GlassFish v3 Application Server Administration Guide

Technology Preview 2

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

Part No: 820–4495–05
May 2008

080502@19860

Contents

Figures

Tables

Examples

Preface

The *GlassFish Application Server v3 Technology Preview 2 Administration Guide* provides instructions for configuring and administering the GlassFish Application Server.

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

Application Server Documentation Set

The Application Server documentation set describes deployment planning and system installation. The Uniform Resource Locator (URL) for Application Server documentation is `http://docs.sun.com/coll/1343.7`. For an introduction to Application Server, refer to the books in the order in which they are listed in the following table.

TABLE P–1 Books in the Application Server Documentation Set

Book Title	Description
Release Notes	Provide late-breaking information about the software and the documentation. Includes a comprehensive, table-based summary of the supported hardware, operating system, Java™ Development Kit (JDK™), and database drivers.
Quick Start Guide	Explains how to get started with the Application Server product.
Installation Guide	Explains how to install the software and its components.
Application Deployment Guide	Explains how to assemble and deploy applications to the Application Server and provides information about deployment descriptors.
Developer's Guide	Explains how to create and implement Java Platform, Enterprise Edition (Java EE platform) applications that are intended to run on the Application Server. These applications follow the open Java standards model for Java EE components and APIs. This guide provides information about developer tools, security, debugging, and creating lifecycle modules.
Java EE 5 Tutorial	Explains how to use Java EE 5 platform technologies and APIs to develop Java EE applications.

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

Book Title	Description
Java WSIT Tutorial	Explains how to develop web applications by using the Web Service Interoperability Technologies (WSIT). The tutorial focuses on developing web service endpoints and clients that can interoperate with Windows Communication Foundation (WCF) endpoints and clients.
Administration Guide	Explains how to configure and manage Application Server subsystems and components from the command line by using the `asadmin`(1M) utility. Instructions for performing these tasks from the Admin Console are provided in the Admin Console online help.
RESTful Web Services Developer's Guide	Explains how to develop Representational State Transfer (RESTful) web services for Application Server.
Getting Started With JRuby on Rails for the GlassFish Application Server	Explains how to develop Ruby on Rails applications for deployment to Application Server.
Getting Started With Project jMaki for the GlassFish Application Server	Explains how to use the jMaki framework to develop Ajax-enabled web applications that are centered on JavaScript™ technology for deployment to Application Server.
Reference Manual	Provides reference information in man page format for Application Server administration commands, utility commands, and related concepts.

Related Documentation

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

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

For information about creating enterprise applications in the NetBeans™ Integrated Development Environment (IDE), see `http://www.netbeans.org/kb/60/index.html`.

For information about the Java DB database for use with the Application Server, see `http://developers.sun.com/javadb/`.

The GlassFish Samples project is a collection of sample applications that demonstrate a broad range of Java EE technologies. The GlassFish Samples are bundled with the Java EE Software Development Kit (SDK), and are also available from the GlassFish Samples project page at `https://glassfish-samples.dev.java.net/`.

Default Paths and File Names

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

TABLE P–2 Default Paths and File Names

Placeholder	Description	Default Value
as-install	Represents the base installation directory for Application Server.	Installations on the Solaris™ operating system and Linux operating system: *user's-home-directory*`/glassfish-v3tp2/glassfish` Windows, all installations: *SystemDrive*`:\Program Files\glassfish-v3tp2\glassfish`
domain-root-dir	Represents the directory containing all domains.	*as-install*`/domains/`
domain-dir	Represents the directory for a domain. In configuration files, you might see *domain-dir* represented as follows: `${com.sun.aas.instanceRoot}`	*domain-root-dir*`/`*domain-name*
instance-dir	Represents the directory for a server instance.	*domain-dir*`/`*instance-name*

Typographic Conventions

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

TABLE P–3 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	A placeholder to be replaced with a real name or value	The command to remove a file is `rm` *filename*.

TABLE P–3 Typographic Conventions *(Continued)*

Typeface	Meaning	Example
AaBbCc123	Book titles, new terms, and terms to be emphasized (note that some emphasized items appear bold online)	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file.

Symbol Conventions

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

TABLE P–4 Symbol Conventions

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

Documentation, Support, and Training

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

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

Searching Sun Product Documentation

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

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

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

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

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

Third-Party Web Site References

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

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

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. To share your comments, go to `http://docs.sun.com` and click Send Comments. In the online form, provide the full document title and part number. The part number is a 7-digit or 9-digit number that can be found on the book's title page or in the document's URL. For example, the part number of this book is 820-4495.

◆ ◆ ◆ CHAPTER 1

About GlassFish Application Server Administration

The GlassFish Application Server provides a Java EE compatible server for the development and deployment of Java EE applications and Java web services.

As administrator, your main responsibilities are to establish a secure Application Server environment and to oversee the services, resources, and users that participate in that environment. Information and instructions on performing the associated tasks using the command-line utility are provided in this document.

In addition, you have responsibilities associated with assembling and deploying application. Information and instructions on performing these tasks are provided in the *GlassFish v3 Application Server Application Deployment Guide*.

The following topics are addressed here:

Overview of the Application Server

The Application Server platform supports services while enabling developers to build applications based on JavaServer Pages (JSP™), Java servlets, and Enterprise JavaBeans™ (EJB™) technology. This section provides a high-level overview of the Application Server.

The following topics are addressed here:

Application Server Concepts

The Application Server consists of one or more domains. Each domain has a domain administration server (DAS) which consists of zero or more standalone instances.

The following topics are addressed here:

Domain

A *domain* is a group of instances that are administered together. The domain provides a preconfigured runtime for user applications. In addition to providing an administration boundary, a domain provides the basic security structure whereby separate administrators can administer specific groups of application server instances. By grouping the server instances into separate domains, different organizations and administrators can share a single installation of Application Server. 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 for other domains are not affected.

The Application Server installer creates a default administrative domain (named `domain1`). The installer also creates an associated DAS (named `server`), with a default administration port of 4848. The installer also queries for the administration user name and master password. After installation, you can create additional administration domains.

Server Instance

An application *server instance* is a single Java EE compatible Java Virtual Machine (JVM) that hosts the Application Server on a single node. Server instances in a domain can run on different physical hosts. Each server instance has a unique name in the domain. An instance belongs to a single domain and has its own directory structure, configuration, and deployed applications as well as the Java EE platform web and EJB containers.

The following figure illustrates an application server instance.

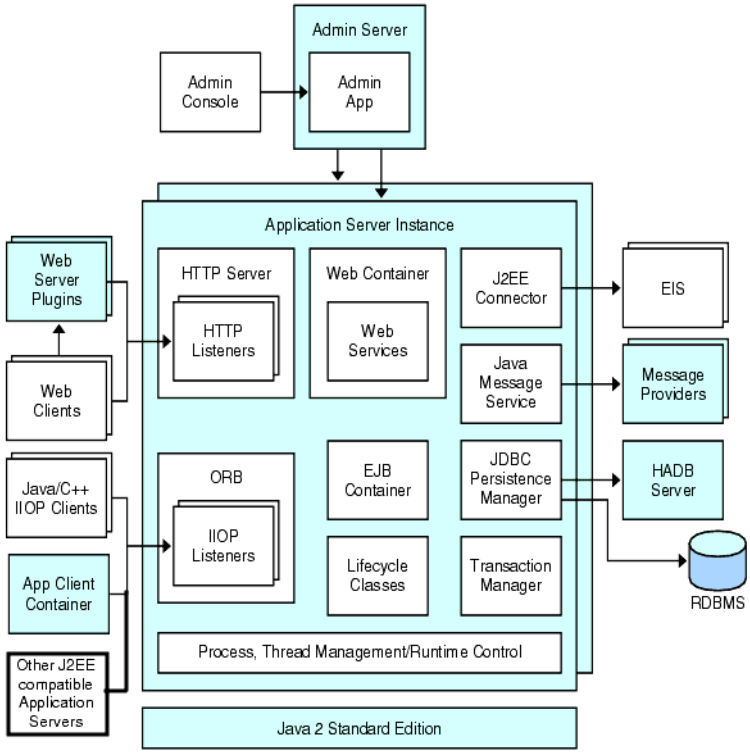

FIGURE 1–1 Application Server Instance

For each application server instance, you can also create virtual servers. Within a single installed instance, you can offer separate domain names, IP addresses, and some administration capabilities to organizations or individuals. For these organizations and individuals, it is as if they have their own Web server, without the hardware and server maintenance. These virtual servers do not span application server instances.

Domain Administration Server (DAS)

The *domain administration server* (DAS) is a specially-designated application server instance that hosts the administrative applications. The DAS authenticates the administrator, accepts requests from administration tools, and communicates with server instances in the domain to carry out requests. The DAS is sometimes referred to as the *default server* because it is the only server instance created during Application Server installation that can be used for deployment.

Each domain has its own DAS with a unique port number. The graphical Admin Console communicates with a specific DAS to administer to administer the domain associated with the

DAS. Each Admin Console session enables you to configure and manage the specific domain. If you create multiple domains, you must start a separate Admin Console session to manage each domain.

Application Server Architecture

This section describes the components of the Application Server architecture as illustrated in Figure 1–2. The following topics are addressed here:

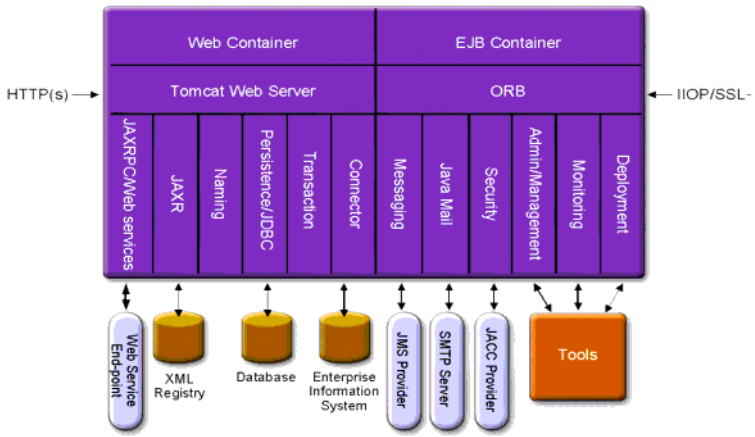

FIGURE 1–2 Application Server Architecture

Containers

A *container* is a runtime environment that provides services such as security and transaction management to Java EE components. The Web and Enterprise JavaBeans (EJB) containers are illustrated in Figure 1–2.

- **Web Container** Web components, such as JSP pages and servlets, run within the Web container.
- **EJB Container** Enterprise beans run within the EJB container. Enterprise beans 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* represent transient objects and processes and typically are used by a single client. *Entity beans* represent persistent data and are typically maintained in a database. *Message-driven beans* are used to pass messages asynchronously to application modules and services.

Services for Applications

The Java EE platform is designed so that containers provide services for the applications.

Naming	A naming and directory service binds objects to names. A Java EE application locates an object by looking up its Java Naming and Directory Interface (JNDI) API name.
Security	The Java Authorization Contract for Containers (JACC) is a set of security contracts defined for the Java EE containers. Based on the client's identity, the containers restrict access to the container's resources and services.
Transactions	A transaction is an indivisible unit of work. For example, transferring funds between bank accounts is a transaction. The transaction service ensures that a transaction either completes fully or is rolled back.

Client Access

At runtime, browser clients access Web applications by communicating with the Web server using HTTP, the protocol used throughout the Internet. Applications that require secure communication use the HTTPS protocol.

The Application Server provides separate listeners for the HTTP and HTTPS protocols. Each listener has exclusive use of a specific port.

Web Services

On the Java EE platform, it is possible to deploy a Web application that provides a Web service implemented by the Java API for XML-Based RPC (JAX-RPC). A Java EE application or component can also be a client to other Web services.

Applications access XML registries through the Java API for XML Registries (JAXR).

Access to External Systems

The Java EE platform enables applications to access systems that are outside the Application Server. Applications connect to these systems through objects called *resources*. One of the responsibilities of an administrator is configuring access to resources. The Java EE 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. The information in databases is often described

	as *persistent data* because it is saved on disk and exists after the application process ends. Most business applications store data in relational databases. In the J2EE platform, applications can access database information by using the Java Database Connectivity (JDBC) API.
Connector	The Java EE connector architecture enables integration between Java EE applications and an existing Enterprise Information System (EIS). An application accesses an EIS through a portable Java EE component called a *connector module*, also known as a resource adapter.
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.
JavaMail	Through the JavaMail API, applications connect to a Simple Mail Transfer Protocol (SMTP) server in order to send and receive email.
JBI	Java Business Integration (JBI) defines an environment for plugin-in components that interact using a services model based on Web Services Description Language (WSDL).

Overview of Administrative Tasks

Application Server administrative tasks include the following:

- Creating and managing domains, server instances, and resources
- Starting and stopping domains, server instances, and databases
- Implementing server security and user security
- Administering services
- Assembling and deploying applications
- Monitoring and managing performance
- Diagnosing and resolving problems

The instructions in this guide are organized around your tasks.

Step-by-step instructions for using the Admin Console are provided in the Admin Console online help. Procedures for using the command-line utilities are provided in the manuals and the command-line help pages. For the most part, you can perform the same tasks by using either the Admin Console or the command-line utility, however, there are exceptions.

Instructions for manually editing system files are provided when there is no way to accomplish the tasks by using the Admin Console or command-line utility.

Overview of Administrative Tools and Files

Most of the tasks for administering the Application Server can be accomplished using the Admin Console or the `asadmin` command-line utility, or by manually editing system files.

Note – Instructions written for the Application Server tools use UNIX-standard forward slashes (/) for directory path separators in commands and files names. If you are running Application Server on a Microsoft Windows system, use blackslashes (\) instead. For example:

- UNIX: *install-dir*`/bin/asadmin`
- Windows: *install-dir*`\bin\asadmin`

Administrative tasks are performed by using the following tools and files:

- "Admin Console" on page 27
- "Command-Line Utility" on page 28
- "Java Monitoring and Management Console (JConsole)" on page 30
- "Configuration Files" on page 31

Admin Console

The *Admin Console* is a browser-based utility that features an easy-to-navigate graphical interface that includes extensive online help. From the General tab of the Admin Console, you can perform such tasks as the following:

- Starting and stopping a domain
- Viewing logs
- Rotating a log file for an instance
- Browsing the JNDI tree for a running instance

The following tabs allow you to perform additional tasks:

- Resources tab: Managing a selected resource
- Properties tab: Configuring instance-specific properties
- Logging tab: Configuring the logging levels
- Applications tab: Deploying a selected application
- Advanced tab: Setting general properties for deploying applications

To Start the Admin Console

To use the Admin Console, the domain administration server (DAS) must be running. Each domain has its own DAS, which has a unique port number. When specifying the URL for the Admin Console, you use the port number for the domain to be administered. When the

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

The format for starting the Admin Console in a web browser is `http://`*hostname*:*port*. For example:

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

If the Admin Console is running on the host where the Application Server was installed, specify `localhost` for the host name. For example:

```
http://localhost:8080/admin
```

For Microsoft Windows, an alternative way to start the Application Server Admin Console is by using the Start menu.

Command-Line Utility

The `asadmin` utility is a command-line tool that invokes commands for identifying the operation or task you want to perform. Short option arguments have a single dash (-); while long option arguments have two dashes (--). Options control how the utility performs a command. Commands and options are case-sensitive.

You usually perform `asadmin` commands either from a command prompt or from a script. Running `asadmin` commands from a script is helpful for automating repetitive tasks.

Remote Commands and Local Commands

Remote commands are always run by connecting to a running administration server and running the command there. Remote commands require the options described in `asadmin`(1M) man page, that is, the `--host`, `--port`, `--user`, and `--passwordfile` options. If you do not specify the `--host` and `--port` options on the command line, the local host and port are assumed. If you do not include the `--user` and `--passwordfile` options on the command line, you are prompted for the login values (unless security is not enabled).

Local commands can be run without a running administration server. However, you must be logged into the host that is hosting the domain and have access (permissions) for the installation and domain directories. For commands that can be run locally:

- You can use the `--domain` option to specify the domain of interest (assuming the domain as the default domain, if there is only one).
- If there is more than one domain, the `--domain` option is a required option.

Help for the asadmin Utility

You can display the help page for an `asadmin` command by using the `--help` option. For example:

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

Note – To display the help page for a remote command, the Application Server must be running.

You can use the `list-commands` command to display the available commands. The `asadmin` utility help pages are available in HTML and PDF format in the *GlassFish v3 Application Server Reference Manual*.

Using the `--help` option without a command displays the help page for the `asadmin` utility. The `asadmin`(1M) help page explains the basics of how the `asadmin` command works. The following help pages provide additional Application Server conceptual material:

`application`(5ASC)	Describes application concepts.
`configuration`(5ASC)	Describes Application Server configuration.
`domain`(5ASC)	Describes Application Server domains.
`dotted-names`(5ASC)	Describes how dotted names work in Application Server.
`instance`(5ASC)	Describes Application Server instances.
`logging`(5ASC)	Describes how Application Server logging works.
`passwords`(5ASC)	Describes how Application Server passwords work.
`resource`(5ASC)	Describes Application Server resources.
`security`(5ASC)	Describes how security works in the Application Server.

To Start the `asadmin` Utility

The default installation directory on Solaris is `/opt/SUNWappserver`.

To start the `asadmin` utility, go to the default *as-install*`/bin` directory and type:

```
./asadmin
```

You can issue an `asadmin` command at the shell's command prompt. For example:

```
asadmin help
```

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

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

To see a list of the `asadmin` commands included in this release, refer to Appendix A, "The `asadmin` Utility Commands".

Java Monitoring and Management Console (JConsole)

Java SE 5 enhances management and monitoring of the Java Virtual Machine (JVM) by including a Platform MBean Server and by including managed beans (MBeans) to configure the JVM. The Application Server leverages these enhancements and registers its MBeans with the Platform MBean Server. Thus, a JMX connector client gets a unified view of JVM MBeans as well as Application Server MBeans.

To view all the MBeans, Application Server provides a configuration of the standard JMX connector server called System JMX Connector Server. As part of Application Server startup, an instance of this JMX Connector Server is started. Any compliant JMX Connector Client can connect to the server using the JMX Connector Server. Java SE also provides tools to connect to an MBean Server and view the MBeans registered with the server. JConsole is one such popular JMX Connector Client and is available as part of the standard Java SE distribution. JConsole is used to monitor the Application Server.

▼ To Set Up JConsole Connectivity

1 Add the following system properties to your domain's JVM options in the `domain.xml` file:

- `com.sun.management.jmxremote.port=jmx-connector-port-number`

  For example, 8686
- `com.sun.management.jmxremote.host=jmx-connector-hostname`

  For example, 129.92.42.134

2 Start JConsole and specify the JMX Service URL:

```
service:jmx:rmi:///jndi/rmi://localhost:8686/jmxrmi
```

3 Browse GlassFish v3 Technology Preview 2 MBeans along with Platform MBeans.

You can use either the JConsole Remote tab, or the Advanced tab to connect to the Application Server.

Remote tab	Specify the username, password, administration server host, and JMS port number (8686 by default), and then select Connect.
Advanced tab	Specify the `JMXServiceURL` as service:`jmx:rmi:///jndi/rmi://host:jms-port/jmxrmi`, and then select Connect. The `JMXServerURL` is printed in the `server.log` file and is

displayed in the command window of the domain creation command.

See Also For more information on JConsole, see
`http://java.sun.com/javase/6/docs/technotes/guides/management/jconsole.html`

Configuration Files

The bulk of the Application Server configuration information is stored in the `domain.xml` file. This file is the central repository for a given administrative domain and contains an XML representation of the Application Server domain model. The contents of the `domain.xml` file are governed by the J2EE specification expressed in the form of the Document Type Definition (DTD) for the domain.

The following configuration files are associated with Application Server administration:

- `asenv.conf`
- `asadminenv.conf`
- `domain.xml`
- `jbi.xml`
- `resources.xml`
- `server.policy`
- `sun.acc.xml`
- `webservices.xml`
- `wss-client-config.xml`

You can use either the Admin Console or the command-line utility to make changes in the configuration files. Either method is preferable to editing the configuration files directly, because direct editing is very prone to error and can have unintended results.

Configuration changes often require that you restart the Application Server for the changes to take effect. In other cases, changes are applied dynamically without shutting down the Application Server.

When making any of the following configuration changes, you must restart the server for the changes to take effect:

- Creating or deleting a resource or entity
- Changing JVM options
- Changing port numbers
- Managing HTTP services
- Managing thread pools
- Modifying the following JDBC connection pool properties:

  - `datasource-classname`
  - `associate-with-thread`
  - `lazy-connection-association`
  - `lazy-connection-enlistment`
  - JDBC driver vendor-specific properties
- Modifying the following connector connection pool properties:
  - `resource-adapter-name`
  - `connection-definition-name`
  - `transaction-support`
  - `associate-with-thread`
  - `lazy-connection-association`
  - `lazy-connection-enlistment`
  - Vendor-specific properties

With *dynamic configuration*, changes take effect while the server is running. To make the following configuration changes, do *NOT* restart the server:

- 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
- Deploying and undeploying and redeploying applications

◆ ◆ ◆ CHAPTER 2

Basic Administration

This chapter provides procedures for performing basic administration tasks in the GlassFish Application Server environment by using the `asadmin` command-line utility.

The following topics are addressed here:

Instructions for accomplishing these tasks by using the Admin Console are contained in the Admin Console online help.

Administering Domains

A domain provides a preconfigured runtime for user applications. This runtime includes a basic security structure where specific groupings of server instances (domains) can be administered by different administrators. The Application Server installer creates a default administrative domain (named `domain1`), as well as an associated domain administration server (named `server`). The default administration port is 4848, but a different port can be specified during installation. The administration user name and master password are also established during installation. Every domain has an associated profile.

The following tasks and information are used to administer domains:

- "To Back Up a Domain" on page 38
- "To Restore a Domain" on page 38

Command Options for Administering Domains

Options for administering domains include the following:

Domain Name	The name of the domain to be created.
`--domaindir`	The directory where the domain is to be created. If specified, the path must be accessible in the file system. If not specified, the domain is created in the default domain directory.
`--template`	The file name of a `domain.xml` template used to create the domain. This option allows domains of different types to be created, and also enables you to define your own template.
`--adminport`	The HTTP/S port for administration. This is the port to which you should point your browser to manage the domain. Either the `--adminport` option or the `--portbase` option must be specified.
`--portbase`	Determines the number with which the port assignment should start. A domain uses a certain number of ports that are statically assigned. The value of the `--portbase` option determines where the assignment should start. Choose this value carefully. For more information on this option, see `create-domain`(1). The `--portbase` option cannot be used with the `--adminport` or the `--instanceport` options.
`--instanceport`	The domain provides services so that applications can run when deployed. This (HTTP) port specifies where the web application context roots are available for a Web browser to connect to. This port is a positive integer and must be available when the domain is created.
`--domainproperties`	Setting the optional name/value pairs overrides the default values for the properties of the domain to be created. The pairs must be separated by the colon (:) character.

For more information on these and other options, see `create-domain`(1).

▼ To Create a Domain

After installation, you can create additional domains by using the local `create-domain` command.

Before You Begin Determine which profile will apply to the domain.

1. **Select a name for the domain that you are creating.**

   You can verify that a name is not already in use by listing the existing domains:

   ```
   asadmin list-domains
   ```

2. **Create a domain by using the `create-domain`(1) command.**

Example 2–1 Creating a Domain

The following example command creates a domain named `mydomain`. The administration server listens on port 5000; the administrative user name is `admin`. When you type the command, you are prompted for the administrative and master passwords.

```
asadmin create-domain --adminport 5000 --adminuser admin mydomain
```

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

`http://`*hostname*`:5000`

For this example, the domain's log files, configuration files, and deployed applications now reside in the following directory:

domain-root-dir`/mydomain`

See Also To see the full syntax of the command, type `asadmin create-domain --help` at the command line.

▼ To List Domains

If the domain directory is not specified, the contents of the default *as-install*`/domains` directory is listed. If there is more than one domain, the domain name must be specified.

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

- **List domains by using the `list-domains`(1) command.**

Example 2–2 Listing Domains

The following example command lists the domains in the default *as-install*`/domains` directory:

```
asadmin list-domains
```

See Also To see the full syntax of the command, type `asadmin list-domain --help` at the command line.

▼ To Delete a Domain

Only the operating system user who can administer the domain (or root) can run this command successfully.

1. **Obtain the exact name of the domain that you are deleting.**

   To list the existing domains:

   ```
   asadmin list-domains
   ```

2. **Notify any users of the domain that the domain is being deleted.**

3. **Ensure that the domain you want to delete is stopped.**

   For instructions, see "To Stop a Domain (or Server)" on page 37.

4. **Delete the domain by using the `delete-domain`(1) command.**

Example 2–3 Deleting a Domain

The following example command deletes a domain named `mydomain`:

```
asadmin delete-domain mydomain
```

See Also To see the full syntax of the command, type `asadmin delete-domain --help` at the command line.

▼ To Start a Domain (or Server)

When you start a domain, the administration server and application server instance are started. After startup, the application server instance runs constantly, listening for and accepting requests. Each domain must be started separately.

Note – For Microsoft Windows, you can use an alternate method to start a domain. From the Windows Start menu, select Programs → Sun Microsystems →Application Server → Start Admin Server.

Restarting the server is the same as stopping and then starting the domain. Instructions for stopping are contained in "To Stop a Domain (or Server)" on page 37.

- **Start a domain by using the `start-domain`(1) command.**

Example 2–4 Starting a Domain

The following example command starts the default domain (`domain1`):

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

If there is only one domain, omit the domain name. If you do not include the password, you are prompted to supply it.

See Also To see the full syntax of the command, type `asadmin start-domain --help` at the command line.

▼ To Stop a Domain (or Server)

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 shutdown process takes a few seconds. While the domain is stopped, the Admin Console and most of the `asadmin` commands cannot be used.

Note – For Microsoft Windows, you can use an alternate method to stop a domain. From the Start menu, select Programs → Sun Microsystems → Application Server → Stop Admin Server.

Restarting the server is the same as stopping and then starting the domain. Instructions for starting are contained in "To Start a Domain (or Server)" on page 36.

1. **Notify any users of the domain that you are going to stop the domain.**
2. **Stop the domain by using the `stop-domain`(1) command.**

Example 2–5 Stopping a Domain (or Server)

The following example command stops the default domain (`domain1`):

```
asadmin stop-domain domain1
```

See Also To see the full syntax of the command, type `asadmin stop-domain --help` at the command line.

▼ To Back Up a Domain

The local `backup-domain` command enables you to make a backup copy of the files under the specified domain.

- **Make a backup copy of a domain by using the `backup-domain`(1) command.**

Example 2–6 Backing Up Domain Files

The following example command makes a backup copy of the files in the default domain (`domain1`):

```
asadmin backup-domain --domaindir /opt/SUNWappserver/mydomaindir domain1
```

See Also To see the full syntax of the command, type `asadmin backup-domain --help` at the command line.

▼ To Restore a Domain

This local `restore-domain` command enables you to restore domain files from the backup copy of the specified domain.

1. **Notify any users of the domain that the domain is being restored from backup.**
2. **Restore backup files for a domain by using the `restore-domain`(1) command.**
3. **Verify that the restore has succeeded.**
4. **Notify users that the domain has been restored and is available.**

Example 2–7 Restoring Backup Files for a Domain

The following example command restores files for the default domain (`domain1`) from a backup copy called `sjssas_backup_v00001.zip`:

```
asadmin restore-domain --domaindir /opt/SUNWappserver/mydomain/domain1
--filename sjssas_backup_v00001.zip
```

See Also To see the full syntax of the command, type `asadmin restore-domain --help` at the command line.

Administering Resources

This section contains instructions for integrating resources into the Application Server environment. Information on administering specific resources, such as JDBC, are contained in other chapters.

The following topics are addressed here:

▼ To Add Resources

The remote `add-resources` command enables you to create the resources named in the specified XML file.

1. **Ensure that the server is running.**

   Remote commands require a running server.

2. **Add resources from an XML file by using the `add-resources`(1) command.**

Example 2–8 Adding Resources

The following example command creates resources using the contents of the `resource.xml` file:

```
asadmin add-resources --user admin --passwordfile password.txt
--host localhost --port 4848 resource.xml
```

See Also To see the full syntax of the command, type `asadmin add-resources --help` at the command line.

▼ To Create a Resource Reference

The remote `create-resource-ref` command enables you to create a reference to an existing resource. This effectively results in the resource being made available in the JNDI tree of the targeted instance.

1. **Ensure that the server is running.**

   Remote commands require a running server.

2. **Create a resource reference by using the `create-resource-ref`(1) command.**

3. **Notify users that the resource has been enabled.**

Example 2–9 Creating a Resource Reference

The following example command adds a resource reference named `jms/Topic` on the default server:

```
asadmin create-resource-ref --user admin
--passwordfile passwords.txt jmsTopic
```

See Also To see the full syntax of the command, type `asadmin create-resource-ref --help` at the command line.

▼ To List Resource References

The remote `list-resource-refs` command enables you to list existing resource references. This effectively lists all the resources (for example, JDBC resources) available in the JNDI tree of the specified target.

1. **Ensure that the server is running.**

   Remote commands require a running server.

2. **List resource references by using the `list-resource-refs`(1) command.**

Example 2–10 Listing Resource References

The following example command lists resource references for the default server:

```
asadmin list-resource-refs --user admin --passwordfile passwords.txt
```

See Also To see the full syntax of the command, type `asadmin list-resource-refs --help` at the command line.

▼ To Delete a Resource Reference

The remote `delete-resource-ref` command enables you to delete a reference to an existing resource. This effectively results in the resource being disabled. The resource is not removed from the domain, but only for the specified resource.

1. **Ensure that the server is running.**

   Remote commands require a running server.

2. **Obtain the exact name of the resource reference that you are deleting.**

   To list the existing resource references:

   ```
   asadmin list-resource-refs
   ```

3. **Notify users that the resource is being disabled.**

4. **Delete a resource reference by using the `delete-resource-ref`(1) command.**

Example 2–11 Deleting a Resource Reference

The following example command deletes a resource reference named `jms/Topic` from the default server:

```
asadmin delete-resource-ref --user admin --passwordfile passwords.txt jms/Topic
```

See Also To see the full syntax of the command, type `asadmin delete-resource-ref --help` at the command line.

Administering System Properties

Shared server instances will often need to override attributes defined in their referenced configuration. Any configuration attribute in a server instance can be overridden through a system property of the corresponding name.

Settings for System Properties

System properties are administered using the following settings:

Dynamic Reconfiguration	If this setting is enabled, modifications to the configuration are applied without restarting the server.
Instance Variable Name	Identities the name of the property.
Default Value	Identifies the default value of the property.

The predefined properties that are supplied with the Application Server include the following:

`HTTP_LISTENER_PORT`	Port number for the secure HTTP listener-2.
`HTTP_SSL_LISTENER_PORT`	Port number for HTTP listener-1.

▼ To Create System Properties

The remote `create-system-properties` command enables you to add or update one or more system properties of the domain, configuration, or server instance.

1. **Ensure that the server is running.**

   Remote commands require a running server.

2. **Create system properties by using the `create-system-properties`(1) command.**

Example 2–12 Creating System Properties

The following example command creates a system property associated with `http-listener-port=1088` on the local host:

```
asadmin create-system-properties --user admin
--passwordfile password.txt --host localhost
--port 4848  http-listener-port=1088
```

See Also To see the full syntax of the command, type `asadmin create-system-properties --help` at the command line.

▼ To List System Properties

The remote `list-system-properties` command enables you to list the system properties that apply to a domain or configuration.

1. **Ensure that the server is running.**

   Remote commands require a running server.

2. **List system properties by using the `list-system-properties`(1) command.**

   The existing system properties are displayed, including predefined properties such as `HTTP_LISTENER_PORT` and `HTTP_SSL_LISTENER_PORT`.

Example 2–13 Listing System Properties

The following example command lists the system properties associated with `http-listener-port` 1088:

```
asadmin list-system-properties --user admin --passwordfile password.txt
--host localhost --port 4848  http-listener-port=1088
```

See Also To see the full syntax of the command, type `asadmin list-system-properties --help` at the command line.

▼ To Delete a System Property

Any configuration attribute can be overwritten through a system property of the corresponding name. You can delete system properties by using the remote `delete-system-property` command.

1. **Ensure that the server is running.**

   Remote commands require a running server.

2. **Obtain the exact name of the system property that you are deleting.**

   To list the existing system properties:

   ```
   asadmin list-system-properties
   ```

3. **Delete the system property by using the `delete-system-property`(1) command.**

4. **If needed, notify users that the system property has been deleted.**

Example 2–14 Deleting a System Property

The following example command deletes a system property named `http-listener-port` from the default server:

```
asadmin delete-system-property --host localhost
--port 4848 http-listener-port
```

See Also To see the full syntax of the command, type `asadmin delete-system-property --help` at the command line.

Listing System Elements

▼ To List Applications

The remote `list-applications` command enables you to list the deployed Java EE 5 applications. If the `--type` option is not specified, all applications are listed.

1. **Ensure that the server is running.**
   Remote commands require a running server.
2. **List applications by using the `list-applications`(1) command.**

Example 2–15 Listing Applications

The following example command lists the web applications on the local host:

```
asadmin list-applications --user admin --passwordfile password.txt --type web
```

See Also To see the full syntax of the command, type `asadmin list-applications --help` at the command line.

▼ To List Commands

The remote `list-commands` command enables you to list the deployed Java EE 5 applications. You can specify that only remote commands or only local commands are listed. By default, this command displays a list of local commands followed by a list of remote commands.

1. **Ensure that the server is running.**
   Remote commands require a running server.
2. **List commands by using the `list-commands`(1) command.**

Example 2–16 Listing Commands

The following example command lists only remote commands:

```
asadmin list-commands --user admin
--passwordfile password.txt --remoteonly
```

See Also To see the full syntax of the command, type `asadmin list-commands --help` at the command line.

▼ To List Containers

The remote `list-containers` command enables you to display a list of application containers.

1. **Ensure that the server is running.**
   Remote commands require a running server.
2. **List containers by using the `list-containers`(1) command.**

Example 2–17 Listing Containers

The following example command lists the application containers on the local host:

```
asadmin list-containers --user admin1 --passwordfile passwords.txt
```

See Also To see the full syntax of the command, type `asadmin list-containers --help` at the command line.

▼ To List Modules

The remote `list-modules` command enables you to display a list of modules that are accessible to the Application Server module subsystem. The status of each module is included. Possible statuses include New and Ready.

1. **Ensure that the server is running.**
   Remote commands require a running server.
2. **List modules by using the `list-modules`(1) command.**

Example 2–18 Listing Modules

The following example command lists the accessible modules:

```
asadmin list-modules --user admin1 --passwordfile passwords.txt
```

See Also To see the full syntax of the command, type `asadmin list-modules --help` at the command line.

◆ ◆ ◆ CHAPTER 3

Administering the Java Virtual Machine (JVM)

This chapter provides procedures for administering the JVM in the GlassFish Application Server environment by using the `asadmin` command-line utility.

The following topics are addressed here:

Instructions for accomplishing these tasks by using the Admin Console are contained in the Admin Console online help.

About the JVM

The Java Virtual Machine (JVM™) is an interpretive computing engine responsible for running the byte codes in a compiled Java program. The JVM translates the Java byte codes into the native instructions of the host machine. The Application Server, being a Java process, requires a JVM in order to run and support the Java applications running on it. JVM settings are part of an Application Server configuration.

Profilers generate information used to analyze server performance. If JVM options are created for a profiler, the options are used to record the settings needed to activate a particular profiler. You can use the `create-jvm-options` command to create JVM options in the Java configuration or profiler elements of the `domain.xml` file.

Managing the JVM Options

The following tasks and information are used to manage JVM options:

Settings for the JVM

As part of configuring the Application Server, you define settings that improve the operation of the JVM.

Java Home	Specifies the installation directory of the Java software. The Application Server relies on the Java SE software. If you enter a nonexistent directory name or the installation directory name of an unsupported version of the Java EE software, then the Application Server will not start.
Javac Options	Specifies the command-line options for the Java programming language compiler (Javac). The Application Server runs this compiler when EJB components are deployed.
Debug	Enables debugging with the Java Platform Debugger Architecture (JPDA). JPDA is used by application developers. Default is Enabled.
Debug Options	Specifies the JPDA options passed to the JVM when debug is enabled.
RMI Compile Options	Specifies the command-line options for the Remote Method Invocation compiler (rmic). The Application Server runs this compiler when EJB components are deployed.
Bytecode Preprocessor	Specifies 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.

▼ To Create JVM Options

The remote `create-jvm-options` command enables you to create JVM options in the Java configuration or the profiler elements of the `domain.xml` file. If JVM options are created for a profiler, these options are used to record the settings that initiate the profiler.

1. **Ensure that the server is running.**

   Remote commands require a running server.

2. **Create JVM options by using the `create-jvm-options`(1) command.**

   To create more than one JVM option, use a colon (:) to separate the options. If the JVM option itself contains a colon (:), use the backslash (\) to offset the colon (:) delimiter.

3. **To apply your changes, restart the Application Server.**

   a. **Stop the Application Server.**

      For instructions, see "To Stop a Domain (or Server)" on page 37.

   b. **Start the Application Server.**

      For instructions, see "To Start a Domain (or Server)" on page 36.

Example 3–1 Creating JVM Options

```
asadmin create-jvm-options --interactive=true --secure=true  --terse=false
 --host localhost --port 4848 server\\-Dunixlocation=/root/example:
-Dvariable=\$HOME:-Dwindowslocation=d\\:\\\sun\\\appserver:-Doption1=-value1
```

See Also To see the full syntax of the command, type `asadmin create-jvm-options --help` at the command line.

▼ To List JVM Options

The remote `list-jvm-options` command enables you to list the existing JVM options.

1. **Ensure that the server is running.**

   Remote commands require a running server.

2. **List JVM options by using the `list-jvm-options`(1) command.**

Example 3–2 Listing JVM Options

The following example command lists all JVM options:

```
asadmin list-jvm-options --user admin1 --passwordfile passwords.txt
```

See Also To see the full syntax of the command, type `asadmin list-jvm-options --help` at the command line.

▼ To Delete JVM Options

The remote `delete-jvm-options` command enables you to delete JVM options from the Java configuration or profiler elements of the `domain.xml` file.

1. **Ensure that the server is running.**

   Remote commands require a running server.

2. **Obtain the exact name of the JVM option that you are deleting.**

   To list the existing JVM options:

   ```
   asadmin list-jvm-options
   ```

3. **Notify users that the JVM option is being deleted.**

4. **Delete JVM options by using the `delete-jvm-options`(1) command.**

   To remove more than one JVM option, use a colon (:) to separate the options. If the JVM option itself contains a colon (:), use the backslash (\) to offset the colon (:) delimiter.

5. **To apply your changes, restart the Application Server.**

   a. **Stop the Application Server.**

      For instructions, see "To Stop a Domain (or Server)" on page 37.

   b. **Start the Application Server.**

      For instructions, see "To Start a Domain (or Server)" on page 36.

Example 3–3 Deleting a JVM Option

The following example command removes a single JVM option:

```
asadmin delete-jvm-options -e --interactive=true --secure=true --terse=false
 server --host localhost --echo=true --port 4848 "\-Dtmp=sun"
```

Example 3–4 Deleting Multiple JVM Options

The following example command removes two JVM options:

```
asadmin delete-jvm-options -e \-Doption1=value1--interactive=true
--secure=true --terse=false  server --host localhost
--echo=true --port 4848 "\-Doption1=value1:-Doption2=value2"
```

See Also To see the full syntax of the command, type `asadmin delete-jvm-options --help` at the command line.

Managing Profilers

A server instance is associated with a a particular profile by the profiler element in the Java configuration.

The following tasks and information are used to manage profiles:

- "Command Options for Managing Profilers" on page 51
- "To Create a Profiler" on page 51
- "To Delete a Profiler" on page 52

Command Options for Managing Profilers

Options for managing profilers include the following:

Profiler Name	Name of the profiler.
`--classpath`	Java classpath string that specifies the classes needed by the profiler.
`--nativelibpath`	The native library path is automatically constructed to be a concatenation of the Application Server installation relative path for its native shared libraries, standard JRE native library path, the shell environment setting (`LD_LIBRARY_PATH` on UNIX) and any path that is specified in the profile element.
`--property`	Name/value pairs of provider specific attributes.

For more details on these and other options, see `create-profiler`(1) in the reference manual.

▼ To Create a Profiler

The remote `create-profiler` command enables you to create a profiler element in the Java configuration.

1. **Ensure that the server is running.**
   Remote commands require a running server.
2. **Create a profiler by using the `create-profiler`(1) command.**
3. **To apply your changes, restart the Application Server.**
   a. **Stop the Application Server.**
      For instructions, see "To Stop a Domain (or Server)" on page 37.
   b. **Start the Application Server.**
      For instructions, see "To Start a Domain (or Server)" on page 36.

Example 3–5 Creating a Profiler

The following example command creates a profiler named `sample_profiler`:

```
asadmin create-profiler --user admin --passwordfile password.txt
--host localhost --port 4848 --classpath /home/appserver/
--nativelibpath /u/home/lib --enabled=false
--property defaultuser=admin:password=adminadmin sample_profiler
```

See Also To see the full syntax of the command, type `asadmin create-profiler --help` at the command line.

▼ To Delete a Profiler

The remote `delete-profiler` command enables you to delete a profiler element from the Java configuration.

1. **Ensure that the server is running.**
   Remote commands require a running server.
2. **Notify users that the profiler is being deleted.**
3. **Delete the profiler by using the `delete-profiler`(1) command.**
4. **To apply your changes, restart the Application Server.**
   a. **Stop the Application Server.**
      For instructions, see "To Stop a Domain (or Server)" on page 37.

b. Start the Application Server.

For instructions, see "To Start a Domain (or Server)" on page 36.

Example 3–6 Deleting a Profiler

The following example command deletes the profiler named `sample_profile`:

```
delete-profiler --user admin --host localhost
--port 4848 sample_profiler
```

See Also To see the full syntax of the command, type `asadmin delete-profiler --help` at the command line.

◆ ◆ ◆ CHAPTER 4

Administering Database Connectivity

This chapter provides procedures for performing database connectivity tasks in the GlassFish Application Server environment by using the `asadmin` command-line utility.

The following topics are addressed here:

Instructions for accomplishing these tasks by using the Admin Console are contained in the Admin Console online help.

About Database Connectivity

Most applications use relational databases to store, organize, and retrieve data. J2EE applications access relational databases through the Java Database Connectivity (JDBC) API.

The following topics are addressed here:

JDBC Resources

A *JDBC resource*, also known as a data source, provides an application with a means of connecting to a database. Typically, you create a JDBC resource for each database that is accessed by the applications deployed in a domain. Multiple JDBC resources can be specified for a database.

A JDBC resource is created by specifying a unique Java Naming and Directory Interface (JNDI) name that identifies the resource. Expect to find the JNDI name of a JDBC resource in `java:comp/env/jdbc` subcontext. For example, the JNDI name for the resource of a payroll database might 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, use only the `jdbc/`*name* format. For example, for a payroll database specify `jdbc/payrolldb`.

JDBC Connection Pools

A *JDBC connection pool* is a group of reusable connections for a particular database. Because creating each new physical connection is time consuming, the Application Server maintains a pool of available connections. 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.

A JDBC resource is created by specifying the connection pool with which the resource is associated. Multiple JDBC resources can specify a single connection pool.

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

How JDBC Resources and Connection Pools Work Together

Before an application can access a database, it must get a connection. At runtime, the following sequence occurs 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.

   Using the JNDI name of the resource, the naming and directory service locates the JDBC resource. Each JDBC resource specifies a connection pool.
2. Using the JDBC resource, the application gets a database connection.

   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. After the database connection is established, the application can read, modify, and add data to the database.

   The application accesses 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 the application is finished accessing the database, the application closes the connection.

The application returns the connection to the connection pool where the connection becomes available for the next application.

Setting Up Database Access

The general sequence of events for implementing database connectivity includes setting up your chosen database, then configuring a connection pool, and finally creating a JDBC resource.

The following topics are addressed here:

▼ To Set Up a Database

1 **Install a supported database product.**

To see the current list of database products supported by the Application Server, refer to the *GlassFish v3 Application Server Release Notes*.

2 **Install a supported JDBC driver for the database product.**

For a list of drivers supported by the Application Server, see "Configuration Specifics for JDBC Drivers" on page 69.

3 **Make the JAR file for the JDBC driver accessible to the domain server instance.**

4 **Create the database.**

Usually, the application provider delivers scripts for creating and populating the database.

5 **Create a connection pool for the database.**

For instructions, see "Managing JDBC Connection Pools" on page 59.

6 **Create a JDBC resource that points to the connection pool.**

For instructions, see "Managing JDBC Resources" on page 66.

7 **Integrate the JDBC driver into an administrative domain.**

Do either of the following:

- **Make the driver accessible to the common class loader, and restart the domain.**

  Copy the driver's JAR and ZIP files into the *domain-dir*`/lib` directory or copy its class files into the *domain-dir*`/lib/ext` directory.

- **Identify the fully-qualified path name for the driver's JAR file.**

▼ To Start the Database

The Application Server includes an implementation of Java DB, however, you can use any JDBC-compliant database engine. The database is not started automatically when you start the Application Server, so if you have applications that need a database, you need to start Java DB manually by using the local `start-database` command.

● **Start the database by using the `start-database`(1) command.**

When the database server starts, or a client connects to it successfully, the following files are created at the location that is specified by the `--dbhome` option:

- The `derby.log` file that contains the database server process log along with its standard output and standard error information
- The database files that contain your schema (for example, database tables)

Example 4–1 Starting a Database

The following example command starts Java DB on `host1`, port 5001:

```
start-database --dbhost host1 --dbport 5001
```

See Also To see the full syntax of the command, type `asadmin start-database --help` at the command line.

▼ To Stop the Database

The local `stop-database` command enables you to stop Java DB on a specified port. A single host can have multiple database server processes running on different ports. This command stops the Java DB process for the specified port only.

1 **Notify users that the database is being stopped.**

2 **Stop the database by using the `stop-database`(1) command.**

Example 4–2 Stopping a Database

The following example command stops Java DB on `host1`, port 5001:

```
stop-database --dbhost host1 --dbport 5001
```

See Also To see the full syntax of the command, type `asadmin stop-database --help` at the command line.

Managing JDBC Connection Pools

A JDBC connection pool is a group of reusable connections for a particular database. When creating the pool, you are defining the aspects of a connection to a specific database.

The following tasks and information are used to manage JDBC connection pools:

- "Settings for a JDBC Connection Pool" on page 59
- "To Create a JDBC Connection Pool" on page 63
- "To List JDBC Connection Pools" on page 64
- "To Contact (Ping) a Connection Pool" on page 65
- "To Delete a JDBC Connection Pool" on page 65

Settings for a JDBC Connection Pool

There are a number of types of settings that you can adjust for a given connection pool. You can change all the settings except the name of the connection pool.

The following settings are explained here:

- "General Settings" on page 59
- "Pool Settings" on page 60
- "Connection Validation Settings" on page 60
- "Transaction Isolation Settings" on page 61
- "Properties Settings" on page 62
- "Advanced Attributes" on page 62
- "Advanced Connection Settings" on page 62

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.

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

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; when the application releases the connection, the connection is returned to the pool.

Parameter	Description
Initial and Minimum Pool Size	The minimum number of connections in the pool. This value also determines the number of connections placed in the pool when the pool is first created or when the Application Server starts.
Maximum Pool Size	The maximum number of connections in the pool.
Pool Resize Quantity	When the pool scales up and scales down towards the maximum and minimum pool sizes respectively, it is resized in batches. This value determines the number of connections in the batch. Making this value too large delays connection creation and recycling, while 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 Settings

Optionally, the Application Server can validate connections before they are passed to applications. This validation allows the Application Server to automatically reestablish database connections if the database becomes unavailable due to network failure or database server crash.

Note – Validation of connections incurs additional overhead and slightly reduces performance.

Parameter	Description
Connection Validation	Select the Required checkbox to enable connection validation.
Validation Method	The Application Server can validate database connections by using the following methods: auto-commit, metadata, and table. auto-commit and metadata - The Application Server validates a connection by calling the `con.getAutoCommit()` and `con.getMetaData()` methods. **Note –** 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 as the Validation Method, specify the name of the database table here.
On Any Failure	If you select Close All Connections, then the Application Server closes all connections in the pool and reestablishes them if a single connection fails. If you do not select Close All Connections, then individual connections are reestablished only when they are used.
Allow Non Component Callers	Select to enable the pool for use by non-component callers such as servlet filters and lifecycle modules.

Transaction Isolation Settings

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

Parameter	Description
Non-transactional Connections	Select if you want the Application Server to return all non-transactional connections.
Transaction Isolation	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.

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

Properties Settings

In the Additional Properties table, you can specify database 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.

Advanced Attributes

The following attributes can be used to configure a connection pool at the time of its creation.

Attribute	Description
Name	Name of the JDBC connection pool whose properties you want to edit. You cannot change the pool name.
Statement Timeout	Time in seconds after which abnormally long running queries will be terminated. The Application Server will set "QueryTimeout" on the statements created. The default value of -1 means that the attribute is not enabled.
Wrap JDBC Objects	When set to true, the application will get wrapped JDBC objects for Statement, `PreparedStatement`, `CallableStatement`, `ResultSet`, `DatabaseMetaData`. The default value is false.

Advanced Connection Settings

Specify the Connection Settings as explained in the following table.

Attribute	Description
Validate Atmost Once	Amount of time, in seconds, after which a connection is validated at most once. This will help reduce the number of validation requests by a connection. The default value of 0 means that connection validation is not enabled.

Leak Timeout	Amount of time, in seconds, to trace connection leaks in a connection pool. The default value of 0 means that connection leak tracing is disabled. If connection leak tracing is enabled, you can get statistics on the number of connection leaks in the Monitoring Resources tab. To view this tab, go to Application Server > Monitoring > Resources.
Leak Reclaim	If this option is enabled, leaked connections will be restored to the pool after leak connection tracing is complete.
Creation Retry Attempts	Number of attempts that will be made if there is a failure in creating a new connection. The default value of 0 means that no attempts will be made to recreate the connection.
Retry Interval	Specifies the interval, in seconds, between two attempts to create a connection. The default value is 10 seconds. This attribute is used only if the value of Creation Retry Attempts is greater than 0.
Lazy Connection Enlistment	Enlists a resource to the transaction only when the resource is actually used in a method.
Lazy Association	Connections are lazily associated when an operation is performed on them, and are disassociated when the transaction is completed and a component method ends. This helps efficient reuse of the physical connections. Default value is false.
Associate with Thread	Enable this option to associate a connection with the thread such that when the same thread is in need of a connection, it can reuse the connection already associated with that thread, thereby not incurring the overhead of getting a connection from the pool. Default value is false.
Match Connections	Switches connection matching for the pool to on or off. Can be set to false if you know that the connections in the pool will always be homogeneous and, therefore, a connection picked from the pool need not be matched by the resource adapter. Default value is false.
Max Connection Usage	Specifies the number of times a connection should be reused by the pool. After a connection is reused for the specified number of times, the connection will be closed. This is useful, for instance, to avoid statement-leaks. The default value of 0 means that no connections will be reused.

▼ To Create a JDBC Connection Pool

The remote `create-jdbc-connection-pool` command enables you to register a new JDBC connection pool with the specified JDBC connection pool name. A JDBC connection pool or a connector connection pool with authentication can be created. You can either use a command option to specify user, password, or other connection information using the `asadmin` utility, or specify the connection information in the XML descriptor file.

Creating a JDBC connection pool is a dynamic event and does not require server restart.

Before You Begin When you are building the connection pool, certain data specific to the JDBC driver and the database vendor must be entered.

- 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

You can find some of these specifics in"Configuration Specifics for JDBC Drivers" on page 69.

Before creating the connection pool, you must first install and integrate the database and its associated JDBC driver. For instructions, see "Setting Up Database Access" on page 57.

1 **Ensure that the server is running.**

Remote commands require a running server.

2 **Create the JDBC connection pool by using the `create-jdbc-connection-pool`(1) command.**

Example 4–3 Creating a JDBC Connection Pool

The following example command creates a JDBC connection pool named `sample_derby_pool` on the local host:

```
asadmin create-jdbc-connection-pool --user admin
--passwordfile passwords.txt --host localhost --port 7070
--datasourceclassname org.apache.derby.jdbc.ClientDataSource
--restype javax.sql.XADataSource
--property portNumber=1527:password=APP:user=APP:serverName=
localhost:databaseName=sun-appserv-samples:connectionAttributes=\;
create\\=true sample_derby_pool
```

See Also To see the full syntax of the command, type `asadmin create-jdbc-connection-pool --help` at the command line.

▼ To List JDBC Connection Pools

The remote `list-jdbc-connection-pools` command enables you to list all existing JDBC connection pools.

1 **Ensure that the server is running.**

Remote commands require a running server.

2 **List the JDBC connection pools by using the `list-jdbc-connection-pools`(1) command.**

Example 4–4 Listing JDBC Connection Pools

The following example command lists the JDBC connection pools that are on the local host:

```
asadmin list-jdbc-connection-pool --user admin --passwordfile password.txt
--host localhost
```

See Also To see the full syntax of the command, type `asadmin list-jdbc-connection-pools --help` at the command line.

▼ To Contact (Ping) a Connection Pool

The remote `ping-connection-pool` command tests if a JDBC connection pool is usable.

Before You Begin Before you can contact a connection pool, the connection pool must be created with authentication, and the server or database must be running.

1. **Ensure that the server is running.**

   Remote commands require a running server.

2. **Ping a connection pool by using the `ping-connection-pool`(1) command.**

Example 4–5 Contacting a Connection Pool

The following example command asks if the `sample_javadb_pool` connection pool is usable:

```
asadmin ping-jdbc-connection-pool --user admin1
--passwordfile password.txt sample_javadb_pool
```

See Also To see the full syntax of the command, type `asadmin ping-connection-pool --help` at the command line.

▼ To Delete a JDBC Connection Pool

The remote `delete-jdbc-connection-pool` command enables you to delete an existing JDBC connection pool. Deleting a JDBC connection pool is a dynamic event and does not require server restart.

Before You Begin Before deleting a JDBC connection pool, all associations to the resource must be removed.

1. **Ensure that the server is running.**

   Remote commands require a running server.

2 **Obtain the exact name of the JDBC connection pool that you are deleting.**

To list the existing JDBC connection pools:

```
asadmin list-jdbc-connection-pool
```

3 **Notify users that the JDBC connection pool is being deleted.**

4 **Delete the connection pool by using the `delete-jdbc-connection-pool`(1) command.**

Example 4–6 Deleting a JDBC Connection Pool

The following example command deletes a connection pool named `sample_javadb_pool`:

```
asadmin delete-jdbc-connection-pool --user admin1
--passwordfile password.txt sample_javadb_pool
```

See Also To see the full syntax of the command, type `asadmin delete-jdbc-connection-pool --help` at the command line.

Managing JDBC Resources

High-level steps for creating a JDBC resource include the following:

1. Identifying the JNDI name
2. Selecting a connection pool to be associated with the new JDBC resource
3. Specifying the settings for the resource
4. Identifying the target (server instance) on which the resource will be available

The following tasks and information are used to manage JDBC resources:

- "Command Options for Managing JDBC Resources" on page 66
- "To Create a JDBC Resource" on page 67
- "To List JDBC Resources" on page 68
- "To Delete a JDBC Resource" on page 68

Command Options for Managing JDBC Resources

Options for managing JDBC resources include the following:

JNDI Name	The unique JNDI name organizes and locates components within a distributed computing environment in a similar manner to how a card catalog organizes the books in a library. This JNDI method is a crucial method for accessing a JDBC resource. By convention, the name begins with the `jdbc/` string. For example, `jdbc/payrolldb`.

`--connectionpoolid`	Specifies the connection pool to be associated with the new JDBC resource.
`--description`	Briefly describes the JDBC resource.
`--enabled`	Determines whether the JDBC resource is enabled at runtime. Default value is true.
`--property`	Optional attribute name/value pairs for configuring the resource.

For details on these and additional options, see `create-jdbc-resource`(1) in the reference manual.

▼ To Create a JDBC Resource

The remote `create-jdbc-resource` command enables you to create a JDBC resource. Creating a JDBC resource is a dynamic event and does not require server restart.

Before You Begin Before creating a JDBC resource, you must first create a JDBC connection pool. For instructions, see "To Create a JDBC Connection Pool" on page 63.

1. **Ensure that the server is running.**

   Remote commands require a running server.

2. **Create a JDBC resource by using the `create-jdbc-resource`(1) command.**

3. **Notify users that the a new resource has been created.**

Example 4–7 Creating a JDBC Resource

The following example command creates a JDBC resource named `jdbc/DerbyPool`:

```
asadmin create-jdbc-resource --user admin1 --passwordfile passwords.txt jdbc/DerbyPool
```

See Also To see the full syntax of the command, type `asadmin create-jdbc-resource --help` at the command line.

▼ To List JDBC Resources

The remote `list-jdbc-resources` command enables you to list the existing JDBC resources.

1 **Ensure that the server is running.**
Remote commands require a running server.

2 **List JDBC resources by using the `list-jdbc-resources`(1) command.**

Example 4–8 Listing JDBC Resources

The following example command lists JDBC resources for the local host:

```
asadmin list-jdbc-resources --user admin1 --passwordfile password.txt
```

See Also To see the full syntax of the command, type `asadmin list-jdbc-resources --help` at the command line.

▼ To Delete a JDBC Resource

This remote command enables you to delete an existing JDBC resource. Deleting a JDBC resource is a dynamic event and does not require server restart.

Before You Begin Before deleting a JDBC resource, all associations with this resource must be removed.

1 **Ensure that the server is running.**
Remote commands require a running server.

2 **Obtain the exact name of the JDBC resource that you are deleting.**
To list the existing JDBC resources:
```
asadmin list-jdbc-resources
```

3 **Notify users that the JDBC resource is being deleted.**

4 **Delete a JDBC resource by using the `delete-jdbc-resource`(1) command.**

Example 4–9 Deleting a JDBC Resource

The following example command deletes a JDBC resource named `jdbc/DerbyPool`:

```
asadmin delete-jdbc-resource --user admin1
--passwordfile jdbc/DerbyPool
```

See Also To see the full syntax of the command, type `asadmin delete-jdbc-resource --help` at the command line.

Configuration Specifics for JDBC Drivers

The Application Server is designed to support connectivity to any database management system by using a corresponding JDBC driver. The following JDBC driver and database combinations have been tested, found to be J2EE compatible, and are supported for container-managed persistence:

- "GlassFish JDBC Driver for DB2 Databases" on page 70
- "GlassFish JDBC Driver for Oracle 8.1.7 and 9.x Databases" on page 70
- "GlassFish JDBC Driver for Microsoft SQL Server Databases" on page 71
- "GlassFish JDBC Driver for Sybase Databases" on page 71
- "IBM DB2 8.1 Type 2 Driver" on page 72
- "Java DB/Derby Type 4 Driver" on page 72
- "JConnect Type 4 Driver for Sybase ASE 12.5 Databases" on page 73
- "MM MySQL Type 4 Driver (Non-XA)" on page 74

To see the most current list of supported JDBC drivers, refer to the *GlassFish v3 Application Server Release Notes*.

The following JDBC drivers can also be used with the Application Server, but J2EE compliance tests have not been completed with these drivers. Although Sun offers no product support for these drivers, Sun does offer limited support for the use of these drivers with the Application Server:

- "MM MySQL Type 4 Driver (XA Only)" on page 74
- "Inet Oraxo JDBC Driver for Oracle 8.1.7 and 9.x Databases" on page 75
- "Inet Merlia JDBC Driver for Microsoft SQL Server Databases" on page 76
- "Inet Sybelux JDBC Driver for Sybase Databases" on page 76
- "Oracle Thin Type 4 Driver for Oracle 8.1.7 and 9.x Databases" on page 77
- "OCI Oracle Type 2 Driver for Oracle 8.1.7 and 9.x Databases" on page 78
- "IBM Informix Type 4 Driver" on page 79
- "CloudScape 5.1 Type 4 Driver" on page 79

Note – An Oracle database user running the `capture-schema` command needs ANALYZE ANY TABLE privileges if that user does not own the schema. These privileges are granted to the user by the database administrator. For information about `capture-schema`, see *GlassFish v3 Application Server Reference Manual*.

GlassFish JDBC Driver for DB2 Databases

The JAR files for this driver are `smbase.jar`, `smdb2.jar`, and `smutil.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `DB2`
- **DataSource Classname:** `com.sun.sql.jdbcx.db2.DB2DataSource`
- **Properties:**
  - **`serverName`** - Specify the host name or IP address of the database server.
  - **`portNumber`** - Specify the port number of the database server.
  - **`databaseName`** - Set as appropriate.
  - **`user`** - Set as appropriate.
  - **`password`** - Set as appropriate.
- **URL:** `jdbc:sun:db2://`*serverName*`:`*portNumber*`;databaseName=`*databaseName*

GlassFish JDBC Driver for Oracle 8.1.7 and 9.x Databases

The JAR files for this driver are `smbase.jar`, `smoracle.jar`, and `smutil.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `Oracle`
- **DataSource Classname:** `com.sun.sql.jdbcx.oracle.OracleDataSource`
- **Properties:**
  - **`serverName`** - Specify the host name or IP address of the database server.
  - **`portNumber`** - Specify the port number of the database server.
  - **`SID`** - Set as appropriate.
  - **`user`** - Set as appropriate.

- **password** - Set as appropriate.
- **URL:** `jdbc:sun:oracle://`*serverName*`[:`*portNumber*`][;SID=`*databaseName*`]`

GlassFish JDBC Driver for Microsoft SQL Server Databases

The JAR files for this driver are `smbase.jar`, `smsqlserver.jar`, and `smutil.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `mssql`
- **DataSource Classname:** `com.sun.sql.jdbcx.sqlserver.SQLServerDataSource`
- **Properties:**
  - **serverName** - Specify the host name or IP address and the port of the database server.
  - **portNumber** - Specify the port number of the database server.
  - **user** - Set as appropriate.
  - **password** - Set as appropriate.
  - **selectMethod** - Set to `cursor`.
- **URL:** `jdbc:sun:sqlserver://`*serverName*`[:`*portNumber*`]`

GlassFish JDBC Driver for Sybase Databases

The JAR files for this driver are `smbase.jar`, `smsybase.jar`, and `smutil.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `Sybase`
- **DataSource Classname:** `com.sun.sql.jdbcx.sybase.SybaseDataSource`
- **Properties:**
  - **serverName** - Specify the host name or IP address of the database server.
  - **portNumber** - Specify the port number of the database server.
  - **databaseName** - Set as appropriate. This is optional.
  - **user** - Set as appropriate.
  - **password** - Set as appropriate.

- **URL:** `jdbc:sun:sybase://`*serverName*`[:`*portNumber*`]`

IBM DB2 8.1 Type 2 Driver

The JAR files for the DB2 driver are `db2jcc.jar`, `db2jcc_license_cu.jar`, and `db2java.zip`. Set environment variables as follows:

```
LD_LIBRARY_PATH=/usr/db2user/sqllib/lib:${j2ee.home}/lib
DB2DIR=/opt/IBM/db2/V8.1
DB2INSTANCE=db2user
INSTHOME=/usr/db2user
VWSPATH=/usr/db2user/sqllib
THREADS_FLAG=native
```

Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `DB2`
- **DataSource Classname:** `com.ibm.db2.jcc.DB2SimpleDataSource`
- **Properties:**
  - **`databaseName`** - Set as appropriate.
  - **`user`** - Set as appropriate.
  - **`password`** - Set as appropriate.
  - **`driverType`** - Set to `2`.
  - **`deferPrepares`** - Set to `false`.

Java DB/Derby Type 4 Driver

The JAR file for the Java DB/Derby driver is `derbyclient.jar`. (Java DB is based upon Apache Derby.) Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `Java DB/Derby`
- **DataSource Classname:** Specify one of the following:

  ```
  org.apache.derby.jdbc.ClientDataSource
  org.apache.derby.jdbc.ClientXADataSource
  ```

- **Properties:**

- **`serverName`** - Specify the host name or IP address of the database server.
- **`portNumber`** - Specify the port number of the database server if it is different from the default.
- **`databaseName`** - Specify the name of the database.
- **`user`** - Specify the database user.

  This is only necessary if Derby is configured to use authentication. Derby does *not* use authentication by default. When the user is provided, it is the name of the schema where the tables reside.
- **`password`** - Specify the database password.

  This is only necessary if Java DB/Derby is configured to use authentication.

- **URL:** `jdbc:derby://`*serverName*`:`*portNumber*`/`*databaseName*`;create=true`

  Include the `;create=true` part only if you want the database to be created if it does not exist.

JConnect Type 4 Driver for Sybase ASE 12.5 Databases

The JAR file for the Sybase driver is `jconn2.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `Sybase`
- **DataSource Classname:**Specify one of the following:

  ```
  com.sybase.jdbc2.jdbc.SybDataSource
  com.sybase.jdbc2.jdbc.SybXADataSource
  ```

- **Properties:**
  - **`serverName`** - Specify the host name or IP address of the database server.
  - **`portNumber`** - Specify the port number of the database server.
  - **`databaseName`** - Set as appropriate. Do not specify the complete URL, only the database name.
  - **`user`** - Set as appropriate.
  - **`password`** - Set as appropriate.
  - **`BE_AS_JDBC_COMPLIANT_AS_POSSIBLE`** - Set to `true`.
  - **`FAKE_METADATA`** - Set to `true`.

MM MySQL Type 4 Driver (Non-XA)

The JAR file for the MySQL driver is `mysql-connector-java-`*version*`-bin-g.jar`, for example, `mysql-connector-java-3.1.12-bin-g.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `mysql`
- **DataSource Classname:** Specify one of the following:

  ```
  com.mysql.jdbc.jdbc2.optional.MysqlDataSource
  ```

- **Properties:**
  - **`serverName`** - Specify the host name or IP address of the database server.
  - **`portNumber`** - Specify the port number of the database server.
  - **`databaseName`** - Set as appropriate.
  - **`user`** - Set as appropriate.
  - **`password`** - Set as appropriate.
  - **`URL`** - If you are using global transactions, you can set this property instead of `serverName`, `port`, and `databaseName`.

    The MM MySQL Type 4 driver doesn't provide a method to set the required `relaxAutoCommit` property, so you must set it indirectly by setting the **`URL`** property:

    ```
    jdbc:mysql://host:port/database?relaxAutoCommit="true"
    ```

MM MySQL Type 4 Driver (XA Only)

The JAR file for the MySQL driver is `mysql-connector-java-`*version*`-bin-g.jar`, for example, `mysql-connector-java-3.1.12-bin-g.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `mysql`
- **DataSource Classname:** Specify one of the following:

  ```
  com.mysql.jdbc.jdbc2.optional.MysqlXADataSource
  ```

- **Properties:**

- **`serverName`** - Specify the host name or IP address of the database server.
- **`portNumber`** - Specify the port number of the database server.
- **`databaseName`** - Set as appropriate.
- **`user`** - Set as appropriate.
- **`password`** - Set as appropriate.
- **`URL`** - If you are using global transactions, you can set this property instead of `serverName`, `port`, and `databaseName`.

  The MM MySQL Type 4 driver doesn't provide a method to set the required `relaxAutoCommit` property, so you must set it indirectly by setting the **`URL`** property:

  ```
  jdbc:mysql://host:port/database?relaxAutoCommit="true"
  ```

Inet Oraxo JDBC Driver for Oracle 8.1.7 and 9.x Databases

The JAR file for the Inet Oracle driver is `Oranxo.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `Oracle`
- **DataSource Classname:** `com.inet.ora.OraDataSource`
- **Properties:**
  - **`serverName`** - Specify the host name or IP address of the database server.
  - **`portNumber`** - Specify the port number of the database server.
  - **`user`** - Specify the database user.
  - **`password`** - Specify the database password.
  - **`serviceName`** - Specify the URL of the database. The syntax is as follows:

    ```
    jdbc:inetora:server:port:dbname
    ```

    For example:

    ```
    jdbc:inetora:localhost:1521:payrolldb
    ```

    In this example,`localhost` is the name of the host running the Oracle server, `1521` is the Oracle server's port number, and `payrolldb` is the SID of the database. For more information about the syntax of the database URL, see the Oracle documentation.

- **`streamstolob`** - If the size of BLOB or CLOB data types exceeds 4 KB and this driver is used for CMP, this property must be set to `true`.
- **`xa-driver-does-not-support-non-tx-operations`** - Set to the value `true`. Optional: only needed if both non-XA and XA connections are retrieved from the same connection pool. Might degrade performance.

  As an alternative to setting this property, you can create two connection pools, one for non-XA connections and one for XA connections.

Inet Merlia JDBC Driver for Microsoft SQL Server Databases

The JAR file for the Inet Microsoft SQL Server driver is `Merlia.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `mssql`
- **DataSource Classname:** `com.inet.tds.TdsDataSource`
- **Properties:**
  - **`serverName`** - Specify the host name or IP address and the port of the database server.
  - **`portNumber`** - Specify the port number of the database server.
  - **`user`** - Set as appropriate.
  - **`password`** - Set as appropriate.

Inet Sybelux JDBC Driver for Sybase Databases

The JAR file for the Inet Sybase driver is `Sybelux.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `Sybase`
- **DataSource Classname:** `com.inet.syb.SybDataSource`
- **Properties:**
  - **`serverName`** - Specify the host name or IP address of the database server.
  - **`portNumber`** - Specify the port number of the database server.

- **`databaseName`** - Set as appropriate. Do not specify the complete URL, only the database name.
- **`user`** - Set as appropriate.
- **`password`** - Set as appropriate.

Oracle Thin Type 4 Driver for Oracle 8.1.7 and 9.x Databases

The JAR file for the Oracle driver is `ojdbc14.jar`.

Note – When using this driver, it is not possible to insert more than 2000 bytes of data into a column. To circumvent this problem, use the OCI driver (JDBC type 2).

Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `Oracle`
- **DataSource Classname:**Specify one of the following:

  ```
  oracle.jdbc.pool.OracleDataSource
  oracle.jdbc.xa.client.OracleXADataSource
  ```

- **Properties:**
  - **`user`** - Set as appropriate.
  - **`password`** - Set as appropriate.
  - **`URL`** - Specify the complete database URL using the following syntax:

    ```
    jdbc:oracle:thin:[user/password]@host[:port]/service
    ```

    For example:

    ```
    jdbc:oracle:thin:@localhost:1521:customer_db
    ```

  - **`xa-driver-does-not-support-non-tx-operations`** - Set to the value `true`. Optional: only needed if both non-XA and XA connections are retrieved from the same connection pool. Might degrade performance.

    As an alternative to setting this property, you can create two connection pools, one for non-XA connections and one for XA connections.

Note – For the Oracle thin driver, the `XAResource.recover` method repeatedly returns the same set of in-doubt Xids regardless of the input flag. According to the XA specifications, the Transaction Manager initially calls this method with TMSTARTSCAN and then with TMNOFLAGS repeatedly until no Xids are returned. The `XAResource.commit` method also has some issues.

To disable this Application Server workaround, the `oracle-xa-recovery-workaround` property value must be set to `false`.

OCI Oracle Type 2 Driver for Oracle 8.1.7 and 9.x Databases

The JAR file for the OCI Oracle driver is `ojdbc14.jar`. Make sure that the shared library is available through `LD_LIBRARY_PATH` and that the `ORACLE_HOME` property is set. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `Oracle`
- **DataSource Classname:**Specify one of the following:

  ```
  oracle.jdbc.pool.OracleDataSource
  oracle.jdbc.xa.client.OracleXADataSource
  ```

- **Properties:**
  - **`user`** - Set as appropriate.
  - **`password`** - Set as appropriate.
  - **`URL`** - Specify the complete database URL using the following syntax:

    ```
    jdbc:oracle:oci:[user/password]@host[:port]/service
    ```

    For example:

    ```
    jdbc:oracle:oci:@localhost:1521:customer_db
    ```

  - **`xa-driver-does-not-support-non-tx-operations`** - Set to the value `true`. Optional: only needed if both non-XA and XA connections are retrieved from the same connection pool. Might degrade performance.

    As an alternative to setting this property, you can create two connection pools, one for non-XA connections and one for XA connections.

IBM Informix Type 4 Driver

Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `Informix`
- **DataSource Classname:**Specify one of the following:

  ```
  com.informix.jdbcx.IfxDataSource
  com.informix.jdbcx.IfxXADataSource
  ```

- **Properties:**
  - **`serverName`** - Specify the Informix database server name.
  - **`portNumber`** - Specify the port number of the database server.
  - **`databaseName`** - Set as appropriate. This is optional.
  - **`user`** - Set as appropriate.
  - **`password`** - Set as appropriate.
  - **`IfxIFXHost`** - Specify the host name or IP address of the database server.

CloudScape 5.1 Type 4 Driver

The JAR files for the CloudScape driver are `db2j.jar`, `db2jtools.jar`, `db2jcview.jar`, `jh.jar`, `db2jcc.jar`, and `db2jnet.jar`. Configure the connection pool using the following settings:

- **Name:** Use this name when you configure the JDBC resource later.
- **Resource Type:** Specify the appropriate value.
- **Database Vendor:** `Cloudscape`
- **DataSource Classname:** `com.ibm.db2.jcc.DB2DataSource`
- **Properties:**
  - **`user`** - Set as appropriate.
  - **`password`** - Set as appropriate.
  - **`databaseName`** - Set as appropriate.

◆ ◆ ◆ CHAPTER 5

Administering System Security

This chapter provides instructions for administering system security in the GlassFish Application Server environment by using the `asadmin` command-line utility.

The following topics are addressed here:

Instructions for accomplishing these tasks by using the Admin Console are contained in the Admin Console online help.

Additional information on security is contained in Chapter 6, "Administering User Security,"

About Application Server Security

Security is about protecting data, that is, how to prevent unauthorized access or damage to data that is in storage or in transit. The GlassFish Application Server provides a dynamic, extensible security architecture based on the Java EE standard. 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. Built-in security features include cryptography, authentication and authorization, and public key infrastructure.

The following topics are addressed here:

Difference Between System Security and Application Security

There are two types of security that apply to a software environment: system security and application security.

System security affects all the applications on the Application Server. The material in this document is intended primarily for system administrators, and so focuses on system security.

Application security affects a particular application. There are basically two types of application security: programmatic and declarative.

- 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 programmatic security, application code written handles security chores. As an administrator, you do not 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. Programmatic security is controlled by the application developer.

Information on application security is contained in the Chapter 3, "Securing Applications," in *GlassFish v3 Application Server Developer's Guide*.

Tools for Managing System Security

The Application Server provides the following tools for managing system security:

Admin Console	The Admin Console is a browser-based utility used to configure security for the entire server. Tasks include managing certificates, users, groups, and realms, and performing other system-wide security tasks. For a general introduction to the Admin Console, see "Admin Console" on page 27.
The `asadmin` utility	The `asadmin` command-line utility performs many of the same tasks as the Admin Console. You might be able to do some things with the `asadmin` utility that you cannot do with the Admin Console. For a general introduction to `asadmin`, see "Command-Line Utility" on page 28.
The `keytool` utility	The `keytool` Java 2 Platform, Standard Edition (J2SE) command-line utility is used for managing digital certificates and

	key pairs. Use `keytool` to manage users in the `certificate` realm. For more information, see "Administering JSSE Certificates Using the `keytool` Utility" on page 88.
The `policytool` utility	The `policytool` J2SE graphical utility is used for managing system-wide Java security policies. As an administrator, you rarely 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`.

Passwords

The following topics are addressed here:

Master Password and Keystores

The master password is the password for the secure keystore. When a new application server domain is created, a new self-signed certificate is generated and stored in the relevant keystore, which is locked using the master password. If the master password is not the default, you are prompted for the master password. (The default password is `changeit`.) After the correct master password is entered, the domain starts.

Encoded Passwords

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

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

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

Authentication and Authorization

The following topics are addressed here:

Authentication Methods

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 might be a user name and password, a digital certificate, or something else. Usually, servers or applications require clients to authenticate. Additionally, clients might require servers to authenticate themselves. 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.

Within its deployment descriptors, an application specifies the type of authentication it uses. The Application Server supports the following types of authentication:

BASIC	Uses the server's built-in login dialog box. The communication protocol is HTTP (SSL optional). There is no user-credentialed encryption unless using SSL.
FORM	The application provides its own custom login and error pages. The communication protocol is HTTP (SSL optional). There is no user-credentialed encryption unless using SSL.
CLIENT-CERT	The server authenticates the client using a public key certificate. The communication protocol is HTTPS (HTTP over SSL). User-credentialed encryption is SSL.

Single Sign-On

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 enables multiple applications in one virtual server instance to share the user authentication state.

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, or CLIENT-CERT) share single sign-on.

For the Application Server, single sign-on is enabled by default for virtual servers.

User Authorization

After a user is authenticated, the level of *authorization* determines what operations the owner can perform. A user's authorization is based on his role. For more information on roles, see "Roles" on page 94.

Java Authorization Contract for Containers (JACC) is the part of the Java EE specification that defines an interface for pluggable authorization providers. This enables you to set up third-party plug-in modules to perform authorization. By default, the Application Server provides a simple, file-based authorization engine that complies with the JACC specification. You can also 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. JAAS implements a Java technology version of the standard Pluggable Authentication Module (PAM) framework.

Audit Trails

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.

Firewall Guidelines

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. The following guidelines pertain primarily to the Application Server:

- 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 set up 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 is required, open the RMI-IIOP listener port as well.

  Note – Opening the RMI-IIOP listener port 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.

For details about a specific firewall technology, refer to the documentation from the firewall vendor.

Certificates and SSL

The following topics are addressed here:

Certificates

Certificates, also called digital 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:

- *Personal certificates* are used by individuals.
- *Server certificates* are used to establish secure sessions between the server and clients through secure sockets layer (SSL) technology.

Certificates are based on *public key cryptography*, which uses pairs of digital keys (very long numbers) to encrypt, or encode, information so the information 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, which is always kept secret. Because the keys are mathematically related, data encrypted with one key can only be decrypted with the other key in the pair.

Certificates are issued by a trusted third party called a *Certification Authority* (CA). The CA is analogous to a passport office: it validates the certificate holder's identity and signs the certificate so that it cannot be forged or tampered with. After 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.

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

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

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

Certificate Chains

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

Certificate Files

During Application Server installation, a certificate is generated in Java Secure Socket Extension (JSSE) format suitable for internal testing. By default, the Application Server stores its certificate information in certificate databases in the *domain-dir*`/config` directory:

Keystore file
: The `key3.db` file contains the Application Server certificate, including its private key. The keystore file is protected with a password. You can change the password using the `asadmin change-master-password` command.

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

Truststore file
: The `cert8.db` file contains the Application Server 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 CAs.

By default, the Application Server is configured with a keystore and truststore that will work with the example applications and for development purposes.

Secure Sockets Layer

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

The newest version of the SSL standard is called Transport Layer Security (TLS). The Application Server supports the SSL 3.0 and the 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 certificate has been installed.

To set up a digital certificate that your Web server can use for SSL, see "Generating a Certificate" on page 90.

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.

Setting Passwords From a File

For security purposes, you can set the password for a subcommand from a file instead of entering the password at the command line. The `--passwordfile` option takes the name of the file containing the passwords. The valid contents for the file are:

```
AS_ADMIN_PASSWORD=value
AS_ADMIN_ADMINPASSWORD=value
AS_ADMIN_USERPASSWORD=value
AS_ADMIN_MASTERPASSWORD=value
```

If `AS_ADMIN_PASSWORD` has been exported to the global environment, specifying the `--passwordfile` option will produce a warning about using the `--password` option. To prevent this warning situation from happening, unset `AS_ADMIN_PASSWORD`. The master password is not propagated on the command line or by using an environment variable, but can be specified by using the `--passwordfile` option on the command line.

To use the `--secure` option, you must use the `set` command to enable security (by using the `--enabled` option) in the `admin` `http-listener` property in the `domain.xml` file, then restart the server.

Administering JSSE Certificates Using the `keytool` Utility

Use the `keytool` utility to set up and work with Java Secure Socket Extension (JSSE) digital certificates.The J2SE SDK ships with the `keytool` utility, which enables you administer public/private key pairs and associated certificates. The utility also enables users to cache the public keys (in the form of certificates) of their communicating peers.

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

The following topics are addressed here:

Basic Certificate Administration

The following examples demonstrate how to administer certificates by using the keytool utility:

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

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

  Another example of creating a certificate is shown in "Generating a Certificate" on page 90.

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

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

  An example of signing a certificate is shown in "Signing a Certificate" on page 91

- Display available certificates from a keystore of type JKS.

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

- Display certificate information from a keystore of type JKS.

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

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

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

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

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

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

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

- Delete a certificate from a keystore of type JKS.

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

  Another example of deleting a certificate from a keystore is shown in "Deleting a Certificate" on page 92.

Generating a Certificate

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

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

   Always generate the certificate in the directory containing the keystore and truststore files, by default *domain-dir*`/config`.

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

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

Use any unique name as your *keyAlias*. If you have changed the keystore or private key password from their default, then substitute the new password for `changeit` in the above command. The default key password alias is "s1as."

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

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

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

4. If a certificate signed by a certificate authority is required, see "Signing a Certificate" on page 91.
5. To create the `cacerts.jks` truststore file and add the certificate to the truststore, enter the following `keytool` command:

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

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

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

   Then the `keytool` utility displays something like this:

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

8. Restart the Application Server.

Signing a Certificate

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

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

   Save the certificate in the directory containing the keystore and truststore files, by default *domain-dir*/config directory.
3. In your shell, change to the directory containing the certificate.
4. Import the certificate into the local keystore and, if necessary, the local truststore. For example:

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

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

Deleting a Certificate

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

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

◆ ◆ ◆ CHAPTER 6

Administering User Security

This chapter provides instructions for administering user security in the GlassFish Application Server environment by using the `asadmin` command-line utility. This chapter assumes that you are familiar with security features such as authentication, authorization, and certificates. Additional information on security is contained in Chapter 5, "Administering System Security."

The following topics are addressed here:

Instructions for accomplishing these tasks by using the Admin Console are contained in the Admin Console online help.

About User Security

The GlassFish Application Server enforces its authentication and authorization policies upon users, groups, roles, and realms. Users and groups are user identities that are designated for the entire Application Server, whereas each application defines its own roles. A realm is a repository where the server stores user and group information.

The following topics are addressed here:

Users and Groups

A *user* is an individual (or application program) identity that has been defined in the Application Server. A user can be associated with multiple groups. A user who has been authenticated is sometimes called a principal.

A *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 might belong to the `preferred` group. Categorizing users into groups makes it easier to control the access of large numbers of users. A group is defined for an entire server or realm.

Roles

A *role* defines which applications and what parts of each application users can access and what users 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 have access to salary information. This application must 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 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`. Users in these groups are also 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. When the application is being packaged and deployed, the application specifies mappings between users/groups and roles, as illustrated in the following figure.

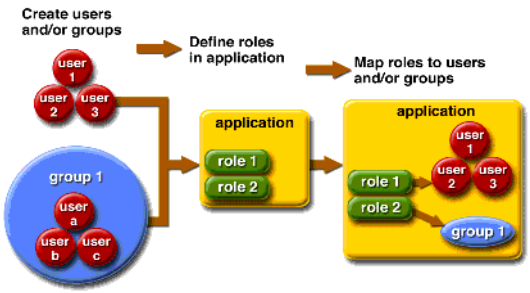

FIGURE 6–1 User Role Mapping

Realms

An *authentication realm*, also called a security policy domain or security domain, is a scope over which the Application Server defines and enforces a common security policy.

The Application Server is preconfigured with the `file`, `certificate`, and `admin-realm` realms. In addition, you can set up `ldap`, `JDCB`, `solaris`, or custom realms. An application can specify which realm to use in its deployment descriptor. If the application does not specify a realm, the Application Server uses its default realm.

`file` realm	The Application Server stores user credentials locally in a file named `keyfile`. You can use the Admin Console or the `asadmin` commands to manage users in the `file` realm. The `file` realm is the initial default realm.
`certificate` realm	The Application 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 "Certificates and SSL" on page 86.
`admin-realm` realm	The `admin-realm` is also a `file` realm and stores administrator user credentials locally in a file named `admin-keyfile`. You can manage users in the `admin-realm` realm in the same way you manage users in the `file` realm.
`ldap` realm	The Application Server gets user credentials from a Lightweight Directory Access Protocol (LDAP) server such as the GlassFish 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.
`JDBC` realm	The Application Server gets user credentials from a database. The server uses the database information and the enabled JDBC realm option in the configuration file.
`solaris` realm	The Application Server gets user credentials from the Solaris operating system. This realm is supported on the Solaris 9 and Solaris 10 operating systems. Consult your Solaris documentation for information on managing users and groups in the `solaris` realm.
Custom realm	You can create other repositories for user credentials, such as a relational database or third-party components. For more information on custom realms, see the Admin Console online help.

The Application Server authentication service can govern users in multiple realms.

Managing File Users

As the administrator, you are responsible for integrating users into the Application Server environment so that their credentials are securely established and they are provided with access to the applications and services that they are entitled to use.

The following tasks and information are used to manage users:

Command Options for Managing File Users

Options for managing file users include the following:

User Name	Unique user name for the file user.
`--groups`	The groups associated with this file user.
`--authrealmname`	The file where the file users are stored.

For details on these and additional options, see `create-file-user`(1) in the reference manual.

▼ To Create a File User

The remote `create-file-user` command enables you to create a new user by adding a new entry to the `keyfile`. The entry includes the username, password, and groups for the user. Multiple groups can be specified by separating the groups with colons (:).

Creating a new file realm user is a dynamic event and does not require server restart.

1 **Ensure that the server is running.**

Remote commands require a running server.

2 **Create a file user by using the `create-file-user`(1) command.**

Example 6–1 Creating a User

The following example command creates a user named Frank on port 5001 of the host `pigeon`:

```
asadmin create-file-user --user admin1 --passwordfile passwords.txt
--host pigeon --port 5001 Frank
```

See Also To see the full syntax of the command, type `asadmin create-file-user --help` at the command line.

▼ To List File Users

The remote `list-file-users` command enables you to list the users that are in the `keyfile`.

1. **Ensure that the server is running.**
   Remote commands require a running server.
2. **List users by using the `list-file-users`(1) command.**

Example 6–2 Listing File Users

The following example command lists file users on a the host `pigeon`:

```
asadmin list-file-users --user admin1 --passwordfile passwords.txt --host pigeon
```

See Also To see the full syntax of the command, type `asadmin list-file-users --help` at the command line.

▼ To Update a File User

The remote `update-file-user` command enables you to modify the information in the keyfile for a specified user.

1. **Ensure that the server is running.**
   Remote commands require a running server.
2. **Notify users that information for the file user is being changed.**
3. **Update theuser by using the `update-file-user`(1) command.**
4. **To apply your changes, restart the Application Server.**
   a. **Stop the Application Server.**
      For instructions, see "To Stop a Domain (or Server)" on page 37.

b. Start the Application Server.

For instructions, see "To Start a Domain (or Server)" on page 36.

Example 6–3 Updating a User

The following command updates the groups for a user named Frank on port 5001 of the host `pigeon`:

```
asadmin update-file-user --host pigeon --port 5001
--groups staff:manager:engineer Frank
```

See Also To see the full syntax of the command, type `asadmin update-file-user --help` at the command line.

▼ To Delete a File User

The remote `delete-file-user` command enables you to remove a user entry from the keyfile by specifying the username.

1 Ensure that the server is running.

Remote commands require a running server.

2 Obtain the exact name of the file user that you are deleting.

To list the existing file users:

```
asadmin list-file-users
```

3 Notify users that the file user is being deleted.

4 Delete the user by using the `delete-file-user`(1) command.

5 To apply your changes, restart the Application Server.

a. Stop the Application Server.

For instructions, see "To Stop a Domain (or Server)" on page 37.

b. Start the Application Server.

For instructions, see "To Start a Domain (or Server)" on page 36.

Example 6–4 Deleting a User

The following example command deletes a user named Frank from host `pigeon`:

```
asadmin delete-file-user --user admin1 --passwordfile passwords.txt
--host pigeon --port 5001 Frank
```

See Also To see the full syntax of the command, type `asadmin delete-file-user --help` at the command line.

Managing Authentication Realms

The following tasks and information are used to manage authentication realms:

Command Options for Managing Authentication Realms

Options for managing authentication realms include the following:

Authentication Realm Name	Name of the realm.
`--target`	Specifies the target on which you are creating the realm. Valid values are `server`, *configuration_name*, *instance_name*.
`--classname`	Java class which implements this realm.
`--property`	Authentication realms require provider-specific properties, which vary depending on what a particular implementation needs. Which properties an authentication realm uses depends on the type of realm. The file realm uses file and jaas-context properties. Other realms use different properties. For more information, see

For details on these and additional options, see `create-auth-realm`(1) in the reference manual.

▼ To Create an Authentication Realm

The remote `create-auth-realm` command enables you to create an authentication realm.

1. **Ensure that the server is running.**
   Remote commands require a running server.
2. **Create a realm by using the `create-auth-realm`(1) command.**
3. **To apply your changes, restart the Application Server.**
   a. **Stop the Application Server.**
      For instructions, see "To Stop a Domain (or Server)" on page 37.
   b. **Start the Application Server.**
      For instructions, see "To Start a Domain (or Server)" on page 36.

Example 6–5 Creating a Realm

The following example command creates a realm named `db`:

```
asadmin create-auth-realm --classname com.iplanet.ias.security.auth.realm.DB.Database
--property defaultuser=admin:Password=admin db
```

See Also To see the full syntax of the command, type `asadmin create-auth-realm --help` at the command line.

▼ To List Authentication Realms

The remote `list-auth-realms` command enables you to list the existing authentication realms.

1. **Ensure that the server is running.**
   Remote commands require a running server.
2. **List realms by using the `list-auth-realms`(1) command.**

Example 6–6 Listing Realms

The following example command lists the authentication realms on the local host:

```
asadmin list-auth-realms --user admin --host localhost --port 4848
```

See Also To see the full syntax of the command, type `asadmin list-auth-realms --help` at the command line.

▼ To Delete an Authentication Realm

The remote `delete-auth-realm` command enables you to delete an existing authentication realm.

1. **Ensure that the server is running.**
   Remote commands require a running server.
2. **Obtain the exact name of the realm that you are deleting.**
   To list the existing realms:
   ```
   asadmin list-auth-realms
   ```
3. **Notify users that the realm is being deleted.**
4. **Delete the realm by using the `delete-auth-realm`(1) command.**
5. **To apply your changes, restart the Application Server.**
   a. **Stop the Application Server.**
      For instructions, see "To Stop a Domain (or Server)" on page 37.
   b. **Start the Application Server.**
      For instructions, see "To Start a Domain (or Server)" on page 36.

Example 6–7 Deleting a Realm

The following example command deletes an authentication realm named `db` from host `pigeon`:

```
asadmin delete-auth-realm --user admin1 --passwordfile password.txt
--host pigeon --port 5001 db
```

See Also To see the full syntax of the command, type `asadmin delete-auth-realm --help` at the command line.

▼ To Configure a JDBC Realm for a Java EE Application

The Application Server enables you to specify a user's credentials (user name and password) in the JDBC realm instead of in the connection pool. Using the JDBC realm instead of the connection pool prevents other applications from browsing the database tables for the user's credentials.

Note – By default, storage of passwords as clear text is not supported in the JDBC realm. Under normal circumstances, passwords should not be stored as clear text.

---

1. **Create the database tables in which to store users' credentials for the realm.**

   How to create the database tables depends on the database that you are using.

2. **Add the users' credentials to the database tables you created.**

   How to add users' credentials to the database tables depends on the database that you are using.

3. **Create a JDBC realm.**

   For instructions, see "To Create an Authentication Realm" on page 100.

4. **Modify the deployment descriptor to specify the JDBC realm.**

   Modify the deployment descriptor that is associated with your application:

   - **For an enterprise application in an Enterprise Archive (EAR) file, modify the `sun-application.xml` file.**
   - **For a web application in a Web Application Archive (WAR) file, modify the `web.xml` file.**
   - **For an enterprise bean in an EJB JAR file, modify the `sun-ejb-jar.xml` file.**

   For more information about how to specify a realm, see "How to Configure a Realm" in *GlassFish v3 Application Server Developer's Guide*.

5. **Assign a security role to users in the realm.**

   To assign a security role to a user, add a `security-role-mapping` element to the deployment descriptor that you modified. The following example shows a `security-role-mapping` element that assigns the security role `Employee` to user `Calvin`.

   ```
   <security-role-mapping>
       <role-name>Employee</role-name>
   ```

```
    <principal-name>Calvin</principal-name>
  </security-role-mapping>
```

◆ ◆ ◆ CHAPTER 7

Administering the HTTP Service

This chapter provides procedures for administering the HTTP service in the GlassFish Application Server environment by using the `asadmin` command-line utility.

The following topics are addressed here:

Instructions for accomplishing these tasks by using the Admin Console are contained in the Admin Console online help.

About the HTTP Service

The HTTP service provides functionality for deploying web applications and for making deployed web applications accessible by HTTP clients. HTTP services are provided by two kinds of related objects:

HTTP Listeners

An *HTTP listener* is a listen socket that has an Internet Protocol (IP) address, a port number, a server name, and a default virtual server. Each virtual server provides connections between the server and clients through one or more HTTP listeners. Each HTTP listener 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 host 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.

Because an HTTP listener is a combination of IP address and port number, you can have multiple HTTP listeners with the same IP address and different port numbers, or with different IP addresses and the same port number (if your host was configured to respond to these addresses). The host 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. However, if the host does have access to more than one IP address, each address can serve a different purpose.

By default, when the Application Server starts, the following HTTP listeners are started automatically:

- 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 does not have security enabled.

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:4848/` or `http://localhost:4848/asadmin/` (its default context root).

Default Ports for HTTP Listeners

The following table describes the Application Server default ports for the listeners that use ports.

TABLE 7–1 Default Ports for Listeners

Listener	Default Port	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 running an `asadmin` command remotely, specify the port number by using the `--port` option.
HTTP	8080	The Web server listens for HTTP requests on a port. To access deployed Web applications and services, clients connect to this port.
HTTPS	8181	Web applications configured for secure communications listen on a separate port.

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 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 example, assume you want to host the following domains on your physical server: `www.aaa.com`, `www.bbb.com`, and `www.ccc.com`. Assume that these domains are respectively associated with web modules `web1`, `web2`, and `web3`. This means that the following 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 server `www.aaa.com`, the second URL is mapped to virtual server `www.bbb.com`, and the third is mapped to virtual server `www.ccc.com`. For this mapping to work, `www.aaa.com`, `www.bbb.com`, and `www.ccc.com` must all resolve to your physical server's IP address. Each virtual server must be registered with the DNS server for your network.

By default, when the Application Server starts, the following virtual servers are started 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.

Managing HTTP Listeners

The following tasks and information are used to manage HTTP listeners:

- "Command Options for Managing HTTP Listeners" on page 108
- "To Create an HTTP Listener" on page 108
- "To List HTTP Listeners" on page 109

Command Options for Managing HTTP Listeners

Options for managing HTTP listeners include the following:

Listener ID	The listener ID of the HTTP listener.
`--listeneraddress`	The IP address or the hostname (resolvable by DNS) of the listener.
`--listenerport`	The port number to create the listen socket on. Legal values are 1–65535. On UNIX, creating sockets that listen on ports 1–1024 requires superuser privileges. Configuring an SSL listen socket to listen on port 443 is recommended.
`--defaultvs`	The ID attribute of the default virtual server for this listener.
`--acceptorthreads`	The number of acceptor threads for the listener socket. The recommended value is the number of processors in the machine. The default value is 1.
`--securityenabled`	If set to true, the HTTP listener runs SSL. You can turn SSL2 or SSL3 On or Off and set ciphers using an SSL element. The security setting globally enables or disables SSL by making certificates available to the server instance. Default value is false.

For details on these and additional options `create-http-listener`(1) in the reference manual.

▼ To Create an HTTP Listener

The remote `create-http-listener` command enables you to create an HTTP listener.

1 Ensure that the server is running.

Remote commands require a running server.

2 Create an HTTP listener by using the `create-http-listener`(1) command.

3 To apply your changes, restart the Application Server.

a. Stop the Application Server.

For instructions, see "To Stop a Domain (or Server)" on page 37.

b. Start the Application Server.

For instructions, see "To Start a Domain (or Server)" on page 36.

Example 7–1 Creating an HTTP Listener Port

The following example command creates an HTTP listener named `sampleListener` that uses a non-default number of acceptor threads and is not enabled at runtime:

```
create-http-listener --user admin1  --host host1 --port 4848
--listeneraddress 0.0.0.0 --listenerport 7272 --defaultvs server
--servername host1.sun.com --acceptorthreads 100 --securityenabled=false
--enabled=false sampleListener
```

See Also To see the full syntax of the command, type `asadmin help create-http-listener` at the command line.

▼ To List HTTP Listeners

The remote `list-http-listeners` command enables you to list the existing HTTP listeners.

1 Ensure that the server is running.

Remote commands require a running server.

2 List HTTP listeners by using the `list-http-listeners`(1) command.

Example 7–2 Listing HTTP Listeners

The following example command lists the HTTP listeners on `host1`:

```
asadmin list-http-listeners --user admin1 --host host1 --port 5001
```

See Also To see the full syntax of the command, type `asadmin list-http-listeners -help` at the command line.

▼ To Delete an HTTP Listener

The remote `delete-http-listener` command enables you to delete an existing HTTP listener.

1 Ensure that the server is running.

Remote commands require a running server.

2. **Obtain the exact name of the HTTP listener that you are deleting.**
   To list the existing HTTP listeners:
   ```
   asadmin list-http-listeners
   ```
3. **Notify users that the HTTP listener is being deleted.**
4. **Delete an HTTP listener by using the `delete-http-listener`(1) command.**
5. **To apply your changes, restart the Application Server.**
   a. **Stop the Application Server.**
      For instructions, see "To Stop a Domain (or Server)" on page 37.
   b. **Start the Application Server.**
      For instructions, see "To Start a Domain (or Server)" on page 36.

Example 7–3 Deleting an HTTP Listener

The following example command deletes the HTTP listener named `sampleListener`:

```
delete-http-listener --host host1 --port 5001 sampleListener
```

See Also To see the full syntax of the command, type `asadmin delete-http-listener -help` at the command line.

▼ To Configure an HTTP Listener for SSL

The remote `create-ssl` command enables you to create and configure an SSL element in the specified listener. This enables secure communication for the listener.

1. **Ensure that the server is running.**
   Remote commands require a running server.
2. **Configure an HTTP listener by using the `create-ssl`(1) command.**
3. **To apply your changes, restart the Application Server.**
   a. **Stop the Application Server.**
      For instructions, see "To Stop a Domain (or Server)" on page 37.

b. Start the Application Server.

For instructions, see "To Start a Domain (or Server)" on page 36.

Example 7–4 Configuring an HTTP Listener for SSL

The following example command configures the HTTP listener named `http-listener-1` for SSL:

```
asadmin create-ssl --type http-listener --certname sampleCert http-listener-1
```

See Also To see the full syntax of the command, type `asadmin create-ssl --help` at the command line.

▼ To Delete SSL From an HTTP Listener

The remote `delete-ssl` command enables you to delete the SSL element in the specified listener.

1 Ensure that the server is running.

Remote commands require a running server.

2 Delete SSL from an HTTP listener by using the `delete-ssl`(1) command.

3 To apply your changes, restart the Application Server.

a. Stop the Application Server.

For instructions, see "To Stop a Domain (or Server)" on page 37.

b. Start the Application Server.

For instructions, see "To Start a Domain (or Server)" on page 36.

Example 7–5 Deleting SSL from an HTTP Listener

The following example command removes SSL from the HTTP listener named `http-listener-1`:

```
delete-ssl --user admin --type http-listener http-listener-1
```

See Also To see the full syntax of the command, type `asadmin delete-ssl --help` at the command line.

Managing Virtual Servers

Virtualization in the Application Server allows multiple URL domains to be served by a single HTTP server process that is listening on multiple host addresses. If the application is available on two virtual servers, the servers still share the same physical resource pools.

The following tasks and information are used to manage virtual servers:

Command Options for Managing Virtual Servers

Options for managing virtual servers include the following:

Virtual Server ID	Identifies the unique ID for the virtual server to be created. This ID must not begin with a number.
`--hosts`	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).
`--httplisteners`	The existing HTTP listeners that are associated with this server. Required only for a virtual server that is not the default virtual server.
`--state`	Indicates the state of the virtual server as On, Off, or Disabled. Default value is On.
`--logfile`	Indicates the path name of the file where messages from this virtual server will appear. Default value is *domain-dir*`/logs/server.log`.
`--property`	Optional attribute name/value pairs for configuring the virtual server.

For details on the properties, see `create-virtual-server`(1) in the reference manual.

▼ To Create a Virtual Server

The remote `create-virtual-server` command enables you to create the named virtual server.

Before You Begin 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.

1. **Ensure that the server is running.**
   Remote commands require a running server.
2. **Create a virtual server by using the `create-virtual-server`(1) command.**
3. **To apply your changes, restart the Application Server.**
   a. **Stop the Application Server.**
      For instructions, see "To Stop a Domain (or Server)" on page 37.
   b. **Start the Application Server.**
      For instructions, see "To Start a Domain (or Server)" on page 36.

Example 7–6 Creating a Virtual Server

The following example command creates a virtual server named `sampleServer` for hosts `pigeon` and `localhost`:

```
asadmin create-virtual-server --user admin
--passwordfile -passwords.txt --hosts pigeon,localhost sampleServer
```

See Also To see the full syntax of the command, type `asadmin create-virutal-server --help` at the command line.

▼ To List Virtual Servers

The remote `list-virtual-servers` command enables you to list the existing virtual servers.

1. **Ensure that the server is running.**
   Remote commands require a running server.
2. **List virtual servers by using the `list-virtual-servers`(1) command.**

Example 7–7 Listing Virtual Servers

The following example command lists all the virtual servers for the server instance:

```
asaadmin list-virtual-servers --user admin --host localhost --port 4848
```

See Also To see the full syntax of the command, type `asadmin list-virutal-servers --help` at the command line.

▼ To Delete a Virtual Server

The remote `delete-virtual-server` command enables you to delete an existing virtual server.

1. **Ensure that the server is running.**
   Remote commands require a running server.
2. **Obtain the exact name of the virtual server that you are deleting.**
   To list the existing virtual servers:
   ```
   asadmin list-virtual-servers
   ```
3. **Notify users that the virtual server is being deleted.**
4. **Delete a virtual server by using the `delete-virtual-server`(1) command.**
5. **To apply your changes, restart the Application Server.**
   a. **Stop the Application Server.**
      For instructions, see "To Stop a Domain (or Server)" on page 37.
   b. **Start the Application Server.**
      For instructions, see "To Start a Domain (or Server)" on page 36.

Example 7–8 Deleting a Virtual Server

The following example command deletes the virtual server named `sample_vs1`: from host `pigeon`:

```
delete-virtual-server --user admin1 --hosts pigeon --port 5001 sample_vs1
```

See Also To see the full syntax of the command, type `asadmin delete-virutal-server --help` at the command line.

◆ ◆ ◆ CHAPTER 8

Administering Logging

This chapter contains instructions on how to configure logging and how to view server logs for the GlassFish Application Server.

The following topics are addressed here:

About Logging

Server components and application components generate output for the Application Server logs. Although application components can use the Apache Commons Logging Library to log messages, the platform standard JSR 047 API is recommended for better log configuration.

Application Server log messages are recorded in the server log, normally found at *domain-dir*`/logs/server.log`. In addition to the server log, the *domain-dir*`/logs` directory contains the following additional logs:

- HTTP service access logs, located in the `/access` subdirectory
- Transaction service logs, located in the `/tx` subdirectory

When a log is rotated, the Application Server creates a new, empty file named `server.log` and renames the old file `server.log_`*date*, where *date* is the date and time that the file was rotated.

The following topics are addressed here:

Log Record Format

The Application Server log records follow a uniform format:

`[#|`*yyyy-mm-dd*`Thh:`*mm*`:`*ss*`.`*SSS*`-Z|`*Log Level*`|`*ProductName-Version*`|`*LoggerName*`|`*Key Value Pairs*`|`*Message*`|#]`

- `[#` and `#]` mark the beginning and end of the record.
- The vertical bar (`|`) separates the fields of the record.
- *yyyy-mm-ddThh:mm:ss.SSSS-Z* specifies the date and time the record was created. For example: `2006-10-21T13:25:53.852-0400`
- *Log Level* specifies the desired log level. You can select any of the following values: `SEVERE`, `WARNING`, `INFO`, `CONFIG`, `FINE`, `FINER`, and `FINEST`. The default is `INFO`.
- *ProductName-Version* refers to the current version of the Application Server. For example: `sun-appserver10`
- *LoggerName* is a hierarchical logger namespace that identifies the source of the log module. For example: `javax.enterprise.system.core`
- *Key Value Pairs* refers to pairs of key names and values, typically a thread ID. For example: `_ThreadID=14;`
- *Message* is the text of the log message. For all Application Server `SEVERE` and `WARNING` messages and for many `INFO` messages, the message begins with a message ID that consists of a module code and a numerical value. For example: `CORE5004`

An example log record might look like this:

```
[#|2006-10-21T13:25:53.852-0400|INFO|sun-appserver10|javax.enterprise.
system.core|_ThreadID=13;|CORE5004: Resource Deployed:
[cr:jms/DurableConnectionFactory].|#]
```

The Admin Console presents log records in a readable display. When you view log records in the Admin Console, you can filter the log records to display only the records you want to see. Filtering options are described in "Viewing Server Logs" on page 119.

Logger Namespace Hierarchy

A logger is provided for each of the Application Server modules. The following table lists the names of the modules and the namespace for each logger in alphabetical order. The last three modules in the table do not appear on the Log Levels page of the Admin Console.

TABLE 8–1 Logger Namespaces for Application Server Modules

Module Name	Namespace
Admin	`javax.enterprise.system.tools.admin`
Classloader	`javax.enterprise.system.core.classloading`
Configuration	`javax.enterprise.system.core.config`
Deployment	`javax.enterprise.system.tools.deployment`
Persistence	`oracle.toplink.essentials`, `javax.enterprise.resource.jdo`, `javax.enterprise.system.container.cmp`
Root	`javax.enterprise`
Security	`javax.enterprise.system.core.security`
Util	`javax.enterprise.system.util`
Verifier	`javax.enterprise.system.tools.verifier`

Web container	`javax.enterprise.system.container.web`
	`org.apache.catalina`
	`org.apache.coyote`
	`org.apache.jasper`

Configuring Logging

You can configure logging settings and log levels using the Admin Console or the `asadmin` utility.

- "Settings for Logging" on page 117
- "To Configure General Logging Settings" on page 118
- "To Configure Log Levels" on page 118

Settings for Logging

The following logging settings can be adjusted:

Log File — You can rename or relocate the server log file using the absolute path. The name contains only alphanumeric, underscore, dash, or dot characters.

Alarms — If enabled, SEVERE and WARNING messages are routed through the JMX framework.

Write to system log — If enabled, the UNIX syslog service is used to produce and manage log messages.

Log Handler	Specifies a custom log handler for logging to a different destination.
File Rotation Limit	Rotates log files when the specified rotation byte limit is reached. For example, 2000000.
File Rotation Time Limit	Rotates log files when the specified rotation minutes limit is reached. If set to 0, rotates based on byte limit specified.
Retain Error Statistics	Specifies the number of hours that error statistics are retained. For example, 5.

If any additional properties have been set, you can adjust them.

▼ To Configure General Logging Settings

You can configure the general logging settings using the Admin Console. For details on setting the various configuration parameters, click Help.

To configure these log settings on the command line, use the `get` and `set` commands of the `asadmin` utility.

1 On the General page, enter appropriate values to customize logging to your requirements.

- For the developer profile, go to Application Server → Logging → General

2 Stop the Application Server.

3 Start the Application Server.

▼ To Configure Log Levels

You can configure log levels using the Admin Console. You can choose from among the following log levels: `SEVERE`, `WARNING`, `INFO` (the default), `CONFIG`, `FINE`, `FINER`, and `FINEST`

To configure these log settings on the command line, use the `get` and `set` commands of the `asadmin` utility.

Changing a log level is a dynamic event and does not require server restart.

1 Go to the Log Levels page:

- For the developer profile, go to Application Server → Logging → Log Levels

2 **Set the log level for the modules listed on this page.**

Use the Additional Properties area to configure log levels for any application loggers. For a list of the module loggers, see "Logger Namespace Hierarchy" on page 116.

Viewing Server Logs

In the Admin Console View Logs page, you can provide search criteria for displaying only those records that you want to see. 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. For instructions, see "To Configure Log Levels" on page 118.

The following filtering options are provided:

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 form *hh:mm:ss.SSS* (*SSS* stands for milliseconds). For example: `17:10:00.000`
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, including more severe levels. Select the checkbox labeled "Do not include more severe messages" to display messages at only the chosen level. ■ The most recent 40 entries in the server log appear, with the settings specified on the Logging Settings and Log Levels pages. ■ Click the arrow next to the Timestamp header to sort the messages so that the most recent message is displayed last. ■ To view a formatted version of any message, click the link marked (details). A window labeled Log Entry Detail presents 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.
Advanced Search	Click Advanced Search in the Search Criteria area to make additional refinements to the log viewer. (Click Basic Search to hide the Advanced Options area.) 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. 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.

◆ ◆ ◆ APPENDIX A

The `asadmin` Utility Commands

This appendix lists the `asadmin` commands that are included with this release of the GlassFish Application Server.

For general information on the `asadmin` utility, see "Command-Line Utility" on page 28.

Online help for the `asadmin` commands can be invoked on the command line. For example, `asadmin create-domain`. The *GlassFish v3 Application Server Reference Manual* also provides a collection of these help pages.

Note – The common options used with remote commands are described in the `asadmin`(1M) help page.

Basic Administration Commands

`backup-domain`(1)	Makes a copy of the files that are under the specified domain.
`create-domain`(1)	Creates the configuration of a domain. A domain can exist independent of other domains. Any user who has access to the `asadmin` utility on a given host can create a domain and store its configuration in a location of choice. By default, the domain configuration is stored in the *as-install*`/domains` directory. You can override this location to store the configuration elsewhere.

`create-system-properties`(1)	Any configuration attribute can be overwritten through a system property of the corresponding name. The `create-system-properties` command creates or updates such properties. Supported in remote mode only.
`delete-domain`(1)	Deletes the specified domain. The domain must be stopped before it can be deleted.
`delete-system-property`(1)	Deletes system properties of a domain or configuration. Supported in remote mode only.
`get`(1)	Gets the names and values of the monitorable or configurable attributes.
`help`(1)	Displays a list of all the `asadmin` utility commands. To display the usage information for a particular command, specify the command. For example, `asadmin list-domains --help`
`list`(1)	Lists the configurable element. On Solaris, quotes are needed when running commands with * as the option value or operand.
`list-applications`(1)	Lists deployed J2EE applications. If the `--type` option is not specified, all applications are listed. Supported in remote mode only.
`list-commands`(1)	Lists all the `asadmin` commands, local commands first, then remote commands. You can specify that only remote commands or only local commands be displayed. Supported in remote mode only.
`list-containers`(1)	Lists application containers and the status of each container. Supported in remote mode only.
`list-domains`(1)	Lists the existing domains. If the domain directory is not specified, the domain in the default *as-install*`/domains` directory is displayed. If there is more than one domain, the *domain_name* operand must be specified.
`list-modules`(1)	Lists modules that are accessible to the Application Server subsystem. The status of each module is included. Supported in remote mode only.
`list-system-properties`(1)	Lists the system properties of a domain or configuration. Supported in remote mode only.
`restore-domain`(1)	Restores files from a backup directory for the specified domain.

`set`(1)	Sets the values of one or more configurable attributes.
`start-domain`(1)	Starts a domain. If the domain directory is not specified, the domain in the default *as-install*`/domains` directory is started. If there are two or more domains, the *domain_name* operand must be specified.
`start-database`(1)	Starts the Java DB database server. Use this command only for working with applications deployed to the Application Server.
`stop-database`(1)	Stops a process of the Java DB database server.
`stop-domain`(1)	Stops the domain administration server (DAS) of the specified domain. Supported in remote mode only.

Deployment Commands

`deploy`(1)	Deploys an enterprise application, web application, EJB module, connector module, or application client module. If the component is already deployed or already exists, you can forcefully redeploy if you set the `--force` option to `true`. Supported in remote mode only.
`disable`(1)	Immediately disables the named component. If the component has not been deployed, an error message is returned. Supported in remote mode only.
`enable`(1)	Enables the specified component. If the component has not been deployed, an error message is returned. If the component is already enabled, then it is re-enabled. Supported in remote mode only.
`list-components`(1)	Lists all deployed Java EE 5 components. If the `--type` option is not specified, all components are listed. Supported in remote mode only.
`redeploy`(1)	Redeploys an application that is already deployed. Supported in remote mode only.
`undeploy`(1)	Removes the specified deployed component. Supported in remote mode only.

HTTP Service Commands

`create-http-listener`(1)	Creates a new HTTP listener socket. Restart the server for the creation to take effect. Supported in remote mode only.
`create-virtual-server`(1)	Creates the specified virtual server. Restart the server for the creation to take effect. Supported in remote mode only.
`create-ssl`(1)	Creates and configures the SSL element in the selected HTTP listener to enable secure communication on that listener/service. Restart the server for the creation to take effect. Supported in remote mode only.
`delete-http-listener`(1)	Deletes the specified HTTP listener. Restart the server for the deletion to take effect. Supported in remote mode only.
`list-http-listeners`(1)	Lists the existing HTTP listeners. Supported in remote mode only.
`delete-ssl`(1)	Deletes the SSL element in the selected HTTP listener. Restart the server for the deletion to take effect. Supported in remote mode only.
`delete-virtual-server`(1)	Deletes the specified virtual server. Restart the server for the deletion to take effect. Supported in remote mode only.
`list-virtual-servers`(1)	Lists the existing virtual servers. Supported in remote mode only.

JVM Commands

`create-jvm-options`(1)	Creates a JVM option in the Java configuration or profiler elements of the `domain.xml` file. If JVM options are created for a profiler, the options are used to record the settings needed to activate a particular profiler. restart the server for newly created JVM options to take effect. Supported in remote mode only.
`create-profiler`(1)	Creates a profiler element. A server instance is tied to a particular profiler, by the profiler element in the Java configuration. Restart the server for a newly created profiler to take effect. Supported in remote mode only.
`delete-jvm-options`(1)	Deletes the specified JVM option from the Java configuration or profiler elements of the `domain.xml` file. Restart the server for the deletion to take effect. Supported in remote mode only.

`delete-profiler`(1)	Deletes the specified profiler element. Restart the server for the deletion to take effect. Supported only in remote mode.
`list-jvm-options`(1)	Lists the command-line options that are passed to the Java application launcher when the Application Server is started. Supported in remote mode only.

Resource Management Commands

`add-resources`(1)	Creates the resources named in the specified XML file. The *xml_file_path* is the path to the XML file containing the resources to be created. The `DOCTYPE` should be specified as *as-install*`/lib/dtds/sun-resources_1_2.dtd` in the `resources.xml` file. Supported in remote mode only.
`create-jdbc-connection-pool`(1)	Registers a new JDBC connection pool with the specified JDBC connection pool name. Supported in remote mode only.
`create-jdbc-resource`(1)	Creates a new JDBC resource. Supported in remote mode only.
`create-resource-ref`(1)	Creates a reference to a previously created resource. Supported in remote mode only.
`delete-jdbc-connection-pool`(1)	Deletes the specified JDBC connection pool. Supported in remote mode only.
`delete-jdbc-resource`(1)	Deletes a JDBC resource. The specified JNDI name identifies the resource to be deleted. Supported in remote mode only.
`delete-resource-ref`(1)	Deletes a reference from a server instance for a resource. The resource is not removed from the domain. Supported in remote mode only.
`list-jdbc-connection-pools`(1)	Lists the existing JDBC connection pools. Supported in remote mode only.
`list-jdbc-resources`(1)	Lists the existing JDBC resources. Supported in remote mode only.
`list-resource-refs`(1)	Lists all resource references in a server instance. Supported in remote mode only.
`ping-connection-pool`(1)	Tests if a JDBC connection pool is usable. Before you can ping a JDBC connection pool, you must create the

	connection pool with authentication and ensure that the server or database is started. Supported in remote mode only.
`version`(1)	Displays the version information for the option specified. If the command cannot communicate with the administration server with the given user/password and host/port, then the command will retrieve the version locally and display a warning message. Supported in remote mode only.

User Management Commands

`create-auth-realm`(1)	Adds the specified authentication realm. Restart the server for the creation to take effect. Supported in remote mode only.
`create-file-user`(1)	Creates an entry in the keyfile with the specified username, password, and groups. Multiple groups can be created by separating them with colons (:). Restart the server for the creation to take effect. Supported in remote mode only.
`delete-auth-realm`(1)	Deletes the specified authentication realm. Restart the server for the deletion to take effect. Supported in remote mode only.
`delete-file-user`(1)	Deletes the specified user entry in the keyfile. Restart the server for the deletion to take effect. Supported in remote mode only.
`list-auth-realms`(1)	Lists the existing authentication realms. Supported in remote mode only.
`list-file-users`(1)	Lists the file users supported by the `file` realm authentication method. Supported in remote mode only.
`update-file-user`(1)	Updates an existing entry in the keyfile using the specified username, password, and groups. Restart the server for the change to take effect. Supported in remote mode only.

Index

D

E

L

M

N

O

P

R

S

T

U

V

W