queues field.
9. Click Save.
10. Restart the Application Server.

Deleting Thread Pools

1. In the tree component select the Configurations node.
2. Select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for all instances, select the `default-config` node.
3. Select the Thread Pools node.
4. Check the thread pool name to be deleted in the Current Pools table.
5. Click Delete.
6. Restart the Application Server.

Equivalent `asadmin` command: `delete-threadpool`

Chapter 20

Configuring 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

Log Records

The Application Server uses the Java 2 platform Logging API specified in JSR 047. Application Server logging messages are recorded in the server log, normally found at *domain_root_dir*/*domain_dir*/`logs/server.log`.

The *domain_root_dir*/*domain_dir*/`logs/` directory contains two other kinds of logs in addition to the server log. In the `access` subdirectory are the HTTP Service access logs, and in the `tx` subdirectory are the Transaction Service logs. For information about these logs, see "Configuring the HTTP Service Access Log" on page 309 and "Configuring Transactions" on page 299.

The components of the Application Server generate logging output. Application components can also generate logging output.

Application components may use the Apache Commons Logging Library to log messages. The platform standard JSR 047 API, however, is recommended for better log configuration.

Log records follow a uniform format:

[#|*yyyy-mm-dd*T*hh*:*mm*:*ss*.*SSS-Z*|*Log Level*|*ProductName_Version*|*LoggerName*|*Key Value Pairs*|*Message*|#]

For example:

```
[#|2004-10-21T13:25:53.852-0400|INFO|sun-appserver-ee8.1|javax.enterprise.
system.core|_ThreadID=13;|CORE5004: Resource Deployed:
[cr:jms/DurableConnectionFactory].|#]
```

In this example,

- [# and #] mark the beginning and end of the record.
- The vertical bar (|) separates the record fields.
- 2004-10-21T13:25:53.852-0400 specifies the date and time.
- The *Log Level* is INFO. This level may have any of the following values: SEVERE, WARNING, INFO, CONFIG, FINE, FINER, and FINEST.
- The *ProductName_Version* is sun-appserver-ee8.1.
- The *LoggerName* is a hierarchical logger namespace that identifies the source of the log module, in this case javax.enterprise.system.core.
- The *Key Value Pairs* are key names and values, typically a thread ID such as _ThreadID=14;.
- The *Message* is the text of the log message. For all Application Server SEVERE and WARNING messages and many INFO messages, it begins with a message ID that consists of a module code and a numerical value (in this case, CORE5004).

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

The Logger Namespace Hierarchy

The Application Server provides a logger for each of its modules. The following table lists the names of the modules and the namespace for each logger in alphabetical order, as they appear on the Log Levels page of the Admin Console (see "Configuring Log Levels" on page 339). The last three modules in the table do not appear on the Log Levels page.

Table 20-1 Application Server Logger Namespaces

Module Name	Namespace
Admin	javax.enterprise.system.tools.admin

Table 20-1 Application Server Logger Namespaces *(Continued)*

Module Name	Namespace
Classloader	`javax.enterprise.system.core.classloading`
CMP	`javax.enterprise.system.container.cmp`
Configuration	`javax.enterprise.system.core.config`
Connector	`javax.enterprise.resource.resourceadapter`
CORBA	`javax.enterprise.resource.corba`
Deployment	`javax.enterprise.system.tools.deployment`
EJB Container	`javax.enterprise.system.container.ejb`
JavaMail	`javax.enterprise.resource.javamail`
JAXR	`javax.enterprise.resource.webservices.registry`
JAX-RPC	`javax.enterprise.resource.webservices.rpc`
JDO	`javax.enterprise.resource.jdo`
JMS	`javax.enterprise.resource.jms`
JTA	`javax.enterprise.resource.jta`
JTS	`javax.enterprise.system.core.transaction`
MDB Container	`javax.enterprise.system.container.ejb.mdb`
Naming	`javax.enterprise.system.core.naming`
Node Agent (Enterprise Edition only)	`javax.ee.enterprise.system.nodeagent`
Root	`javax.enterprise`
SAAJ	`javax.enterprise.resource.webservices.saaj`
Security	`javax.enterprise.system.core.security`
Server	`javax.enterprise.system`
Synchronization (Enterprise Edition only)	`javax.ee.enterprise.system.tools.synchronization`
Util	`javax.enterprise.system.util`
Verifier	`javax.enterprise.system.tools.verifier`
Web Container	`javax.enterprise.system.container.web`
Core	`javax.enterprise.system.core`
System Output (`System.out.println`)	`javax.enterprise.system.stream.out`
System Error (`System.err.println`)	`javax.enterprise.system.stream.err`

Admin Console Tasks for Logging

Configuring General Logging Settings

1. In the tree component, expand the Node Agents or Configurations node.
2. Select a node agent, or select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.
3. For a node agent, select the Logger Settings tab. For a configuration, select the Logger Settings node.
4. On the Logging Settings page, use the following fields to customize logging:
   - **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 *domain_root_dir*/*domain_dir*/`logs/server.log`.
   - **Alarms**: To route `SEVERE` and `WARNING` messages through the JMX framework, select the Enabled checkbox.
   - **Write to System Log**: On Solaris and Linux systems only, to send logging output to the `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 is installed during server startup. The log records from the custom handler will have the format described in "Log Records" on page 335.

- **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 is 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 megabytes. The minimum value for the limit is 500 kilobytes; if you specify a lower value, the file rotates when it reaches 500 Kbytes. To turn off log file rotation, set the value to 0.
- **File Rotation Time Limit**: Rotate the server log after the specified number of minutes is reached. The default value is zero, which means that the file is rotated when it reaches the size specified in the File Rotation Limit field. *If you specify one or more minutes, the time limit takes precedence over the size limit.*

5. Click Save to save changes. Click View Log Files to view the server log.

Configuring Log Levels

1. In the tree component, expand the Node Agents or Configurations node.
2. Select a node agent, or select the instance to configure:
   a. To configure a particular instance, select the instance's config node. For example, for the default instance, `server`, select the `server-config` node.
   b. To configure the default settings for future instances that use a copy of `default-config`, select the `default-config` node.
3. For a node agent, select the Log Levels tab. For a configuration, select the Logger Settings node, then select the Log Levels tab.
4. On the Module Log Levels page, choose a new value from the drop-down list opposite the module or modules whose log level is to be changed. The default level is `INFO`, meaning that messages at that level or higher (`WARNING`, `SEVERE`) appear in the log. Choose 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. The property name is the logger namespace, and the value is one of the eight possible levels. For example, the property name could be `samples.logging.simple.servlet`, and the value could be `FINE`.

   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 the changes, or click Load Defaults to restore the default values.

Calls to `System.out.println` are logged at the `INFO` level using the logger name `javax.enterprise.system.stream.out`. Calls to `System.err.println` are 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 for use when the server restarts.

Viewing the Server Log

1. In the tree component, expand the node for the server instance whose log you want to view.
2. On the General Information page, click View Log Files.

Use the Search Criteria area to customize and filter the log viewer. Use the basic fields as follows:

- **Instance Name**: Choose an instance name from the drop-down list to view the log for that server instance. The default is the current server instance.

- **Log File**: Choose a log file name from the drop-down list to view the contents of that log. The default is `server.log`.
- **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 that appear. 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`

  If the From value is later than the To value, an error message appears.
- **Log Level**: To filter messages by log level, choose a log level from the drop-down list. By default, the display includes all messages that appear in the server log at the chosen log level and more severe levels. Select the checkbox labeled "Do not include more severe messages" to display messages at only the chosen level.

  To ensure that the messages you want to view appear in the server log, first set the appropriate log levels on the Log Levels page. See "Configuring Log Levels" on page 339.

  If you choose to filter log messages based on log level, only messages matching the specified filter criteria are shown. However, this filtering does not affect which messages are logged to the server log.

The most recent 40 entries in the server log appear, with the settings 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.

At the end of the list of entries, click the buttons to view earlier or later entries in the log file.

Click Advanced Search in the Search Criteria area to make additional refinements to the log viewer. Use the Advanced Options fields as follows:

- **Logger**: To filter by module, choose one or more namespaces from the drop-down list. Use shift-click or control-click to choose multiple namespaces.

  Selecting a namespace at a higher level selects all the namespaces below it. For example, selecting `javax.enterprise.system` also selects 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 a particular application, type the logger names in the text field, one per line. If the application has several modules, you can view any or all of them. For example, suppose the 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`.

  When you specify one or more custom loggers, messages from Application Server modules appear only if you specify them explicitly in the Logger area.

- **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. To refine the log viewer to show only the output from a single thread, specify that module's logger in the Custom Logger field, and then specify 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 drop-down list (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; to view them, click the `(details)` link for a message.

Click Basic Search to hide the Advanced Options area.

Chapter 21

Monitoring Components and Services

This chapter contains information about monitoring components using the Application Server Admin Console. This chapter contains the following sections:

About Monitoring

Monitoring in the Application Server

Use monitoring to observe the runtime state of various components and services deployed in a server instance of the Sun Java System Application Server Enterprise Edition 8.1 2005Q1. With the information on the state of runtime components and processes, it is possible to identify performance bottlenecks for tuning purposes, aid capacity planning, predict failures, do root cause analysis in case of failures, and ensure that everything is functioning as expected.

Turning monitoring on reduces performance by increasing overhead.

Overview of Monitoring

To monitor the Application Server, perform these steps:

1. Enable the monitoring of specific services and components using either the Admin Console or the `asadmin` tool.

   For more information on this step, refer to "Admin Console Tasks for Enabling and Disabling Monitoring".

2. View monitoring data for the specified services or components using either the Admin Console or the `asadmin` tool.

   For more information on this step, refer to "Admin Console Tasks for Viewing Monitoring Data".

About the Tree Structure of Monitorable Objects

The Application Server uses a tree structure to track monitorable objects. Because the tree of monitoring objects is dynamic, it changes as components are added, updated, or removed in the instance. The root object in the tree is the server instance name, for example, `server`. (In the Platform Edition, just one server instance is permitted.)

The following command displays the top level of the tree:

```
asadmin> list --monitor server
server.applications
server.http-service
server.connector-service
server.jms-service
server.jvm
server.orb
server.resources
server.thread-pools
```

The following sections describe these sub-trees:

- The Applications Tree
- The HTTP Service Tree
- The Connector Service Tree
- The Connector Service Tree
- The JMS Service 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 (*). For more information, refer to "EJB Container Statistics" and "Web Container Statistics".

Figure 21-1 Applications Node Tree Structure

```
applications
   |--- application1
   |    |--- ejb-module-1
   |    |        |--- ejb1 *
   |    |                 |--- cache (for entity/sfsb) *
   |    |                 |--- pool (for slsb/mdb/entity) *
   |    |                 |--- methods
   |    |                      |---method1 *
   |    |                      |---method2 *
   |    |                 |--- stateful-session-store (for sfsb)*
   |    |                 |--- timers (for s1sb/entity/mdb) *
   |    |--- web-module-1
   |    |        |--- virtual-server-1 *
   |    |                      |---servlet1  *
   |    |                      |---servlet2  *
   |--- standalone-web-module-1
   |    |        |----- virtual-server-2 *
   |    |                      |---servlet3 *
   |    |                      |---servlet4 *
   |    |        |----- virtual-server-3 *
   |    |                    |---servlet3 *(same servlet on different vs)
   |    |                     |---servlet5 *
   |--- standalone-ejb-module-1
   |    |        |--- ejb2 *
   |    |                 |--- cache (for entity/sfsb) *
   |    |                 |--- pool (for slsb/mdb/entity) *
   |    |                 |--- methods
   |    |                      |--- method1 *
   |    |                      |--- method2 *
   |--- application2
```

The HTTP Service Tree

The nodes of the HTTP service are shown in the following schematic. The nodes at which monitoring information is available are marked with an asterisk (*). See "The HTTP Service Tree".

Figure 21-2 HTTP Service Schematic (PE version)

```
http-service
        |--- virtual-server-1
           |--- http-listener-1 *
           |--- http-listener-2 *
        |--- virtual-server-2
           |--- http-listener-1 *
           |--- http-listener-2 *
```

Figure 21-3 HTTP Service Schematic (EE version)

```
http-service *
            |---connection-queue *
            |---dns *
            |---file-cache *
            |---keep-alive *
            |---pwc-thread-pool *
            |---virtual-server-1*
            |        |--- request *
            |---virtual-server-2*
            |        |--- request *
```

The Resources Tree

The resources node holds monitorable attributes for pools such as the JDBC connection pool and connector connection pool. The following schematic shows the top and child nodes for the various resource components. The nodes at which monitoring statistics are available are marked with an asterisk (*). See "JDBC Connection Pools Statistics" and "JMS/Connector Service Statistics".

Figure 21-4 Resources Schematic

```
resources
       |---connection-pool1(either connector-connection-pool or jdbc)*
       |---connection-pool2(either connector-connection-pool or jdbc)*
```

The Connector Service Tree

The connector services node holds monitorable attributes for pools such as the connector connection pool. The following schematic shows the top and child nodes for the various connector service components. The nodes at which monitoring statistics are available are marked with an asterisk (*). See "JMS / Connector Service Statistics".

Figure 21-5 Connector Service Schematic

```
connector-service
       |--- resource-adapter-1
       |      |-- connection-pools
       |      |      |-- pool-1 (All pool stats for this pool)
       |      |-- work-management (All work mgmt stats for this RA)
```

The JMS Service Tree

The JMS services node holds monitorable attributes for pools such as the connector connection pool. The following schematic shows the top and child nodes for the various JMS service components. The nodes at which monitoring statistics are available are marked with an asterisk (*).

Figure 21-6 JMS Service Schematic

```
jms-service
       |-- connection-factories [AKA conn. pools in the RA world]
       |      |-- connection-factory-1 (All CF stats for this CF)
       |-- work-management (All work mgmt stats for the MQ-RA)
```

The ORB Tree

The ORB node holds monitorable attributes for connection managers. The following schematic shows the top and child nodes for the ORB components. The nodes at which monitoring statistics are available are marked with an asterisk (*). See "Statistics for Connection Managers in an ORB".

Figure 21-7 ORB Schematic

```
orb
       |--- connection-managers
       |       |--- connection-manager-1 *
       |       |--- connection-manager-1 *
```

The Thread Pool Tree

The thread pool node holds monitorable attributes for connection managers. The following schematic shows the top and child nodes for the ORB components. The nodes at which monitoring statistics are available are marked with an asterisk (*). See "Thread Pools Statistics".

Figure 21-8 Thread Pool Schematic

```
thread-pools
       |   |--- thread-pool-1 *
       |   |--- thread-pool-2 *
```

About Statistics for Monitored Components and Services

This section describes the monitoring statistics that are available:

- EJB Container Statistics
- Web Container Statistics
- HTTP Service Statistics
- JDBC Connection Pools Statistics
- JMS / Connector Service Statistics
- Statistics for Connection Managers in an ORB
- Thread Pools Statistics
- Transaction Service Statistics
- Java Virtual Machine (JVM) Statistics
  - JVM Statistics in J2SE 5.0
- Production Web Container (PWC) Statistics

EJB Container Statistics

EJB statistics are described in Table 21-1.

Table 21-1 EJB Statistics

Attribute Name	Data Type	Description
`createcount`	Count Statistic	Number of times an EJB's `create` method is called.
`removecount`	Count Statistic	Number of times an EJB's `remove` method is 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 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 21-2.

Table 21-2 EJB Method Statistics

Attribute Name	Datatype	Description
`methodstatistic`	Time Statistic	Number of times an operation is called; the total time that is 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 for EJB Session Stores are listed in Table 21-3.

Table 21-3 EJB Session Store Statistics

Attribute Name	Datatype	Description
currentSize	Range Statistic	Number of passivated or checkpointed sessions currently in the store.
activationCount	Count Statistic	Number of sessions activated from the store.
activationSuccessCount	Count Statistic	Number of sessions successfully activated from the store
activationErrorCount	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.
passivationCount	Count Statistic	Number of sessions passivated (unactivated) using this store.
passivationSuccessCount	Count Statistic	Number of sessions successfully passivated using this store.
passivationErrorCount	Count Statistic	Number of sessions that could not be passivated using this store.
expiredSessionCount	Count Statistic	Number of expired sessions that were removed by this store.
passivatedBeanSize	Count Statistic	Total number of bytes passivated by this store, including total, minimum, and maximum.
passivationTime	Count Statistic	Time spent on passivating beans to the store, including the total, minimum, and maximum.
checkpointCount (EE only)	Count Statistic	Number of sessions checkpointed using this store.
checkpointSuccessCount (EE only)	Count Statistic	Number of sessions checkpointed successfully.
checkpointErrorCount (EE only)	Count Statistic	Number of sessions that couldn't be checkpointed.
checkpointedBeanSize (EE only)	Value Statistic	Total number of beans checkpointed by the store.
checkpointTime (EE only)	Time Statistic	Time spent on checkpointing beans to the store.

The statistics available for EJB pools are listed in Table 21-4.

Table 21-4 EJB Pool Statistics

Attribute Name	Data Type	Description
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 21-5.

Table 21-5 EJB Cache Statistics

Attribute Name	Datatype	Description
cachemisses	Bounded Range Statistic	The number of times a user request does 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.
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.

The statistics available for Timers are listed in Table 21-6.

Table 21-6 Timer Statistics

Statistic	Data Type	Description
`numtimerscreated`	CountStatistic	Number of timers created in the system.
`numtimersdelivered`	CountStatistic	Number of timers delivered by the system.
`numtimersremoved`	CountStatistic	Number of timers removed from the system.

Web Container Statistics

The web container fits into the tree of objects as shown in Figure 21-1. Web container statistics are displayed for each individual web application. Statistics available for the web container for Servlets are shown in Table 21-7 and statistics available for web modules are shown in Table 21-8.

Table 21-7 Web Container (Servlet) Statistics

Statistic	Units	Data Type	Comments
`errorcount`	Number	CountStatistic	Cumulative number of cases where the response code is greater than or equal to 400.
`maxtime`	Milliseconds	CountStatistic	The maximum amount of time the web container waits for requests.
`processingtime`	Milliseconds	CountStatistic	Cumulative value of the amount of time required to process 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.

Statistics available for web modules are shown in Table 21-8.

Table 21-8 Web Container (Web Module) Statistics

Statistic	Data Type	Comments
`jspcount`	CountStatistic	Number of JSP pages that have been loaded in the web module.
`jspreloadcount`	CountStatistic	Number of JSP pages that have been reloaded in the web module.

Table 21-8 Web Container (Web Module) Statistics *(Continued)*

Statistic	Data Type	Comments
`sessionstotal`	CountStatistic	Total number of sessions that have been created for the web module.
`activesessionscurrent`	CountStatistic	Number of currently active sessions for the web module.
`activesessionshigh`	CountStatistic	Maximum number of concurrently active sessions for the web module.
`rejectedsessionstotal`	CountStatistic	Total number of rejected sessions for the web module. This is the number of sessions that were not created because the maximum allowed number of sessions were active.
`expiredsessionstotal`	CountStatistic	Total number of expired sessions for the web module.
`sessionsize (EE only)`	AverageRange Statistic	Size of the session for the web module. Value is either high, low, or average, or is in bytes for serialized sessions.
`containerlatency (EE only)`	AverageRange Statistic	Latency for the web container's part of the overall latency request. Value is either high, low, or average.
`sessionpersisttime (EE only)`	AverageRange Statistics	Time (in ms, low, high, or average) taken to persist HTTP session state to back-end store for the web module.
`cachedsessionscurrent (EE only)`	CountStatistic	Current number of sessions cached in memory for the web module.
`passivatedsessionscurrent (EE only)`	CountStatistic	Current number of sessions passivated for the web module.

HTTP Service Statistics

The statistics available for the HTTP service are shown in Table 21-9. These statistics are applicable to the Platform Edition only. For statistics for the HTTP Service on the Enterprise Edition, see Table 21-32.

Table 21-9 HTTP Service Statistics (applicable to Platform Edition only)

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.
`errorcount`	Number	Count Statistic	The cumulative value of the error count, which represents the number of cases where the response code is greater than or equal to 400.
`maxsparethreads`	Number	Count Statistic	The maximum number of unused response processing threads that can exist.
`minsparethreads`	Number	Count Statistic	The minimum number of unused response processing threads that can exist.
`maxthreads`	Number	Count Statistic	The maximum number of request processing threads created by the listener.
`maxtime`	Milliseconds	Count Statistic	The maximum amount of time for processing 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

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 21-10.

Table 21-10 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	Range Statistic	Count Statistic	The total number of free connections in the pool as of the last sampling.
numconntimedout	Count Statistic	Bounded Range Statistic	The total number of connections in the pool that timed out between the start time and the last sample time.
averageconnwaittime	Number	Count Statistic	Indicates the average wait time of connections for successful connection request attempts to the connector connection pool.
waitqueuelength	Number	Count Statistic	Number of connection requests in the queue waiting to be serviced.
connectionrequestwaittime		Range Statistic	The longest and shortest wait times of connection requests. The current value indicates the wait time of the last request that was serviced by the pool.
numconncreated	Milliseconds	CountStatistic	The number of physical connections that were created since the last reset.

Table 21-10 JDBC Connection Pool Statistics *(Continued)*

Statistic	Units	Data Type	Description
numconndestroyed	Number	Count Statistic	Number of physical connections that were destroyed since the last reset.
numconnacquired	Number	Count Statistic	Number of logical connections acquired from the pool.
numconnreleased	Number	Count Statistic	Number of logical connections released to the pool.

JMS/Connector Service Statistics

The statistics available for the connector connection pool are shown in Table 21-11. Statistics for Connector Work Management are shown in Table 21-12.

Table 21-11 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.
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	Range Statistic	The total number of free connections in the pool as of the last sampling.
numconntimedout	Number	Count Statistic	The total number of connections in the pool that timed out between the start time and the last sample time.
averageconnwaittime	Number	Count Statistic	Average wait time of connections before they are serviced by the connection pool.
waitqueuelenght	Number	Count Statistic	Number of connection requests in the queue waiting to be serviced.

Table 21-11 Connector Connection Pool Statistics *(Continued)*

Statistic	Units	Data Type	Description
`connectionrequest waittime`		Range Statistic	The longest and shortest wait times of connection requests. The current value indicates the wait time of the last request that was serviced by the pool.
`numconncreated`	Milliseconds	CountStatistic	The number of physical connections that were created since the last reset.
`numconndestroyed`	Number	Count Statistic	Number of physical connections that were destroyed since the last reset.
`numconnacquired`	Number	Count Statistic	Number of logical connections acquired from the pool.
`numconnreleased`	Number	Count Statistic	Number of logical connections released to the pool.

Statistics available for Connector Work Management are listed in Table 21-12,

Table 21-12 Connector Work Management Statistics

Statistic	Data Type	Description
`activeworkcount`	Range Statistic	Number of work objects executed by the connector.
`waitqueuelength`	Range Statistic	Number of work objects waiting in the queue before executing.
`workrequestwaittime`	Range Statistic	Longest and shortest wait of a work object before it gets executed.
`submittedworkcount`	Count Statistic	Number of work objects submitted by a connector module.
`rejectedworkcount`	Count Statistic	Number of work objects rejected by the Application Server.
`completedworkcount`	Count Statistic	Number of work objects that were completed.

Statistics for Connection Managers in an ORB

The statistics available for the connection manager in an ORB are listed in Table 21-13.

Table 21-13 Connection Manager (in an ORB) Statistics

Statistic	Units	Data Type	Description
`connectionsidle`	Number	CountStatistic	Provides total number of connections that are idle to the ORB.
`connectionsinuse`	Number	CountStatistic	Provides total number of connections in use to the ORB.
`totalconnections`	Number	BoundedRange Statistic	Total number of connections to the ORB.

Thread Pools Statistics

The statistics available for the thread pool are shown in Table 21-14.

Table 21-14 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.
`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 21-15.

Table 21-15 Transaction Service Statistics

Statistic	Data Type	Description
`activecount`	CountStatistic	Number of transactions currently active.
`activeids`	String Statistic	The ID's of the transactions that are currently active. Every such transaction can be rolled back after freezing the transaction service.
`committedcount`	CountStatistic	Number of transactions that have been committed.
`rolledbackcount`	CountStatistic	Number of transactions that have been rolled back.
`state`	String Statistic	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 21-16.

Table 21-16 JVM Statistics

Statistic	Data Type	Description
`heapsize`	BoundedRange Statistic	The resident memory footprint with the higher and lower bounds of the JVM's memory heap size.
`uptime`	CountStatistic	The amount of time the JVM has been running.

JVM Statistics in J2SE 5.0

If the Application Server is configured to run on J2SE version 5.0 or higher, additional monitoring information can be obtained from the JVM. Set the monitoring level to LOW to enable the display of this additional information. Set the monitoring level to HIGH to also view information pertaining to each live thread in the system. More information on the additional monitoring features available in J2SE 5.0 is available in a document titled *Monitoring and Management for the Java Platform*, which is available from the following URL:

`http://java.sun.com/j2se/1.5.0/docs/guide/management/`

The J2SE 5.0 monitoring tools are discussed at:

`http://java.sun.com/j2se/1.5.0/docs/tooldocs/#manage`

The statistics available for class loading in the JVM in J2SE 5.0 are shown in Table 21-17.

Table 21-17 JVM Statistics for J2SE 5.0 - Class Loading

Statistic	Data Type	Description
`loadedclasscount`	CountStatistic	Number of classes that are currently loaded in the JVM.
`totalloadedclasscount`	CountStatistic	Total number of classes that have been loaded since the JVM began execution.
`unloadedclasscount`	CountStatistic	Number of classes that have been unloaded from the JVM since the JVM began execution.

The statistics available for compilation in the JVM in J2SE 5.0 are shown in Table 21-18.

Table 21-18 JVM Statistics for J2SE 5.0 - Compilation

Statistic	Data Type	Description
`totalcompilationtime`	CountStatistic	Accumulated time (in milliseconds) spent in compilation.

The statistics available for garbage collection in the JVM in J2SE 5.0 are shown in Table 21-19.

Table 21-19 JVM Statistics for J2SE 5.0 - Garbage Collection

Statistic	Data Type	Description
`collectioncount`	CountStatistic	Total number of collections that have occurred.
`collectiontime`	CountStatistic	Accumulated collection time (in milliseconds).

The statistics available for memory in the JVM in J2SE 5.0 are shown in Table 21-20.

Table 21-20 JVM Statistics for J2SE 5.0 - Memory

Statistic	Data Type	Description
`objectpendingfinalizationcount`	CountStatistic	Approximate number of objects that are pending finalization.
`initheapsize`	CountStatistic	Size of the heap initially requested by the JVM.
`usedheapsize`	CountStatistic	Size of the heap currently in use.
`maxheapsize`	CountStatistic	Maximum amount of memory (in bytes) that can be used for memory management.
`committedheapsize`	CountStatistic	Amount of memory (in bytes) that is committed for the JVM to use.
`initnonheapsize`	CountStatistic	Size of the non-heap area initially requested by the JVM.
`usednonheapsize`	CountStatistic	Size of the non-heap area currently in use.
`maxnonheapsize`	CountStatistic	Maximum amount of memory (in bytes) that can be used for memory management.
`committednonheapsize`	CountStatistic	Amount of memory (in bytes) that is committed for the JVM to use.

The statistics available for the operating system in the JVM in J2SE 5.0 are shown in Table 21-21.

Table 21-21 JVM Statistics for J2SE 5.0 - Operating System

Statistic	Data Type	Description
`arch`	StringStatistic	Operating system architecture.
`availableprocessors`	CountStatistic	Number of processors available to the JVM.
`name`	StringStatistic	Operating system name.
`version`	StringStatistic	Operating system version.

The statistics available for the runtime in the JVM in J2SE 5.0 are shown in Table 21-22.

Table 21-22 JVM Statistics for J2SE 5.0 - Runtime

Statistic	Data Type	Description
`name`	StringStatistic	Name representing the running JVM
`vmname`	StringStatistic	JVM implementation name.
`vmvendor`	StringStatistic	JVM implementation vendor.
`vmversion`	StringStatistic	JVM implementation version.
`specname`	StringStatistic	JVM specification name.
`specvendor`	StringStatistic	JVM specification vendor.
`specversion`	StringStatistic	JVM specification version.
`managementspecversion`	StringStatistic	Management spec. version implemented by the JVM.
`classpath`	StringStatistic	Classpath that is used by the system class loader to search for class files.
`librarypath`	StringStatistic	Java library path.
`bootclasspath`	StringStatistic	Classpath that is used by the bootstrap class loader to search for class files.
`inputarguments`	StringStatistic	Input arguments passed to the JVM. Does not include the arguments to the `main` method.
`uptime`	CountStatistic	Uptime of the JVM (in milliseconds).

The statistics available for `ThreadInfo` in the JVM in J2SE 5.0 are shown in Table 21-23.

Table 21-23 JVM Statistics for J2SE 5.0 - ThreadInfo

Statistic	Data Type	Description
`threadid`	CountStatistic	Id of the thread.
`threadname`	StringStatistic	Name of the thread.
`threadstate`	StringStatistic	State of the thread.
`blockedtime`	CountStatistic	Time elapsed (in milliseconds) since the thread entered the `BLOCKED` state. Returns -1 if thread contention monitoring is disabled.
`blockedcount`	CountStatistic	Total number of times that the thread entered the `BLOCKED` state.
`waitedtime`	CountStatistic	Elapsed time (in milliseconds) that the thread has been in a `WAITING` state. Returns -1 if thread contention monitoring is disabled.
`waitedcount`	CountStatistic	Total number of times the thread was in `WAITING` or `TIMED_WAITING` states.
`lockname`	StringStatistic	String representation of the monitor lock that the thread is blocked to enter or waiting to be notified through the `Object.wait` method.
`lockownerid`	CountStatistic	Id of the thread that holds the monitor lock of an object on which this thread is blocking.
`lockownername`	StringStatistic	Name of the thread that holds the monitor lock of the object this thread is blocking on.
`stacktrace`	StringStatistic	Stack trace associated with this thread.

The statistics available for threads in the JVM in J2SE 5.0 are shown in Table 21-24.

Table 21-24 JVM Statistics for J2SE 5.0 - Threads

Statistic	Data Type	Description
`threadcount`	CountStatistic	Current number of live daemon and non-daemon threads.

Table 21-24 JVM Statistics for J2SE 5.0 - Threads *(Continued)*

Statistic	Data Type	Description
`peakthreadcount`	CountStatistic	Peak live thread count since the JVM started or the peak was reset.
`totalstartedthreadcount`	CountStatistic	Total number of threads created and/or started since the JVM started.
`daemonthreadcount`	CountStatistic	Current number of live daemon threads.
`allthreadids`	StringStatistic	List of all live thread ids.
`currentthreadcputime`	CountStatistic	CPU time for the current thread (in nanoseconds) if CPU time measurement is enabled. If CPU time measurement is disabled, returns -1.
`monitordeadlockedthreads`	StringStatistic	List of thread ids that are monitor deadlocked.

Production Web Container (PWC) Statistics

Statistics are available for the following PWC components and services on the Enterprise Edition (EE) of the Application Server:

- Table 21-25, PWC Virtual Server
- Table 21-26, PWC Request
- Table 21-27, PWC File Cache
- Table 21-28, PWC Keep Alive
- Table 21-29, PWC DNS
- Table 21-30, PWC Thread Pool
- Table 21-31, PWC Connection Queue
- Table 21-32, PWC HTTP Service

Statistics for PWC virtual servers are listed in Table 21-25.

Table 21-25 PWC Virtual Server Statistics (EE only)

Attribute Name	Data Type	Description
`id`	String Statistic	The ID of the virtual server.

Table 21-25 PWC Virtual Server Statistics (EE only) *(Continued)*

Attribute Name	Data Type	Description
`mode`	String Statistic	The mode the virtual server is in. Options include `unknown` or `active`.
`hosts`	String Statistic	Name of the hosts serviced by this virtual server.
`interfaces`	String Statistic	Type of interfaces (listeners) for which the virtual server is configured.

The statistics available for PWC requests are listed in Table 21-26.

Table 21-26 PWC Request Statistics (EE only)

Attribute Name	Datatype	Description
`method`	String Statistic	Method used for request.
`uri`	String Statistic	Last URI served.
`countrequests`	Count Statistic	Number of requests served.
`countbytestransmitted`	Count Statistic	Number of bytes transmitted, or 0 if this information is not available
`countbytesreceived`	Count Statistic	Number of bytes received, or 0 if this information is not available.
`ratebytesreceived`	Count Statistic	Rate at which data was transmitted over some server-defined interval, or 0 if this information is not available
`maxbytestransmissionrate`	Count Statistic	Maximum rate at which data was transmitted over some server-defined interval, or 0 if this information is not available.
`countopenconnections`	Count Statistic	Number of connections that are currently open, or 0 if this information is not available.
`maxopenconnections`	Count Statistic	Maximum number of concurrently open connections, or 0 if this information is not available.
`count2xx`	Count Statistic	Total number of responses of code 2XX.
`count3xx`	Count Statistic	Total number of responses of code 3XX.

Table 21-26 PWC Request Statistics (EE only) *(Continued)*

Attribute Name	Datatype	Description
count4xx	Count Statistic	Total number of responses of code 4XX.
count5xx	Count Statistic	Total number of responses of code 5XX.
countother	Count Statistic	Total number of responses with other response codes.
count200	Count Statistic	Total number of responses of code 200.
count302	Count Statistic	Total number of responses of code 302.
count304	Count Statistic	Total number of responses of code 304.
count400	Count Statistic	Total number of responses of code 400.
count401	Count Statistic	Total number of responses of code 401.
count403	Count Statistic	Total number of responses of code 403.
count404	Count Statistic	Total number of responses of code 404.
count503	Count Statistic	Total number of responses of code 503.

The cache information section provides information on how the file cache is being used. Statistics for PWC file caches are listed in Table 21-27.

Table 21-27 PWC File Cache Statistics (EE only)

Attribute Name	Data Type	Description
flagenabled	Count Statistic	Indicates whether the file cache is enabled. Valid values are 0 for no or 1 for yes.
secondsmaxage	Count Statistic	Maximum age of a valid cache entry, in seconds.
countentries	Count Statistic	Number of current cache entries. A single cache entry represents a single URI.
maxentries	Count Statistic	Maximum number of simultaneous cache entries.

Table 21-27 PWC File Cache Statistics (EE only) *(Continued)*

Attribute Name	Data Type	Description
countopenentries	Count Statistic	Number of entries associated with an open file.
maxopenentries	Count Statistic	Maximum number of simultaneous cache entries associated with an open file.
sizeheapcache	Count Statistic	Heap space used for cache content.
maxheapcachesize	Count Statistic	Maximum heap space used for cache file content.
sizemmapcache	Count Statistic	Amount of address space used by memory mapped file content.
maxmmapcachesize	Count Statistic	Maximum amount of address space used by the file cache for memory mapped file content.
counthits	Count Statistic	Number of successful cache lookups.
countmisses	Count Statistic	Number of failed cache lookups.
countinfohits	Count Statistic	Number of times a file information lookup succeeded.
countinfomisses	Count Statistic	Number of misses on cached file information.
countcontenthits	Count Statistic	Number of hits on cached file content.
countcontentmisses	Count Statistic	Number of times a file information lookup failed.

This section provides information about the server's HTTP-level keep-alive system. The statistics available for PWC Keep Alive are listed in Table 21-28.

Table 21-28 PWC Keep Alive Statistics (EE only)

Attribute Name	Datatype	Description
countconnections	Count Statistic	Number of connections in keep-alive mode.
maxconnections	Count Statistic	Maximum number of connections simultaneously allowed in keep-alive mode.

Table 21-28 PWC Keep Alive Statistics (EE only) *(Continued)*

Attribute Name	Datatype	Description
counthits	Count Statistic	The total number of times connections in keep-alive mode have subsequently made a valid request.
countflushes	Count Statistic	The number of times keep-alive connections have been closed by the server.
countrefusals	Count Statistic	The number of times the server could not hand off the connection to a keep-alive thread, possibly due to too many persistent connections.
counttimeouts	Count Statistic	The number of times the server terminated keep-alive connections as the client connections timed out without any activity.
secondstimeout	Count Statistic	The time (in seconds) before idle keep-alive connections are closed.

The DNS Cache caches IP addresses and DNS names. The server's DNS cache is disabled by default. A single cache entry represents a singular IP address or DNS name lookup. The statistics available for PWC DNS are listed in Table 21-29.

Table 21-29 PWC DNS Statistics (EE only)

Attribute Name	Datatype	Description
flagcacheenabled	Count Statistic	Indicates whether the DNS cache is enabled (on). Either 0 for off or 1 for on.
countcacheentries	Count Statistic	Number of DNS entries presently in the cache.
maxcacheentries	Count Statistic	Maximum number of DNS entries that can be accommodated by the cache.
countcachehits	Count Statistic	Number of times a DNS cache lookup has succeeded.
countcachemisses	Count Statistic	Number of times a DNS cache lookup has failed.
flagasyncenabled	Count Statistic	Indicates whether asynchronous DNS lookups are enabled (on). Either 0 for off or 1 for on.
countasyncnamelookups	Count Statistic	Total number of asynchronous DNS name lookups.
countasyncaddrlookups	Count Statistic	Total number of asynchronous DNS address lookups.

Table 21-29 PWC DNS Statistics (EE only) *(Continued)*

Attribute Name	Datatype	Description
countasynclookupsinprogress	Count Statistic	Number of asynchronous lookups in progress.

Statistics for PWC thread pools are listed in Table 21-30.

Table 21-30 PWC Thread Pool Statistics (EE only)

Attribute Name	Data Type	Description
id	String Statistic	ID of the thread pool.
countthreadsidle	Count Statistic	Number of request-processing threads currently idle.
countthreads	Count Statistic	Current number of request-processing threads.
maxthreads	Count Statistic	Maximum number of request processing threads that can exist concurrently.
countqueued	Count Statistic	Number of requests queued for processing by this thread pool.
peakqueued	Count Statistic	The largest number of requests in the queue simultaneously.
maxqueued	Count Statistic	Maximum number of requests that can be in the queue at one time.

The Connection Queue is the queue in which requests are held prior to being serviced. Statistics for the connection queue show the number of sessions in the queue and the average delay before the connection is accepted. Statistics for PWC connection queues are listed in Table 21-31.

Table 21-31 PWC Connection Queue Statistics (EE only)

Attribute Name	Data Type	Description
id	String Statistic	ID of the connection queue.
counttotalconnections	Count Statistic	Total number of connections that have been accepted.

Table 21-31 PWC Connection Queue Statistics (EE only) *(Continued)*

Attribute Name	Data Type	Description
countqueued	Count Statistic	Number of connections currently in the queue.
peakqueued	Count Statistic	Largest number of connections that were in the queue simultaneously.
maxqueued	Count Statistic	Maximum size of the connection queue.
countoverflows	Count Statistic	The number of times the queue has been too full to accommodate a connection.
counttotalqueued	Count Statistic	The total number of connections that have been queued. A given connection may be queued multiple times, so counttotalqueued may be greater than or equal to counttotalconnections.
tickstotalqueued	Count Statistic	The total number of ticks that connections have spent in the queue. A tick is a system-dependent unit of time.
countqueued1minuteaverage	Count Statistic	Average number of connections queued in the last 1 minute.
countqueued5minuteaverage	Count Statistic	Average number of connections queued in the last 5 minutes.
countqueued15minuteaverage	Count Statistic	Average number of connections queued in the last 15 minutes.

Statistics for PWC HTTP service are listed in Table 21-32.

Table 21-32 PWC HTTP Service Statistics (EE only)

Attribute Name	Data Type	Description
id	String Statistic	Instance name of the HTTP service.
versionserver	String Statistic	Version number of the HTTP service.
timestarted	String Statistic	Time the HTTP service was started (GMT).
secondsrunning	Count Statistic	Time (in seconds) since the HTTP service started.

Table 21-32 PWC HTTP Service Statistics (EE only) *(Continued)*

Attribute Name	Data Type	Description
`maxthreads`	Count Statistic	Maximum number of worker threads in each instance.
`maxvirtualservers`	Count Statistic	Maximum number of virtual servers that can be configured in each instance.
`flagprofilingenabled`	Count Statistic	Whether or not HTTP service performance profiling is enabled. Valid values are 0 or 1.
`flagvirtualserveroverflow`	Count Statistic	Indicates whether more than `maxvirtualservers` are configured. If this is set to 1, statistics are not being tracked for all virtual servers.
`load1minuteaverage`	Count Statistic	Average load for requests in the last 1 minute.
`load5minuteaverage`	Count Statistic	Average load for requests in the last 5 minutes.
`load15minuteaverage`	Count Statistic	Average load for requests in the last 15 minutes.
`ratebytestransmitted`	Count Statistic	The rate at which data is transmitted over some server-defined interval. The result is 0 when this information is not available.
`ratebytesreceived`	Count Statistic	The rate at which data is received over some server-defined interval. The result is 0 when this information is not available.

Admin Console Tasks for Enabling and Disabling Monitoring

Configuring Monitoring Levels Using the Admin Console

1. Access the Monitoring Service page. To do this,

a. In the tree, expand the Configurations node.

b. In the tree, expand the node for the server instance to be configured for monitoring, for example, `server-config`.

c. In the tree, select Monitoring.

2. On the Monitoring Service page, choose the appropriate value from the combo box opposite the component(s) or service(s) whose monitoring level is changing.

   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. It is possible to turn monitoring on or off for the following components and services:

   - **JVM** - Set the monitoring level to LOW for this option to monitor the Java Virtual Machine.
   - **HTTP Service** - Set the monitoring level to LOW for this option to monitor all HTTP listeners and virtual servers.
   - **Transaction Service** - Set the monitoring level to LOW for this option to monitor any transaction subsystem.
   - **JMS/Connector Service** - Set the monitoring level to LOW for this option to monitor any Java Message Service (JMS).
   - **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.
   - **Web Container** - Set the monitoring level to LOW for this option to monitor all deployed servlets.
   - **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.
   - **JDBC Connection Pool** - Set the monitoring level to LOW for this option to monitor all JDBC connection pools.
   - **Thread Pool** - Set the monitoring level to LOW for this option to monitor all thread pools.

3. Click Save.

There are no Additional Monitoring Service Properties in this release, therefore ignore the Additional Properties table.

Equivalent `asadmin` command: `set`, for example, to turn on monitoring for the HTTP Service, use the following `asadmin` command:

```
asadmin> set --user admin_user
server.monitoring-service.module-monitoring-levels.http-service=LOW
```

Configuring Monitoring Using the asadmin Tool

To turn monitoring off, or to set a level for monitoring a component or service, you can use the Admin Console as described in "Configuring Monitoring Levels Using the Admin Console", or use the `asadmin` tool as described in this section.

1. Use the `get` command to find out what services and components currently have monitoring enabled:

   ```
   asadmin> get --user admin_user
   server.monitoring-service.module-monitoring-levels.*
   ```

   Returns:

   ```
   server.monitoring-service.module-monitoring-levels.
   connector-connection-pool = OFF
   server.monitoring-service.module-monitoring-levels.
   connector-service = OFF
   server.monitoring-service.module-monitoring-levels.ejb-container = OFF
   server.monitoring-service.module-monitoring-levels.http-service = OFF
   server.monitoring-service.module-monitoring-levels.jdbc-connection-pool
   = OFF
   server.monitoring-service.module-monitoring-levels.jms-service = OFF
   server.monitoring-service.module-monitoring-levels.jvm = OFF
   server.monitoring-service.module-monitoring-levels.orb = OFF
   server.monitoring-service.module-monitoring-levels.thread-pool = OFF
   server.monitoring-service.module-monitoring-levels.transaction-service
   = OFF
   server.monitoring-service.module-monitoring-levels.web-container = OFF
   ```

2. Use the `set` command to enable monitoring.

   For example, to enable monitoring for the HTTP service:

```
asadmin> set --user admin_user
server.monitoring-service.module-monitoring-levels.http-service=LOW
```

To disable monitoring, use the `set` command and specify `OFF` for the monitoring level.

Admin Console Tasks for Viewing Monitoring Data

Viewing Monitoring Data in the Admin Console

To view monitoring data for a component or service deployed in a server instance using the Application Server Admin Console, follow these steps. For more description on the attributes for each component or service, refer to "About Statistics for Monitored Components and Services".

1. Access the Monitoring page. To do this:
   a. In the tree component, expand the Standalone Instances node, for example `server (Admin Server)`.
   b. Select a stand-alone server instance from the list.
   c. Select the Monitor page.
   d. Select the Monitor tab on the Monitor page.
2. From the View list, select a component or service that has been deployed onto the server instance and for which monitoring is enabled.

   Monitoring data for the selected component or service displays below the View field. For a description of the monitorable properties, refer to "About Statistics for Monitored Components and Services".

   On this page, it is possible to view monitoring data for JVM, Server, Thread Pools, HTTP Service, and Transaction Service if monitoring is enabled for these components and services. A diagram showing how these components and services are organized is shown in "About the Tree Structure of Monitorable Objects".

3. If you do not see the component or service you wish to monitor in this list, select the Configure Monitoring link to enable and disable monitoring for selected components and services. Select OFF to disable monitoring for a component or service. Select LOW or HIGH to enable monitoring for a component or service.

   For more information on enabling and disabling monitoring, refer to "Configuring Monitoring Levels Using the Admin Console" or "Configuring Monitoring Using the asadmin Tool".

4. Select the *Applications* tab of the Monitor page to view monitoring data for application components that are deployed onto the server instance and for which monitoring is enabled. Select the application from the Application list. Select the specific component from the Component list.

   If no monitoring data appears for the application or component, select the Configure Monitoring link to enable or disable monitoring for a component or service. To monitor applications, turn on the container in which they execute: for example, select Low or High for the Web Container for web applications and/or the EJB Container for EJB applications.

   If no monitoring data is displayed for applications, it is most likely that the application does not exist or is not exercised. Applications monitoring data is available only when the application exists, monitoring is enabled for the application, and the application is being exercised. Once the application is executed, it is registered in the monitoring registry and the monitoring data displays.

   Use the Admin Console to monitor remote applications and instances. For this to occur, the remote instance must be running and the configuration set.

   Monitoring data for the selected component displays below the selected component. For a description of the monitorable properties, refer to "About Statistics for Monitored Components and Services". A diagram showing how these components and services are organized for applications can be viewed in "About the Tree Structure of Monitorable Objects".

5. Select the *Resources* page to view monitoring data for resources that are deployed onto the server instance and for which monitoring has been enabled. Select the resource from the View list. If the resource for which you wish to view monitoring data does not appear, select the Configure Monitoring link to enable or disable monitoring for a resource.

   If no monitoring data is displayed for resources, it is most likely that the resource does not exist or is not exercised. Resources monitoring data is available only when the resources exist, monitoring is enabled for the resource at a level of HIGH, and the resource is being exercised. For example, if you

have created a JDBC connector service, but applications that use that connector service have not yet requested a connector from the service, that service has not yet been created, therefore, no service exists and no monitoring data is available. Once the JDBC application is executed and requests a connector from a service, the service is registered in the monitoring registry and monitoring data appears.

Monitoring data for the selected component or service displays below the View field. For a description of the monitorable properties, refer to "About Statistics for Monitored Components and Services". A diagram showing how these components and services are organized for resources can be viewed in "About the Tree Structure of Monitorable Objects".

6. Select the *Transactions* page 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. To enable monitoring for the Transaction Service, select the Configure Monitoring link and make sure that Transaction Service is set to LOW. To freeze the Transaction Service in order to roll back transactions, select Freeze. To rollback a transaction, select the checkbox beside the transaction and click Rollback.

Equivalent `asadmin` command: `get --monitor`, for example, to view monitoring data for the JVM, use the following `asadmin` command:

```
asadmin> get --monitor server.jvm.*
```

Viewing Monitoring Data With the asadmin Tool

Using the asadmin Tool to View Monitoring Data

To view monitoring data using the `asadmin` tool, use the `asadmin list` and `asadmin get` commands followed by the dotted name of a monitorable object. As a general approach to using the `asadmin` tool to view monitoring data, follow these steps:

1. To view the names of the objects that can be monitored, use the `asadmin list` command. For example, to view a list of application components and subsystems that have monitoring enable for the server instance, type the following command in a terminal window:

   ```
   asadmin> list --monitor server
   ```

   The preceding command returns a list of application components and subsystems that have monitoring enabled, for example:

   ```
   server.resources
   server.connector-service
   server.orb
   server.jms-service
   server.jvm
   server.applications
   server.http-service
   server.thread-pools
   ```

   Sun Java System Application Server Enterprise Edition 8.1 2005Q1For further examples using the `list` command, refer to "Examples of the list and get Commands". For further information on the dotted names you can use with the `list` command, refer to "Understanding and Specifying Dotted Names".

2. To display monitoring statistics for an application component or subsystem for which monitoring has been enabled, use the `asadmin get` command. To get the statistics, type the `asadmin get` command in a terminal window, specifying a name displayed by the `list` command in the preceding step. The following example attempts to get all attributes from a subsystem for a specific object:

   ```
   asadmin> get --monitor server.jvm.*
   ```

The command returns the following attributes and data:

```
server.jvm.dotted-name = server.jvm
server.jvm.heapsize-current = 21241856
server.jvm.heapsize-description = Provides statistical information
about the JVM's memory heap size.
server.jvm.heapsize-highwatermark = 21241856
server.jvm.heapsize-lastsampletime = 1080232913938
server.jvm.heapsize-lowerbound = 0
server.jvm.heapsize-lowwatermark = 0
server.jvm.heapsize-name = JvmHeapSize
server.jvm.heapsize-starttime = 1080234457308
server.jvm.heapsize-unit = bytes
server.jvm.heapsize-upperbound = 518979584
server.jvm.uptime-count = 1080234457308
server.jvm.uptime-description = Provides the amount of time the JVM has
```

```
been running.
server.jvm.uptime-lastsampletime = 1080234457308
server.jvm.uptime-name = JvmUpTime
server.jvm.uptime-starttime = 1080232913928
server.jvm.uptime-unit = milliseconds
```

Sun Java System Application Server Enterprise Edition 8.1 2005Q1For further examples using the `get` command, refer to "Examples of the list and get Commands". For further information on the dotted names you can use with the `get` command, refer to "Understanding and Specifying Dotted Names".

Understanding and Specifying Dotted Names

In the `asadmin list` and `get` commands, specify the dotted name of monitorable objects. All child objects are addressed using the dot (.) character as separator, thus these are referred to as *dotted names*. If a child node is of singleton type, then only the monitoring object type is needed to address the object, otherwise a name of the form `type.name` is needed to address the object.

For example, `http-service` is one of the valid monitorable object types and is a singleton. To address a singleton child node representing the `http-service` of instance `server`, the dotted name is:

```
server.http-service
```

Another example, `application`, is a valid monitorable object type and is not a singleton. To address a non-singleton child node representing, for example, the application PetStore, the dotted name is:

```
server.applications.petstore
```

The dotted names can also address specific attributes in monitorable objects. For example, `http-service` has a monitorable attribute called `bytesreceived-lastsampletime`. The following name addresses the `bytesreceived` attribute:

```
server.http-service.server.http-listener-1.
  bytesreceived-lastsampletime
```

The administrator is not expected to know the valid dotted names for `asadmin list` and `get` commands. The `list` command displays available monitorable objects, while the `get` command used with a wildcard parameter allows the inspection of all available attributes on any monitorable object.

The underlying assumptions for using the `list` and `get` commands with dotted names are:

- Any `list` command that has a dotted name that is **not** followed by a wildcard (*) gets as its result the current node's immediate children. For example, `list --monitor server` lists all immediate children belonging to the `server` node.
- Any `list` command that has a dotted name followed by a wildcard of the form `.*` gets as its result a hierarchical tree of children nodes from the current node. For example, `list --monitor server.applications.*` lists all children of `applications` and their subsequent child nodes and so on.
- Any `list` command that has a dotted name preceded or followed by a wildcard of the form **dottedname* or *dotted * name* or **`dotted name *`** gets as its result all nodes and their children matching the regular expression created by the provided matching pattern.
- A `get` command followed by a `.*` or a `*` gets as its result the set of attributes and their values belonging to the current node to be matched.

For more information, read "Expected Output for list and get Commands at All Levels".

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".

Example 1

```
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.jms-service
```

```
server.connector-service
server.applications
server.http-service
server.thread-pools
```

It is also possible to list applications that are currently monitored in the specified server instance. This is useful when particular monitoring statistics are sought from an application using the get command.

Example 2

```
asadmin> list --monitor server.applications
```

Returns:

```
server.applications.adminapp
server.applications.admingui
server.applications.myApp
```

For a more comprehensive example, see "Petstore Example".

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" for more information on the use of the get command.

Example 1

Attempt to get all attributes from a subsystem for a specific object:

```
asadmin> get --monitor server.jvm.*
```

Returns:

```
server.jvm.dotted-name= server.jvm
server.jvm.heapsize-current = 21241856
server.jvm.heapsize-description = Provides statistical information
about the JVM's memory heap size.
server.jvm.heapsize-highwatermark = 21241856
server.jvm.heapsize-lastsampletime = 1080232913938
server.jvm.heapsize-lowerbound = 0
```

```
server.jvm.heapsize-lowwatermark = 0
server.jvm.heapsize-name = JvmHeapSize
server.jvm.heapsize-starttime = 1080234457308
server.jvm.heapsize-unit = bytes
server.jvm.heapsize-upperbound = 518979584
server.jvm.uptime-count = 1080234457308
server.jvm.uptime-description = Provides the amount of time the JVM has
been running.
server.jvm.uptime-lastsampletime = 1080234457308
server.jvm.uptime-name = JvmUpTime
server.jvm.uptime-starttime = 1080232913928
server.jvm.uptime-unit = milliseconds
```

Example 2

Attempt to get all attributes from a J2EE application:

```
asadmin> get --monitor server.applications.myJ2eeApp.*
```

Returns:

```
No matches resulted from the wildcard expression.
CLI137 Command get failed.
```

There are no monitorable attributes exposed at the J2EE-application level, therefore this reply displays.

Example 3

Attempt to get a specific attribute from a subsystem:

```
asadmin> get --monitor server.jvm.uptime-lastsampletime
```

Returns:

```
server.jvm.uptime-lastsampletime = 1093215374813
```

Example 4

Attempt to get an unknown attribute from within a subsystem attribute:

```
asadmin> get --monitor server.jvm.badname
```

Returns:

```
No such attribute found from reflecting the corresponding Stats
interface: [badname]
CLI137 Command get failed.
```

Petstore Example

The following example illustrates how the `asadmin` tool might be used for monitoring purposes.

A user wants to inspect the number of calls made to a method in the sample Petstore application after it has been deployed onto the Application Server. The instance onto which it has been deployed is named `server`. A combination of the `list` and `get` commands are used to access desired statistics on a method.

1. Start the Application Server and the `asadmin` tool.
2. Set some useful environment variables to avoid entering them for every command:

   ```
   asadmin>export AS_ADMIN_USER=admin AS_ADMIN_PASSWORD=admin123
   ```

   ```
   asadmin>export AS_ADMIN_HOST=localhost AS_ADMIN_PORT=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. To also get a specific statistic, such as execution time, use a command such as the following:

```
asadmin>get -m server.applications.petstore.signon-ejb_jar.
UserEJB.bean-methods.getUserName.executiontime-count
```

Returns:

```
server.applications.petstore.signon-ejb_jar.UserEJB.bean-methods.
getUserName.executiontime-count = 1
```

Expected Output for list and get Commands at All Levels

The following tables show the command, dotted name, and corresponding output at each level of the tree.

Table 21-33 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 except a message saying there are no attributes at this node.

Table 21-34 shows the command, dotted name, and corresponding output for the applications level.

Table 21-34 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 except message saying there are no attributes at this node.

Table 21-35 shows the command, dotted name, and corresponding output for stand-alone modules and enterprise applications at the applications level.

Table 21-35 Applications - Enterprise Applications and Standalone Modules

Command	Dotted Name	Output

`list -m`	`server.applications.app1`
	or
	`*app1`
	Note: this level is only applicable if an enterprise application has been deployed. It is not applicable if a standalone module is deployed.
		`ejb-module1_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 except message saying there are no attributes at this node.

Table 21-35 Applications - Enterprise Applications and Standalone Modules *(Continued)*

Command	Dotted Name	Output

`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 except message saying there are no attributes at this node.

`list -m`	`server.applications.app1.ejb-module1_jar.bean1`
	Note: In standalone modules, the node containing the application name (app1 in this example) will not appear.
		List of child nodes:
		`bean-pool`
		`bean-cache`
		`bean-method`

`list -m`	`server.applications.app1.ejb-module1_jar.bean1` *Note: In standalone modules, the node containing the application name (app1 in this example) will not appear.*	Hierarchy of child nodes and a list of all attributes for this node and for any subsequent child nodes.

Table 21-35 Applications - Enterprise Applications and Standalone Modules *(Continued)*

Command	Dotted Name	Output

`get -m`	`server.applications.app1.ejb-module1_jar.bean1.*`
	Note: In standalone modules, the node containing the application name (app1 in this example) does not appear.
		The following attributes and their associated values:
		`CreateCount_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 1-4.
`list -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-cache` *Note: In standalone modules, the node containing the application name (app1 in this example) will not appear.*	No attributes, but a message saying "Use get command with the --monitor option to view this node's attributes and values."
`get -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-cache.*` *Note: In standalone modules, the node containing the application name (app1 in this example) does not appear.*	List of attributes and values corresponding to EJB Cache attributes as described in Table 1-5.

Table 21-35 Applications - Enterprise Applications and Standalone Modules *(Continued)*

Command	Dotted Name	Output
`list -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-method.method1` *Note: In standalone modules, the node containing the application name (app1 in this example) will not appear.*	No attributes, but a message saying "Use get command with the --monitor option to view this node's attributes and values."
`get -m`	`server.applications.app1.ejb-module1_jar.bean1.bean-method.method1.*` *Note: In standalone modules, the node containing the application name (app1 in this example) will not appear.*	List of attributes and values corresponding to EJB Methods attributes as described in Table 1-2.
`list -m`	`server.applications.app1.web-module1_war`	Displays the virtual server(s) assigned to the module.
`get -m`	`server.applications.app1.web-module1_war.*`	No output except a message saying there are no attributes at this node.
`list -m`	`server.applications.app1.web-module1_war.virtual_server`	Displays list of servlets registered.
`get -m`	`server.applications.app1.web-module1_war.virtual_server.*`	No output except a message saying there are no attributes at this node.
`list -m`	`server.applications.app1.web-module1_war.virtual_server.servlet1`	No attributes, but a message saying "Use get command with the --monitor option to view this node's attributes and values."
`get -m`	`server.applications.app1.web-module1_war.virtual_server.servlet1.*`	List of attributes and values corresponding to web container (Servlet) attributes as described in Table 1-7.

Table 21-36 shows the command, dotted name, and corresponding output for the HTTP Service level.

Table 21-36 HTTP-Service Level

Command	Dotted Name	Output
`list -m`	`server.http-service`	List of virtual servers.
`get -m`	`server.http-service.*`	No output except message saying there are no attributes at this node.
`list -m`	`server.http-service.server`	List of HTTP Listeners.
`get -m`	`server.http-service.server.*`	No output except message saying there are no attributes at this node.

Table 21-36 HTTP-Service Level *(Continued)*

Command	Dotted Name	Output
`list -m`	`server.http-service.server.http-listener1`	No attributes, but a message saying "Use get command with the --monitor option to view this node's attributes and values."
`get -m`	`server.http-service.server.*`	List of attributes and values corresponding to HTTP Service attributes as described in Table 1-9.

Table 21-37 shows the command, dotted name, and corresponding output for the thread pools level.

Table 21-37 Thread-Pools Level

Command	Dotted Name	Output
`list -m`	`server.thread-pools`	List of thread-pool names.
`get -m`	`server.thread-pools.*`	No output except message saying there are no attributes at this node.
`list -m`	`server.thread-pools.orb\.threadpool\.thread-pool-1`	No attributes, but a message saying "Use get command with the --monitor option to view this node's attributes and values."
`get -m`	`server.thread-pools..orb\.threadpool\.thread-pool-1.*`	List of attributes and values corresponding to Thread Pool attributes as described in Table 1-14.

Table 21-38 shows the command, dotted name, and corresponding output for the resources level.

Table 21-38 Resources Level

Command	Dotted Name	Output
`list -m`	`server.resources`	List of pool names.
`get -m`	`server.resources.*`	No output except message saying there are no attributes at this node.
`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."

Table 21-38 Resources Level *(Continued)*

Command	Dotted Name	Output
`get -m`	`server.resources.jdbc-connection-pool-pool.connection-pool1.*`	List of attributes and values corresponding to Connection Pool attributes as described in Table 1-10.

Table 21-39 shows the command, dotted name, and corresponding output for the transaction service level.

Table 21-39 Transaction-Service Level

Command	Dotted Name	Output
`list -m`	`server.transaction-service`	No attributes, but a message saying "Use get command with the --monitor option to view this node's attributes and values."
`get -m`	`server.transaction-service.*`	List of attributes and values corresponding to Transaction Service attributes as described in Table 1-15.

Table 21-40 shows the command, dotted name, and corresponding output for the ORB level.

Table 21-40 ORB Level

Command	Dotted Name	Output
`list -m`	`server.orb`	`server-orb.connection-managers`
`get -m`	`server.orb.*`	No output except message saying there are no attributes at this node.
`list -m`	`server.orb.connection-managers`	Name(s) of ORB connection managers.
`get -m`	`server.orb.connection-managers.*`	No output except message saying there are no attributes at this node.
`list -m`	`server.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnections`	No attributes, but a message saying "Use get command with the --monitor option to view this node's attributes and values."

Table 21-40 ORB Level *(Continued)*

Command	Dotted Name	Output
`get -m`	`server.orb.connection-managers.orb\.Connections\.Inbound\.AcceptedConnections.*`	List of attributes and values corresponding to ORB Connection Manager attributes as described in Table 1-13.

Table 21-34 shows the command, dotted name, and corresponding output for the JVM level.

Table 21-41 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 1-16.

Using JConsole

For JConsole to work with the Application Server, security has to be disabled for the JMX connector. The current version of the Application Server (SE/EE edition) has security enabled by default.

To disable security for the JMX Connector, use one of these techniques:

1. Use the Admin Console to disable security for the JMX Connector. To do this from the Admin Console,
   a. Expand the Configurations node.
   b. Expand the Admin Service node.
   c. Select the `system` node.
   d. In the SSL section, unselect SSL3 and TLS.
   e. Select Save.
2. Use `asadmin` to disable security for the JMX Connector. To do this from a terminal window or command prompt,

a. Enter this command:

```
asadmin set
server.admin-service.jmx-connector.system.security-enabled=false
```

b. Restart the domain application server (DAS).

For the PE version, the JMX Connector is disabled by default, therefore there is no need to change any configuration for PE.

3. Start JConsole and enter the JMX URL, user name, and password on the Advanced tab for logging in. The JMX URL is of the form:

```
service:jmx:rmi:///jndi/rmi://<your machine
name>:<port>/management/rmi-jmx-connector
```

NOTE: You can get the exact JMX URL from the admin `server.log` file if you search for `message ADM1501`.

Chapter 22

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 Application Server 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 Application Server 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 Application Server 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 Advanced Domain Attributes

1. In the tree component, select the Application Server node.

2. Select the Advanced tab.

3. On the Domain Attributes page, you may do the following:
   a. In the Application Root field, identify the full directory path where the applications will be deployed.
   b. In the Log Root field, specify where the server instance log files are kept.
   c. Typically, you will leave the Locale field blank to 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.

Appendix A

Compiling and Configuring Apache Web Server

This appendix describes how to compile the Apache source code and configure an installation of Apache Web Server to use the Sun Java System Application Server load balancer plug-in.

Download the appropriate Apache source code. For information about the versions and platforms of Apache Web Server supported for Sun Java System Application Server, see the *Sun Java System Application Server Release Notes*.

This appendix contains the following topics:

Minimum Requirements

This section describes the minimum requirements to successfully compile Apache web server to run the load balancer plug-in. The Apache source must be compiled and built to run with SSL.

This section contains the following topics:

Minimum Requirements for Apache 1.3

For requirements on Microsoft Windows platforms, see:
http://httpd.apache.org/docs/windows.html#req
http://httpd.apache.org/docs/win_compiling.html

Requirements for other platforms:

- openssl-0.9.7d (source)
- mod_ssl-2.8.16-1.3.29 (source)
- apache_1.3.29 (sources)
- gcc-3.3-sol9-sparc-local packages (for Solaris 9 SPARC / x86).
- flex-2.5.4a-sol9-sparc-local packages (for Solaris 9 SPARC)
- flex-2.5.4a-sol9-intel-local packages (for Solaris 9 x86)

In addition, before compiling Apache:

- On Linux, install Sun Java System Application Server on the same machine.
- On Solaris 8, ensure that gcc and make are in the PATH.
- On Solaris 9, ensure that gcc version 3.3 and make are in the PATH, and flex is installed.
- If you are using gcc on Red Hat Enterprise Linux Advanced Server 2.1, the version must be later than gcc 3.0.

NOTE

- To use another C compiler, set the path of the C compiler and make utility in the PATH environment variable. For example:

  exportLD_LIBRARY_PATH=$LD_LIBRARY_PATH:appserver_installdir/lib
- These software sources are available at http://www.sunfreeware.com

Minimum Requirements for Apache 2

For requirements on Microsoft Windows platforms, see:

http://httpd.apache.org/docs-2.0/platform/windows.html

Requirements for other platforms:

- openssl-0.9.7d (source)

- `httpd-2.0.49` (source)
- `gcc-3.3-sol9-sparc-local` packages (for Solaris 9 SPARC / x86).
- `flex-2.5.4a-sol9-sparc-local` packages (for Solaris 9 SPARC)
- `flex-2.5.4a-sol9-intel-local` packages (for Solaris 9 x86)

In addition, before compiling Apache:

- On Linux, install Sun Java System Application Server on the same machine.
- On Solaris 8, ensure that `gcc` and `make` are in the `PATH`.
- On Solaris 9, ensure that `gcc` version 3.3 and `make` are in the `PATH`, and `flex` is installed.
- If you are using `gcc` On Red Hat Enterprise Linux Advanced Server 2.1, the version must be later than gcc 3.0.

NOTE

- To use another C compiler, set the path of the C compiler and make utility in the PATH environment variable. For example:

  ```
  export LD_LIBRARY_PATH=app_server_install_dir/lib:$LD_LIBRARY_PATH
  ```

  This example is for `sh`.
- These software sources are available at `http://www.sunfreeware.com`

Installing SSL-aware Apache

For instructions on compiling and installing Apache on Microsoft Windows platforms, see the following web sites:

Apache 1.3:

`http://httpd.apache.org/docs/win_compiling.html`

Apache 2:

`http://httpd.apache.org/docs-2.0/platform/win_compiling.html`

Follow these steps to compile, configure, and install SSL-aware Apache web server on other platforms. Although the examples show compiling and building Apache 1.3.29, the same procedures apply for Apache 2.

NOTE Untar `mod_ssl`, OpenSSL, and Apache at the same directory level.

Compiling and Building OpenSSL

This step is not required on Linux if the version of OpenSSL installed with Linux is 0.9.7d.

For more information on OpenSSL, see:

http://www.openssl.org/

Unpack the `openssl-0.9.7d` source and follow these steps.

1. `cd openssl-0.9.7d`
2. `./config`
3. `make`
4. `make test`
5. `make install`

For more information on building OpenSSL from the source, see the `INSTALL` file in the `openssl` directory.

Configuring Apache with mod_ssl

This section only applies to Apache 1.3. For Apache 2.0 installations, skip to "Compiling and Building Apache" on page 407.

For more information on mod_ssl, see:

http://www.modssl.org/

1. Download the `apache_1.3.29` source distribution.

   Unpack the source distribution. The source distribution comes as a compressed archive. For `apache_1.3.29`, the source distribution archive reads `apache_1.3.29.tar.gz`.

2. Decompress and untar the archive using the following command:

   ```
   tar -zxvf apache_1.3.29.tar.gz
   ```

   This command creates a directory called `apache_1.3.29` in the current working directory.

3. Unpack the `mod_ssl-2.8.14-1.3.29` source.

4. `cd mod_ssl-2.8.14-1.3.29`

5. Run `./configure --with-apache=../apache_1.3.29 --with-ssl=../openssl-0.9.7d --prefix=`*install path* `--enable-module=ssl --enable-shared=ssl --enable-rule=SHARED_CORE --enable-module=so`

The directory specified in the above command examples is a variable. The *prefix* argument indicates where to install Apache. This command outputs several lines on the screen.

This command creates the `make` files for the build according to your system configuration. Errors in `configure` can cause some header files or utility programs to be missing. Install them before proceeding.

Compiling and Building Apache

The instructions for compiling and building Apache vary depending upon the version of Apache.

- Compiling and Building Apache 1.3
- Compiling and Building Apache 2

Compiling and Building Apache 1.3

This procedure installs Apache in the location provided in the `--prefix` attribute described in "Configuring Apache with mod_ssl" on page 406.

1. On Linux, include the following lines in `src/MakeFile` after `End of automatically generated section`:

   ```
   LIBS+= -licuuc -licui18n -lnspr4 -lpthread -lxerces-c -lsupport
   -lnsprwrap -lns-httpd40
   ```

   `LDFLAGS+= -L/`*appserver_installdir*`/lib`

2. On Linux, put the Application Server installation directory in the LD_LIBRARY_PATH:

   `export LD_LIBRARY_PATH=/`*app_server_install_dir*`/lib:$LD_LIBRARY_PATH`

3. Compile Apache using the `make` command, as follows:

   a. cd to the `mod_ssl` directory.

   b. `make`

   c. `make certificate`

   d. `make install`

NOTE The command `make certificate` asks for a secure password. Remember this password as it's required for starting secure Apache.

The command `make install` outputs several lines on the screen indicating that the process is compiling Apache source code and linking Apache. This process normally concludes without errors. However, if any errors occur, check that all the library files and utility programs of Apache have been properly downloaded.

Configure the installation of Apache by entering the appropriate values for your environment in the *apache_install_path*`/conf/httpd.conf` file.

Compiling and Building Apache 2

1. Download the Apache 2_0_*NN* source distribution.

   NN denotes a minor version, for example, 52.

2. Unpack the source distribution.

   The source distribution comes as a compressed archive. For Apache 2_0_*NN*, the source distribution archive is `httpd-2_0_`*NN*`.tar.gz`.

3. Decompress and untar the archive using the following command:

   `tar -zxvf httpd-2_0_`*NN*`.tar.gz`

   This command creates a directory called `httpd-2_0_`*NN* in the current working directory.

4. `cd httpd-2_0_`*NN*.

5. Run `./configure --with-ssl=`*open_ssl_install_path* `--prefix=`*install_path* `--enable-ssl --enable-so`

6. On Linux, modify *apache_src*`/build/config_vars.mk` and add the following lines:

   ```
   EXTRA_LIBS += -licuuc -licui18n -lnspr4 -lpthread -lxerces-c
   -lsupport -lnsprwrap -lns-httpd40

   LDFLAGS+=-L<appserver install dir>/lib
   ```

7. On Linux, put the Application Server installation directory in the LD_LIBRARY_PATH:

   ```
   export LD_LIBRARY_PATH=/app_server_install_dir/lib:$LD_LIBRARY_PATH
   ```

8. Compile Apache using the make command, as follows:

   From the `httpd-2_0_`*NN* directory:

   a. `make`

   b. `make install`

   The command `make install` outputs several lines on the screen indicating that the process is compiling Apache source code and linking Apache. This process normally concludes without errors. However, if any errors occur, check that all the library files and utility programs of Apache have been properly downloaded.

   Configure the installation of Apache by entering the appropriate values for your environment in the *apache_install_path*`/conf/httpd.conf` file.

NOTE If you encounter errors, try putting the Application Server installation directory in the PATH:

```
export LD_LIBRARY_PATH=/app_server_install_dir/lib
```

or adding the OpenSSL library, for example:

```
export
LD_LIBRARY_PATH=/openssl_install_dir/lib:/app_server_install_di
r/lib
```

NOTE On Apache 2, you must create and install the certificate manually. For more information, see the Apache documentation.

Starting and Stopping Apache

Apache comes bundled with a script titled `apachectl` that facilitates starting, stopping and restarting Apache.

- Run the follow command to start Apache:

  apache_install_dir`/bin/apachectl start`

- Run the follow command to start Apache in SSL mode:

  apache_install_dir`/bin/apachectl startssl`

- To stop apache, run the following command:

  apache_install_dir`/bin/apachectl stop`

After starting Apache, test the installation. Once Apache is running, type the following address in a web browser: http://*server_name*:*port_number*/. If the installation was successful and Apache is running, a test page is displayed.

Once you have completed the Apache installation, see "Modifications to Apache Web Server" on page 69 for information on Apache configuration during and after plug-in installation.

Appendix B

Automatically Restarting a Domain or Node Agent

If your domain or node agent is stopped unexpectedly (for example, if you need to restart your machine), you can configure your system to automatically restart the domain or node agent.

This Appendix contains the following topics:

Restarting Automatically on UNIX Platforms

To restart your domain on a UNIX platform, add a line of text to the `/etc/inittab` file.

For example, to restart domain1, for an Application Server installed in the `opt/SUNWappserver` directory, using a password file called `password.txt`:

```
das:3:respawn:/opt/SUNWappserver/bin/asadmin start-domain --user admin
--passwordfile /opt/SUNWappserver/password.txt domain1
```

Put the text on one line. The first three letters are a unique designator for the process and can be altered.

To restart a node agent, the syntax is similar. For example, to restart agent1, for an Application Server installed in the `opt/SUNWappserver` directory, using a password file called `password.txt`:

```
das:3:respawn:/opt/SUNWappserver/bin/asadmin start-node-agent --user admin
--passwordfile /opt/SUNWappserver/password.txt agent1
```

Restarting Automatically on the Microsoft Windows Platform

To restart automatically on Microsoft Windows, create a Windows Service. Use the `appservService.exe` and `appserverAgentService.exe` executables shipped with Sun Java System Application Server in conjunction with the Service Control command (`sc.exe`) provided by Microsoft.

The `sc.exe` command comes with Windows XP and is either located in the `C:\windows\system32` directory or `C:\winnt\system32` directory. As of this writing, the Windows 2000 `sc.exe` is available for download at: `ftp://ftp.microsoft.com/reskit/win2000/sc.zip`. For more information on using `sc.exe`, see `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndllpro/html/msdn_scmslite.asp`.

Use `appservService.exe` and `appservAgentService.exe` as follows:

```
C:\winnt\system32\sc.exe create service_name binPath=
\"fully_qualified_path_to_appservService.exe \"fully_qualified_path_to_asadmin.bat start_command\"
\"fully_qualified_path_to_asadmin.bat stop_command\"" start= auto DisplayName=
"display_name"
```

For example, to create a service called `SunJavaSystemAppServer DOMAIN1` that starts and stops the domain `domain1`, using a password file `C:\Sun\AppServer\password.txt`:

```
C:\windows\system32\sc.exe create domain1 binPath=
"C:\Sun\AppServer\lib\appservService.exe
\"C:\Sun\AppServer\bin\asadmin.bat start-domain --user admin --passwordfile
C:\Sun\AppServer\password.txt domain1\" \"C:\Sun\AppServer\bin\asadmin.bat
stop-domain domain1\"" start= auto DisplayName= "SunJavaSystemAppServer
DOMAIN1"
```

To create a service that starts and stops the node agent `agent1`:

```
C:\windows\system32\sc.exe create agent1 binPath=
"C:\Sun\AppServer\lib\appservAgentService.exe
\"C:\Sun\AppServer\bin\asadmin.bat start-node-agent --user admin
--passwordfile C:\Sun\AppServer\password.txt agent1\"
\"C:\Sun\AppServer\bin\asadmin.bat stop-node-agent agent1\"" start= auto
DisplayName= "SJESAS_SE8.1 AGENT1"
```

NOTE The start and stop commands entered as part of the `binPath=` parameter must have the correct syntax. To test, run the commands from the command prompt. If the commands do not properly start or stop the domain or node agent, the service does not work correctly.

NOTE Don't use a mixture of `asadmin` start and stop commands and service start and stops. Mixing the two can cause the server status to be out of sync. For example, the service might not show that the component has started even though the component is not running. To avoid this situation, always use the `sc.exe` command to start and stop the component when using services.

Security for Automatic Restarts

Handle the password and master password required when starting in one of the following ways:

- On Microsoft Windows, configure the service to ask the user for the password.
  - **a.** In the Services Control Panel, double-click the service you created.
  - **b.** In the Properties window, click the Log On tab.
  - **c.** Check "Allow service to interact with desktop" to prompt for the required passwords when starting the component.

    You have to log in to see the prompts, and entries are not echoed back as you type them. This method is the most secure way to use the services option, but user interaction is required before the service becomes available.

    If the "interact with desktop" option is not set, the service stays in a "start-pending" state and appears to hang. Kill the service process to recover from this state.
- On Windows or UNIX, create a domain using the `--savemasterpassword=true` option and create a password file to store the admin password. When starting the component, use the `--passwordfile` option to point to the file that contains the password.

For example:

a. Create domain with a saved master password. In this syntax, you are prompted for the admin password and master password:

```
asadmin create-domain --adminport 4848 --adminuser admin
--savemasterpassword=true --instanceport 8080 domain1
```

b. On Windows, create a service using a password file to populate the admin password:

```
C:\windows\system32\sc.exe create domain1 binPath=
"C:\Sun\AppServer\lib\appservService.exe
\"C:\Sun\AppServer\bin\asadmin.bat start-domain --user admin
--passwordfile C:\Sun\AppServer\password.txt domain1\"
\"C:\Sun\AppServer\bin\asadmin.bat stop-domain domain1\"" start=
auto DisplayName= "SJESAS_PE8.1 DOMAIN1"
```

The path to the password file `password.txt` is `C:\Sun\AppServer\password.txt`. It contains the password in the following format

`AS_ADMIN_password=`*password*

For example, for a password `adminadmin`:

```
AS_ADMIN_password=adminadmin
```

c. On UNIX, use the `--passwordfile` option in the line you add to the `inittab` file:

```
das:3:respawn:/opt/SUNWappserver/bin/asadmin start-domain --user
admin --passwordfile /opt/SUNWappserver/password.txt domain1
```

The path to the password file `password.txt` is `/opt/SUNWappserver/password.txt`. It contains the password in the following format

`AS_ADMIN_password=`*password*

For example, for a password `adminadmin`:

```
AS_ADMIN_password=adminadmin
```

Appendix C

Dotted Name Attributes for domain.xml

This appendix describes the dotted name attributes that can be used to address the Mbean and its attributes. Every element in the `domain.xml` file has a corresponding MBean. Because the syntax for using these names involves separating names between periods, these names are called “dotted names.”

This appendix contains the following topics:

Top Level Elements

The following conditions must be adhered to for all top level elements in the domain.xml file:

1. Each server, configuration, cluster, or node agent must have a unique name.
2. Servers, configurations, clusters, or node agents cannot be named “domain.”
3. Server instances can be named “agent.”

The following table identifies the top level elements and the corresponding dotted name prefix.

Table C-1 Top Level Elements

Element Name	Dotted Name Prefix
applications	domain.applications
resources	domain.resources

Table C-1 Top Level Elements

Element Name	Dotted Name Prefix
configurations	domain.configs
servers	domain.servers
	Every server contained in this element is accessible as *server-name*. Where *server-name* is the value of the name attribute for the server sub-element.
clusters	domain.clusters
	Every cluster contained in this element is accessible as *cluster-name*. Where *cluster-name* is the value of the name attribute for the cluster sub-element.
node-agents	domain.note-agents
lb-configs	domain.lb-configs
system-property	domain.system-property

Two levels of aliasing are available:

1. The first level of alias allows access to attributes of server instances or clusters without going through the domain.servers or domain.clusters prefix. So, for example, a dotted name of the form “server1” maps to the dotted name domain.servers.server1 (where server1 is a server instance).
2. The second level of alias is used to refer to configurations, applications, and resources of a cluster or a standalone server instance (target).

The following table identifies dotted names beginning with the server name, or cluster name, that are aliased to top level names under the domain:

Table C-2 Dotted Names Server Name under the Domain

Dotted Name	Aliased to	Comments
target.applications.*	domain.applications.*	The alias resolves to applications referenced by the *target* only.
target.resources.*	domain.resources.*	The alias resolves to all `jdbc-connection-pool`, `connector-connection-pool`, `resource-adapter-config`, and all other resources referenced by the *target*.

The following table identifies dotted names beginning with the server name, or cluster name, that are aliased to top level names within the configuration referenced by the server or cluster.

Table C-3 Dotted Names for Configurations Referenced by the Server or Cluster

Dotted Name	Aliased to
target.http-service	*config-name*.http-service
target.iiop-service	*config-name*.iiop-service
target.admin-service	*config-name*.admin-service
target.web-container	*config-name*.web-container
target.ejb-container	*config-name*.ejb-container
target.mdb-container	*config-name*.mdb-container
target.jms-service	*config-name*.jms-service
target.log-service	*config-name*.log-service
target.security-service	*config-name*.security-service
target.transaction-service	*config-name*.transaction-service
target.monitoring-service	*config-name*.monitoring-service
target.java-config	*config-name*.java-config
target.availability-service	*config-name*.availability-service
target.thread-pools	*config-name*.thread-pools

Elements Not Aliased

A clustered instance should not be aliased. To get a system property for a clustered instance, the dotted name attribute should use as follows:
domain.servers.clustered-instance-name.system-property , not
clustered-instance-name.system-property.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W