Oracle® GlassFish Server 3.1 Security Guide
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>9</td>
</tr>
<tr>
<td>1 Administering System Security</td>
<td>17</td>
</tr>
<tr>
<td>About System Security in GlassFish Server</td>
<td>17</td>
</tr>
<tr>
<td>Authentication</td>
<td>18</td>
</tr>
<tr>
<td>Authorization</td>
<td>23</td>
</tr>
<tr>
<td>Auditing</td>
<td>28</td>
</tr>
<tr>
<td>Firewalls</td>
<td>28</td>
</tr>
<tr>
<td>Certificates and SSL</td>
<td>28</td>
</tr>
<tr>
<td>Tools for Managing System Security</td>
<td>35</td>
</tr>
<tr>
<td>Administering Passwords</td>
<td>36</td>
</tr>
<tr>
<td>▼ To Change the Master Password</td>
<td>36</td>
</tr>
<tr>
<td>Additional Considerations for the start-instance and start-cluster</td>
<td>37</td>
</tr>
<tr>
<td>Subcommands</td>
<td>37</td>
</tr>
<tr>
<td>▼ Using start-instance and start-cluster With a Password File</td>
<td>38</td>
</tr>
<tr>
<td>▼ To Change an Administration Password</td>
<td>40</td>
</tr>
<tr>
<td>▼ To Set a Password From a File</td>
<td>41</td>
</tr>
<tr>
<td>Administering Password Aliases</td>
<td>41</td>
</tr>
<tr>
<td>Administering Audit Modules</td>
<td>45</td>
</tr>
<tr>
<td>▼ To Create an Audit Module</td>
<td>45</td>
</tr>
<tr>
<td>▼ To List Audit Modules</td>
<td>45</td>
</tr>
<tr>
<td>▼ To Delete an Audit Module</td>
<td>46</td>
</tr>
<tr>
<td>Administering JSSE Certificates</td>
<td>47</td>
</tr>
<tr>
<td>▼ To Generate a Certificate by Using keytool</td>
<td>47</td>
</tr>
<tr>
<td>▼ To Sign a Certificate by Using keytool</td>
<td>49</td>
</tr>
<tr>
<td>▼ To Delete a Certificate by Using keytool</td>
<td>52</td>
</tr>
<tr>
<td>Administering JACC Providers</td>
<td>52</td>
</tr>
<tr>
<td>▼ Administering JACC Providers From the Administration Console</td>
<td>53</td>
</tr>
</tbody>
</table>
2 Administering User Security ................................................................. 55

Overview of Authentication Realms ..................................................... 56
▼ To Create an Authentication Realm ................................................. 57
▼ To List Authentication Realms ......................................................... 58
▼ To Update an Authentication Realm .............................................. 58
▼ To Delete an Authentication Realm ............................................... 59
▼ To Configure a JDBC or Digest Authentication Realm ................. 59
▼ To Configure LDAP Authentication with OID............................ 61
▼ To configure LDAP Authentication with OVD ......................... 62
▼ To Enable LDAP Authentication on the GlassFish Server DAS ... 63

Administering File Users ................................................................. 64
▼ To Create a File User ................................................................ 65
▼ To List File Users ....................................................................... 65
▼ To List File Groups ................................................................. 66
▼ To Update a File User ............................................................... 67
▼ To Delete a File User ................................................................. 67

3 Administering Message Security ..................................................... 69

About Message Security in GlassFish Server ................................. 69
Security Tokens and Security Mechanisms ...................................... 70
Authentication Providers ............................................................... 71
Message Protection Policies .......................................................... 72
Application-Specific Web Services Security ................................. 72
Message Security Administration ................................................ 73
Sample Application for Web Services ............................................. 74

Enabling Default Message Security Providers for Web Services .... 75
▼ To Enable a Default Server Provider ......................................... 75
▼ To Enable a Default Client Provider .......................................... 76
Configuring Message Protection Policies ....................................... 76
Message Protection Policy Mapping .............................................. 76
▼ To Configure the Message Protection Policies for a Provider ...... 78
Setting the Request and Response Policy for the Application Client Configuration .......... 78
6 Running in a Secure Environment ................................................................. 107
   Determining Your Security Needs .............................................................. 107
   Understand Your Environment ................................................................. 108
   Hire Security Consultants or Use Diagnostic Software .......................... 108
   Read Security Publications ....................................................................... 108
   Installing GlassFish Server in a Secure Environment ............................... 109
   Enable the Secure Administration Feature .............................................. 109
   Remove Unused Components ................................................................... 109
      Removing Installed Components .......................................................... 110
      Remove Services You Are Not Using .................................................. 111
   Run on the Web Profile if Possible .......................................................... 112
   Securing the GlassFish Server Host ......................................................... 113
   Securing GlassFish Server ....................................................................... 116
   Securing Applications ............................................................................. 119

7 Integrating Oracle Access Manager ............................................................. 121
   About OAM Security Provider for Glassfish .......................................... 122
      Obtaining Oracle Access Manager Group Information ........................ 123
   About Oracle Access Manager ................................................................ 123
   Understanding OAM Security Provider Use Cases ................................. 125
      Use Case: Authentication for Web Resources Via Access Gate ............ 125
      Use Case: Identity Assertion for Web Resources via WebGate ............ 127
      Use Case: Authorization Checks Based on Policy Manager ............... 129
   Configuring the OAM Security Provider .................................................. 130
   Determining Which Authentication Method is Used ............................... 132
   Integrating OAM Security Provider with Oracle Access Manager 10g .... 133
      ▼ Integrating OAM Security Provider with Oracle Access Manager 10g 134
   Integrating OAM Security Provider with Oracle Access Manager 11g .... 147
      ▼ Integrating OAM Security Provider with Oracle Access Manager 11g 147
      Additional Considerations for Certificate Authentication .................. 162
   Integrating OAM Security Provider with Oracle Access Manager 11g and WebGate ... 164
      ▼ Integrating OAM Security Provider with Oracle Access Manager 11g and WebGate 165
      Additional Considerations for Certificate Authentication With a WebGate 175
      Session Synchronization ..................................................................... 178
Index ................................................................................................................................................... 181
Preface

The Oracle GlassFish Server Security Guide provides instructions for configuring and administering GlassFish Server security.

This preface contains information about and conventions for the entire Oracle GlassFish Server (GlassFish Server) documentation set.

GlassFish Server 3.1 is developed through the GlassFish project open-source community at http://glassfish.java.net/. The GlassFish project provides a structured process for developing the GlassFish Server platform that makes the new features of the Java EE platform available faster, while maintaining the most important feature of Java EE: compatibility. It enables Java developers to access the GlassFish Server source code and to contribute to the development of the GlassFish Server. The GlassFish project is designed to encourage communication between Oracle engineers and the community.

The following topics are addressed here:

- “GlassFish Server Documentation Set” on page 9
- “Related Documentation” on page 11
- “Typographic Conventions” on page 12
- “Symbol Conventions” on page 13
- “Default Paths and File Names” on page 13
- “Documentation, Support, and Training” on page 14
- “Searching Oracle Product Documentation” on page 14
- “Third-Party Web Site References” on page 15

GlassFish Server Documentation Set

The GlassFish Server documentation set describes deployment planning and system installation. For an introduction to GlassFish Server, refer to the books in the order in which they are listed in the following table.
### Books in the GlassFish Server Documentation Set

<table>
<thead>
<tr>
<th>Book Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release Notes</td>
<td>Provides late-breaking information about the software and the documentation and includes a comprehensive, table-based summary of the supported hardware, operating system, Java Development Kit (JDK), and database drivers.</td>
</tr>
<tr>
<td>Quick Start Guide</td>
<td>Explains how to get started with the GlassFish Server product.</td>
</tr>
<tr>
<td>Installation Guide</td>
<td>Explains how to install the software and its components.</td>
</tr>
<tr>
<td>Upgrade Guide</td>
<td>Explains how to upgrade to the latest version of GlassFish Server. This guide also describes differences between adjacent product releases and configuration options that can result in incompatibility with the product specifications.</td>
</tr>
<tr>
<td>Deployment Planning Guide</td>
<td>Explains how to build a production deployment of GlassFish Server that meets the requirements of your system and enterprise.</td>
</tr>
<tr>
<td>Administration Guide</td>
<td>Explains how to configure, monitor, and manage GlassFish Server subsystems and components from the command line by using the <code>asadmin(1M)</code> utility. Instructions for performing these tasks from the Administration Console are provided in the Administration Console online help.</td>
</tr>
<tr>
<td>Security Guide</td>
<td>Provides instructions for configuring and administering GlassFish Server security.</td>
</tr>
<tr>
<td>Application Deployment Guide</td>
<td>Explains how to assemble and deploy applications to the GlassFish Server and provides information about deployment descriptors.</td>
</tr>
<tr>
<td>Application Development Guide</td>
<td>Explains how to create and implement Java Platform, Enterprise Edition (Java EE platform) applications that are intended to run on the GlassFish Server. These applications follow the open Java standards model for Java EE components and application programmer interfaces (APIs). This guide provides information about developer tools, security, and debugging.</td>
</tr>
<tr>
<td>Add-On Component Development Guide</td>
<td>Explains how to use published interfaces of GlassFish Server to develop add-on components for GlassFish Server. This document explains how to perform only those tasks that ensure that the add-on component is suitable for GlassFish Server.</td>
</tr>
<tr>
<td>Embedded Server Guide</td>
<td>Explains how to run applications in embedded GlassFish Server and to develop applications in which GlassFish Server is embedded.</td>
</tr>
<tr>
<td>High Availability Administration Guide</td>
<td>Explains how to configure GlassFish Server to provide higher availability and scalability through failover and load balancing.</td>
</tr>
<tr>
<td>Performance Tuning Guide</td>
<td>Explains how to optimize the performance of GlassFish Server.</td>
</tr>
</tbody>
</table>
Related Documentation

The following tutorials explain how to develop Java EE applications:

- **Your First Cup: An Introduction to the Java EE Platform** ([http://download.oracle.com/javaee/6/firstcup/doc/](http://download.oracle.com/javaee/6/firstcup/doc/)). For beginning Java EE programmers, this short tutorial explains the entire process for developing a simple enterprise application. The sample application is a web application that consists of a component that is based on the Enterprise JavaBeans specification, a JAX-RS web service, and a JavaServer Faces component for the web front end.

- **The Java EE 6 Tutorial** ([http://download.oracle.com/javaee/6/tutorial/doc/](http://download.oracle.com/javaee/6/tutorial/doc/)). This comprehensive tutorial explains how to use Java EE 6 platform technologies and APIs to develop Java EE applications.
Javadoc tool reference documentation for packages that are provided with GlassFish Server is available as follows.

- The API specification for version 6 of Java EE is located at http://download.oracle.com/javaee/6/api/.
- The API specification for GlassFish Server 3.1, including Java EE 6 platform packages and nonplatform packages that are specific to the GlassFish Server product, is located at http://glassfish.java.net/nonav/docs/v3/api/.

Additionally, the Java EE Specifications (http://www.oracle.com/technetwork/java/javaee/tech/index.html) might be useful.

For information about creating enterprise applications in the NetBeans Integrated Development Environment (IDE), see the NetBeans Documentation, Training & Support page (http://www.netbeans.org/kb/).

For information about the Java DB database for use with the GlassFish Server, see the Java DB product page (http://www.oracle.com/technetwork/java/javadb/overview/index.html).

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

## Typographic Conventions

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

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

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

**TABLE P–3 Symbol Conventions**

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

Default Paths and File Names

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

**TABLE P–4 Default Paths and File Names**

<table>
<thead>
<tr>
<th>Placeholder</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>as-install</code></td>
<td>Represents the base installation directory for GlassFish Server.</td>
<td>Installations on the Oracle Solaris operating system, Linux operating system, and Mac OS operating system: <code>user's-home-directory/glassfish3/glassfish</code> Windows, all installations: <code>SystemDrive:\glassfish3\glassfish</code></td>
</tr>
</tbody>
</table>
TABLE P-4  Default Paths and File Names  (Continued)

<table>
<thead>
<tr>
<th>Placeholder</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
</table>
| as-install-parent | Represents the parent of the base installation directory for GlassFish Server. | Installations on the Oracle Solaris operating system, Linux operating system, and Mac operating system:  
user's-home-directory/glassfish3  
Windows, all installations:  
SystemDrive: \glassfish3 |
| domain-root-dir | Represents the directory in which a domain is created by default.           | as-install/domains/                                                                               |
| domain-dir      | Represents the directory in which a domain's configuration is stored.       | domain-root-dir/domain-name                                                                       |
|                 | In configuration files, domain-dir is represented as follows:               |                                                    |
|                 | ${com.sun.aas.instanceRoot}                                                 |                                                    |

Documentation, Support, and Training

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

- Documentation (http://www.oracle.com/technetwork/indexes/documentation/index.html)
- Training (http://education.oracle.com/)

Searching Oracle Product Documentation

Besides searching Oracle product documentation from the Oracle Documentation (http://www.oracle.com/technetwork/indexes/documentation/index.html) web site, you can use a search engine by typing the following syntax in the search field:

search-term site:oracle.com

For example, to search for “broker,” type the following:

broker site:oracle.com
Third-Party Web Site References

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

Note – Oracle is not responsible for the availability of third-party web sites mentioned in this document. Oracle 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. Oracle 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.
This chapter describes general information about administering system security.

The following topics are addressed here:

- "About System Security in GlassFish Server" on page 17
- "Administering Passwords" on page 36
- "Administering Audit Modules" on page 45
- "Administering JSSE Certificates" on page 47
- "Administering JACC Providers" on page 52

Instructions for accomplishing many of these tasks by using the Administration Console are contained in the Administration Console online help.

Information on application security is contained in Chapter 4, “Securing Applications,” in Oracle GlassFish Server 3.1 Application Development Guide.

About System Security in GlassFish Server

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 Server is built on the Java security model, which uses a sandbox where applications can run safely, without potential risk to systems or users. System security affects all the applications in the GlassFish Server environment.

System security features include the following:

- "Authentication" on page 18
- "Authorization" on page 23
- "Auditing" on page 28
- "Firewalls" on page 28
- "Certificates and SSL" on page 28
- "Tools for Managing System Security" on page 35
Authentication

Authentication is the way in which 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 themselves. 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, GlassFish 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.

Authentication Types

Within its deployment descriptors, an application specifies the type of authentication that it uses. GlassFish 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. This type is not considered to be a secure method of user authentication unless used in conjunction with some external secure system such as 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.

- **DIGEST**
  - The server authenticates a user based on a user name and a password. Unlike BASIC authentication, the password is never sent over the network. The use of SSL with HTTP Digest is optional.

JSR 196 Server Authentication Modules

GlassFish Server implements the Servlet Container Profile of JSR 196 Java Authentication Service Provider Interface for Containers specification.

JSR 196 defines a standard service-provider interface (SPI) for integrating authentication mechanism implementations in message processing runtimes. JSR 196 extends the concepts of the Java Authentication and Authorization Service (JAAS) to enable pluggability of message
authentication modules in message processing runtimes. The standard defines profiles that establish contracts for the use of the SPI in specific contexts.

JSR 196 allows you to develop plug-ins at different layers. You can define plug-ins that change the way new authentication mechanism are configured, such as, AuthConfigProvider and AuthConfigFactory. You can also define new authentication mechanisms, such as ServerAuthModule and ClientAuthModule.

The OAM Security Provider for GlassFish, described in Chapter 7, “Integrating Oracle Access Manager,” is a JSR 196 Server Authentication Module (SAM) that provides authentication, authorization (optional), and single sign-on across Java EE Web applications that are deployed on GlassFish Server.

**Passwords**

Passwords are your first line of defense against unauthorized access to the components and data of GlassFish Server. For Information about how to use passwords for GlassFish Server, see “Administering Passwords” on page 36.

**Master Password and Keystores**

The *master password* is not tied to a user account and it is not used for authentication. Instead, GlassFish Server uses the master password only to encrypt the keystore and truststore for the DAS and instances.

When you create a new GlassFish Server domain, a new self-signed certificate is generated and stored in the domain keystore and truststore. The DAS needs the master password to open these stores at startup. Similarly, the associated server instances need the master password to open their copy of these stores at startup.

If you use a utility such as keytool to modify the keystore or truststore, you must provide the master password in that case as well.

The master password is a shared password and must be the same for the DAS and all instances in the domain in order to manage the instances from the DAS. However, because GlassFish Server never transmits the master password over the network, it is up to you to keep the master password in sync between the DAS and instances.

If you change the master password, you can choose to enter the master password manually when required, or save it in a file.
**Understanding Master Password Synchronization**

The master password is used to encrypt the keystore and truststore for the DAS and instances. The DAS needs the master password to open these stores at startup. Similarly, the associated server instances need the master password to open their copy of these stores at startup.

GlassFish Server keeps the keystore and truststore for the DAS and instances in sync, which guarantees that all copies of the stores are encrypted with the same master password at any given time.

However, GlassFish Server does not synchronize the master password itself, and it is possible that the DAS and instances might attempt to use different master passwords.

Consider the following potential scenario:

1. You create a domain and instances, using the default master password (changeit). As a result, the DAS and instances have keystores and truststores encrypted using changeit.

2. You use the change-master-password subcommand on the DAS to change the master password to ichangedit. As a result, the DAS and instance keystores and truststores are encrypted using ichangedit.

3. Access to the keystore and truststore from an instance now requires the master password ichangedit. You are responsible for changing the master password as needed.

If you do not use a master password file, you assume the responsibility for using the change-master-password subcommand on the DAS and instances to keep the master passwords in sync. Be aware that not using a master password file has additional considerations for the start-instance and start-cluster subcommands, as described in "Additional Considerations for the start-instance and start-cluster Subcommands" on page 37.

If you do use a master password file, you assume the responsibility for using the change-master-password subcommand on the DAS and instances to keep the master password file in sync.

**Using the Default Master Password**

GlassFish Server uses the known phrase "changeit" as the default master password. This master password is not stored in a file. The default password is a convenience feature and provides no additional security because it is assumed to be widely known.

All GlassFish Server subcommands work as expected with the default master password and there are no synchronization issues.
Saving the Master Password to a File

The `change-master-password --savemasterpassword` option indicates whether the master password should be written to the file system in the `master-password` file for the DAS or a node. The default is false.

For a domain, the master password is kept in `domains/directory/domain-name/master-password`.

For a node, the master-password file is kept in `nodes/directory/node-name/agent/master-password`. You can set a master password at the node level and all instances created on that node will use that master-password file. To do this, use the `--nodedir` option and provide a node name.

You might want to save the master password to the file so that the `start-domain` subcommand can start the server without having to prompt the user. There are additional considerations for using a master password with the `start-instance` and `start-cluster` subcommands, as described in "Additional Considerations for the start-instance and start-cluster Subcommands" on page 37.

The master-password file is encoded, not encrypted. You must use filesystem permissions to protect the file.

Using the Master Password When Creating a Domain

The `create-domain --usemasterpassword` option specifies whether the keystore is encrypted with a master password that is built into the system, or by a user-defined master password.

- If false (default), the keystore is encrypted with a well-known password (change it) that is built into GlassFish Server.

- If true, the subcommand obtains the master password from the `AS_ADMIN_MASTERPASSWORD` entry in the password file you specified in the `--passwordfile` option of the `asadmin` utility. Or, if none is defined, `--usemasterpassword` prompts the user for the master password.

Administration Password

An administration password, also known as the admin password, is used to invoke the Administration Console and the `asadmin` utility. As with the default admin username, the default admin password is usually set during installation but it can be changed. For instructions, see "To Change an Administration Password" on page 40.
Encoded Passwords

Files that contain encoded passwords 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 `-password-file` argument to the `asadmin` utility should be protected with file system permissions.

For instructions, see “To Set a Password From a File” on page 41.

Web Browsers and Password Storage

Most web browsers can save login credentials entered through HTML forms. This function can be configured by the user and also by applications that employ user credentials. If the function is enabled, then credentials entered by the user are stored on their local computer and retrieved by the browser on future visits to the same application. This function is convenient for users, but can also be a security risk. The stored credentials can be captured by an attacker who gains access to the computer, either locally or through some remote compromise. Further, methods have existed whereby a malicious web site can retrieve the stored credentials for other applications, by exploiting browser vulnerabilities or through application-level cross-domain attacks.

To prevent your web browser from saving login credentials for the GlassFish Server Administration Console, choose “No” or “Never for this page” when prompted by the browser during login.

Password Aliases

To avoid storing passwords in the domain configuration file in clear text, you can create an alias for a password. This process is also known as encrypting a password. For more information, see "Administering Password Aliases” on page 41.

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 is based on groups. Single sign-on applies to web applications configured for the same realm and virtual server. The realm is defined by the `realm-name` element in the `web.xml` file.

On GlassFish Server, single sign-on behavior can be inherited from the HTTP Service, enabled, or disabled. By default, it is inherited from the HTTP Service. If enabled, single sign-on is
enabled for web applications on this virtual server that are configured for the same realm. If disabled, single sign-on is disabled for this virtual server, and users must authenticate separately to every application on the virtual server.

**Authorization**

*Authorization*, also known as access control, is the means by which users are granted permission to access data or perform operations. After a user is authenticated, the user’s level of authorization determines what operations the owner can perform. A user’s authorization is based on the user’s role.

**Roles**

A *role* defines which applications and what parts of each application users can access and what those users or groups can do with the applications. For example, in a personnel application, all employees might be able to see phone numbers and email addresses, but only managers have access to salary information. This application would 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, the personnel application specify groups such as *full-time*, *part-time*, and *on-leave*. Users in these groups are all employees (the *employee* role). In addition, each user has its own designation that defines an additional level of employment.

Roles are defined in the deployment descriptor for the application. The application developer or deployer maps roles to one or more groups in the deployment descriptor for each application. When the application is being packaged and deployed, the application specifies mappings between users, groups, and roles, as illustrated in the following figure.
Java Authorization Contract for Containers

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 GlassFish Server provides a simple, file-based authorization engine that complies with the JACC specification.

This release includes Administration Console support and CLI subcommands to create (create-jacc-provider), delete (delete-jacc-provider), and list (list-jacc-providers) JACC providers. "Administering JACC Providers" on page 52 for additional information.

You can also specify additional third-party JACC providers.

Working With the server.policy Policy File

Each GlassFish Server domain has its own global Java SE policy file, located in domain-dir/config. The file is named server.policy.
This section covers the following topics:

- “Contents of server.policy” on page 25
- “Changing the Default Permissions” on page 27

Contents of server.policy

A sample server.policy file is as follows. Comments in the file describe why various permissions are granted. These permissions are described in more detail in the next section.

Note – This server.policy file is presented for example purposes only and is subject to change.

```java
// classes in lib get all permissions by default
grant codeBase "file:${com.sun.aas.installRoot}/lib/*" {
   permission java.security.AllPermission;
};

// Core server classes get all permissions by default
grant codeBase "file:${com.sun.aas.installRoot}/modules/*" {
   permission java.security.AllPermission;
};

// Felix classes get all permissions by default
grant codeBase "file:${com.sun.aas.installRoot}/osgi/felix/bin/*" {
   permission java.security.AllPermission;
};

// iMQ classes get all permissions by default
grant codeBase "file:${com.sun.aas.imqLib}/-*" {
   permission java.security.AllPermission;
};

// Derby driver classes get all permissions by default
grant codeBase "file:${com.sun.aas.derbyRoot}/lib/*" {
   permission java.security.AllPermission;
};

// permission for JDK’s tools.jar to enable webservice annotation processing
// at runtime by wsgen tool:
// permission java.lang.RuntimePermission "createClassLoader";
//
// permission for JDK’s tools.jar to sign JARs at runtime for
// Java Web Start support:
// permissions java.security.AllPermission;
// on the advice of the JDK tools folks. Should be refined later.
grant codeBase "file:${com.sun.aas.javaRoot}/lib/tools.jar" {
   permission java.security.AllPermission;
};

// Loading MBeans from anywhere, to take care of side effects of 6235678.
grant {
   permission javax.management.MBeanTrustPermission "register";
};
```
Loading MBeans from anywhere, to take care of side effects of 6235678.

Basic set of required permissions granted to all remaining code
The permission FilePermission "<<ALL FILES>>", "read,write"
allows all applications to read and write any file in the filesystem.
It should be changed based on real deployment needs. If you know your
applications just need to read/write a few directories consider removing
this permission and adding grants indicating those specific directories.
against the codebase of your application(s).

grant {
//Workaround for bugs #6484935, 6513799
permission java.lang.RuntimePermission "getProtectionDomain";
permission com.sun.corba.ee.impl.presentation.rmi.DynamicAccessPermission "access";
permission java.util.PropertyPermission "*", "read,write";

permission java.lang.RuntimePermission "loadLibrary.*";
permission java.lang.RuntimePermission "queuePrintJob";
permission java.net.SocketPermission "*", "connect";
permission java.io.FilePermission "<<ALL FILES>>", "read,write";

// work-around for pointbase bug 4864405
permission java.io.FilePermission
"${com.sun.aas.instanceRoot}${/}lib${/}databases${/}" -
"delete";
permission java.io.FilePermission "${java.io.tmpdir}${/}" -
"delete";

permission java.util.PropertyPermission "*", "read";

permission java.lang.RuntimePermission "modifyThreadGroup";
permission java.lang.RuntimePermission "getClassLoader";
permission java.lang.RuntimePermission "setContextClassLoader";
permission javax.management.MBeanPermission
"[com.sun.messaging.jms.*:*"]", "*";
}

Following grant block is only required by Connectors. If Connectors
are not in use the recommendation is to remove this grant.
grant {
  permission javax.security.auth.PrivateCredentialPermission
  "javax.resource.spi.security.PasswordCredential * /*", "read";
}

Following grant block is only required for Reflection. If Reflection
is not in use the recommendation is to remove this section.
grant {
  permission java.lang.RuntimePermission "accessDeclaredMembers";
}

Permissions to invoke CORBA objects in server
grant {
  permission com.sun.enterprise.security.CORBAObjectPermission "+", "*";
}
Changing the Default Permissions

The GlassFish Server internal server code is granted all permissions. These grants are covered by the `AllPermission` grant blocks to various parts of the server infrastructure code. Do not modify these entries.

Application permissions are granted in the default grant block. These permissions apply to all code not part of the internal server code listed previously.

The last section, beginning with the comment "Basic set of required permissions..." provides the basic set of permissions granted to all remaining code.

Depending on your GlassFish Server implementation, deleting or modifying these permissions might be appropriate.

Specifically, the following permission allows all applications to read and write all properties and read and write all files on the filesystem.

```java
permission java.util.PropertyPermission "*", "read,write";
permission java.io.FilePermission "<ALL FILES>\", "read,write";
```

While this grant provides optimum flexibility, it is inherently unsecure. For enhanced security, change this permission based on your real deployment needs.

For example, consider removing this permission and assign default read and write permissions only to the application's install directory (context-root). (This example uses `com.sun.aas.instanceRoot`, which specifies the top level directory for a server instance.)

```java
grant codeBase "file:${com.sun.aas.instanceRoot}/applications/MyApp/-
{ permission java.io.FilePermission "file:${com.sun.aas.instanceRoot}
/applications/MyApp/\", "read,write";
}
```

For any application that needs to read and write additional directories, you would then have to explicitly allow such permissions by adding specific grants. In general, you should add extra permissions only to the applications or modules that require them, not to all applications deployed to a domain.

Additional permissions (see the embedded comments in `server.policy`) are granted specifically for using connectors and reflection. If connectors or reflection are not used in a particular domain, you should remove these permissions, because they are otherwise unnecessary.
Auditing

*Auditing* is the means used to capture security-related events for the purpose of evaluating the effectiveness of security measures. GlassFish Server uses audit modules to capture audit trails of all authentication and authorization decisions. GlassFish Server provides a default audit module, as well as the ability to plug in custom audit modules. The scope of the audit module is the entire server, which means that all the applications on the server will use the same audit module.

For administration instructions, see “Administering Audit Modules” on page 45.

Firewalls

*A firewall* controls the flow of data between two or more networks, and manages the links between the networks. A firewall can consist of both hardware and software elements. The following guidelines pertain primarily to GlassFish Server:

- In general, firewalls should be configured 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 8081, you must configure the firewalls to allow HTTPS requests on port 8081.

- 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 GlassFish Server behind the firewall.

Certificates and SSL

The following topics are addressed here:

- “Certificates” on page 29
- “Certificate Chains” on page 29
- “Certificate Files” on page 30
- “Secure Sockets Layer” on page 30
“Custom Authentication of Client Certificate in SSL Mutual Authentication” on page 32

For administration instructions, see “Administering JSSE Certificates” on page 47.

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

A certificate chain is a series of certificates issued by successive CA certificates, eventually ending in a root CA certificate.
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.

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

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

### Certificate Files

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

- **Keystore file**  
  The `keystore.jks` file contains GlassFish Server certificate, including its private key. The keystore file is protected with a password. Each keystore entry has a unique alias. After installation, the GlassFish Server keystore has a single entry with an alias of `s1as`.

- **Truststore file**  
  The `cacerts.jks` file contains the GlassFish 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, GlassFish 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.

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

1. The browser sends a message over the network requesting a secure session (typically, by requesting a URL that begins with https instead of http).
2. The server responds by sending its certificate (including its public key).
3. The browser verifies that the server’s certificate is valid and is signed by a CA whose certificate is in the browser’s database (and who is trusted). It also verifies that the CA certificate has not expired.
4. If the certificate is valid, the browser generates a one time, unique session key and encrypts it with the server’s public key. The browser then sends the encrypted session key to the server so that they both have a copy.
5. The server decrypts the message using its private key and recovers the session key.

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

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

To use SSL, GlassFish 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. For instructions on applying SSL to HTTP listeners, see “To Configure an HTTP Listener for SSL” in Oracle GlassFish Server 3.1 Administration Guide.

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. During a secure connection, the client and the server agree to use the strongest cipher that they both have enabled for communication, so it is usually sufficient to enable all ciphers.
Name-based Virtual Hosts

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

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

Custom Authentication of Client Certificate in SSL Mutual Authentication

Release 3.1 of GlassFish Server extends the Certificate realm to allow custom authentication and group assignment based on the client certificate received as part of SSL mutual (two-way) authentication.

As in previous releases, you can create only one certificate realm. However, you can now use a convenient abstract base class to configure a JAAS LoginModule for the Certificate realm. Specifically, your LoginModule can now extend com.sun.appserv.security.AppservCertificateLoginModule. When you do this, you need to implement only the authenticateUser method and call the commitUserAuthentication method to signify success.

This section describes the following topics:

- “Understanding the AppservCertificateLoginModule Class” on page 32
- “Example AppservCertificateLoginModule Code” on page 33
- “Setting the JAAS Context” on page 34

Understanding the AppservCertificateLoginModule Class

The AppservCertificateLoginModule class provides some convenience methods for accessing the certificates, the application name and so forth, and for adding the group principals to the subject. The convenience methods include the following:

getName()

Returns the name of the application to be authenticated. This may be useful when a single LoginModule has to handle multiple applications that use certificates.
getCerts()
    Returns the certificate chain as an array of java.security.cert.X509Certificate
certificates.

getX500Principal()
    Returns the Distinguished principal from the first certificate in the chain.

getSubject()
    Returns the subject that is being authenticated.

commitUserAuthentication(final String[] groups)
    This method sets the authentication status to success if the groups parameter is non-null.
    Note that this method is called after the authentication has succeeded. If authentication
failed, do not call this method.

See the Javadoc at AppservCertificateLoginModule(http://javadoc.glassfish.org/v3/apidoc/com/sun/appserv/security/AppservCertificateLoginModule.html) for
complete information.

Note – You do not have to extend the convenience base class, you can extend the JAAS
LoginModule javax.security.auth.spi.LoginModule instead if you so choose. (See
Implement the Abstract LoginModule Methods(http://download.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASLMDevGuide.html#Step%203) for
additional information.)

Example AppservCertificateLoginModule Code

"Example AppservCertificateLoginModule Code" on page 33 shows a sample instance of the
AppservCertificateLoginModule class.

Note – This sample code is part of a sample AppservCertificateLoginModule project
(http://blogs.sun.com/nasradu8/resource/certRealm/certificateLM.zip). See the
complete sample project for information on how to build and run the sample.

Take note of the following points from the example:

- The getX500Principal() method returns the subject (subject distinguished name) value
  from the first certificate in the client certificate chain as an X500Principal.
- From that X500Principal, the getName() method then returns a string representation of
  the X.500 distinguished name using the format defined in RFC 2253.
- The example uses the getAppName() method to determine the application name. It also
determines the organizational unit (OU) from the distinguished name.
- The example concatenates the application name with the value of OU, and uses it as the group
  name in the commitUserAuthentication method.
EXAMPLE 1-1  Sample AppservCertificateLoginModule Code
/**
 * @author nasradu8
 */
public class CertificateLM extends AppservCertificateLoginModule {

    @Override
    protected void authenticateUser() throws LoginException {
        // Get the distinguished name from the X500Principal.
        String dname = getX500Principal().getName();
        StringTokenizer st = new StringTokenizer(dname, "B \t\n\r\f,");
        while (st.hasMoreTokens()) {
            String next = st.nextToken();
            // Set the appname:OU as the group.
            // At this point, one has the application name and the DN of
            // the certificate. A suitable login decision can be made here.
            if (next.startsWith("OU=")) {
                commitUserAuthentication(new String[] {getAppName() + ":" + next.substring(3)});
                return;
            }
        }
        throw new LoginException("No OU found.");
    }
}

Setting the JAAS Context
After you create your LoginModule, you must plug it in to a jaas-context, which you then
specify as a parameter to the certificate realm in GlassFish Server.

To do this, perform the following steps:
1. Specify a new jaas-context for the Certificate realm in the file
   <domain-dir>/config/login.conf. For example, using the CertificateLM class from
   "Example AppservCertificateLoginModule Code" on page 33:
   certRealm {
   };
2. Specify this jaas-context as a parameter to the set subcommand in the
   configs.config.server-config.security-service.auth-realm.certificate.property.
   jaas-context=<jaas-context-name> property. For example:
   asadmin> set
   configs.config.server-config.security-service.auth-realm.certificate.property.
   jaas-context=certRealm
   configs.config.server-config.security-service.auth-realm.certificate.property.
   jaas-context=certRealm
   Command set executed successfully.
3. Optionally, get the value you just set to make sure that it is correct.
Tools for Managing System Security

GlassFish Server provides the following tools for managing system security:

Administration Console

The Administration 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 Administration Console, see “Administration Console” in Oracle GlassFish Server 3.1 Administration Guide.

The asadmin utility

The asadmin command-line utility performs many of the same tasks as the Administration Console. You might be able to do some things with the asadmin utility that you cannot do with the Administration Console. For a general introduction to asadmin, see “asadmin Utility” in Oracle GlassFish Server 3.1 Administration Guide.

The keytool utility

The keytool Java Platform, Standard Edition (Java SE) command-line utility is used for managing digital certificates and key pairs. For more information, see “Administering JSSE Certificates” on page 47.

The policytool utility

The policytool Java SE graphical utility is used for managing system-wide Java security policies. As an administrator, you rarely use policytool.

Administering Passwords

There are multiple ways to administer passwords. You can rely on administrators to keep passwords secret and change the passwords regularly. You can set up files for storing passwords so that asadmin subcommands can access these files rather than having users type the commands. You can encrypt passwords by setting up aliases so that sensitive passwords are not visible in the domain.xml file.

The following topics are addressed here:

- “To Change the Master Password” on page 36
- “Additional Considerations for the start-instance and start-cluster Subcommands” on page 37
- “Using start-instance and start-cluster With a Password File” on page 38
- “To Change an Administration Password” on page 40
- “To Set a Password From a File” on page 41
- “Administering Password Aliases” on page 41

\section*{To Change the Master Password}

The master password gives access to the keystore used with the domain. This password is not tied to a UNIX user. You should treat this overall shared password as sensitive data. GlassFish Server never uses it for authentication and never transmits it over the network.

You can choose to type the password manually when required, or to obscure the password in a password file. If there is no password file, you are prompted for the master password. If there is a password file, but you want to change access to require prompting, remove the file. The default master password is changeit.

When changing the master password, it has to be changed on all nodes as well as on the DAS. The master password on nodes is only stored once in the node, for all instances that are on that node.

Use the change-master-password subcommand in local mode to modify the master password.

\textbf{Note} – If you change the master password and are not using a master password file, the start-instance and start-cluster subcommands are not able to determine the master password. In this case, you must start those instances locally by using start-local-instance.

When the master password is saved, it is saved in the master-password file.

\textbf{Before You Begin}  This subcommand will not work unless the domain is stopped.
1. **Stop the domain whose password you are changing.**
   See “To Stop a Domain” in Oracle GlassFish Server 3.1 Administration Guide.

2. **Change the master password for the domain by using the `change-master-password(1)` subcommand.**
   You are prompted for the old and new passwords. All dependent items are re-encrypted.

3. **Start the domain.**
   See "To Start a Domain" in Oracle GlassFish Server 3.1 Administration Guide.

**Example 1–2 Changing the Master Password**

The `change-master-password` subcommand is interactive in that you are prompted for the old master password as well as the new master password. This example changes the master password for `domain44ps`:

```
asadmin$ change-master-password domain44ps
```

If you have already logged into the domain using the `login(1)` subcommand, you are prompted for the new master password:

```
Please enter the new master password>
Please enter the new master password again>
```

If you are not logged into the domain, you are prompted for both the old and the new master passwords:

```
Please enter the master password>
Please enter the new master password>
Please enter the new master password again>
```

Information similar to the following is displayed:

```
Master password changed for domain44ps
```

**See Also**
You can also view the full syntax and options of the subcommand by typing `asadmin --help change-master-password` at the command line.

**Additional Considerations for the start-instance and start-cluster Subcommands**

If you change the master password for DAS, the `start-domain` and `start-local-instance` subcommands allow you to provide it during domain or instance startup in one of three ways:
Administering Passwords

- Via the master-password file
- By entering it interactively
- Via the asadmin passwordfile

The start-instance and start-cluster subcommands are more problematic. If you create a domain with a master password other than the default, an associated remote instance or cluster must have access to the master password in order to start. However, for security reasons GlassFish Server never transmits the master password or the master password file over the network.

Consider the following scenario:

1. Change the master password on the DAS and save it with --savemasterpassword.
2. Create an instance on another host using the subcommand create-instance. GlassFish Server copies the keystore and truststore from the DAS to the instance, but it does not copy the master password file.
3. Try to start the instance using the start-instance subcommand. An error results.

The start-instance command is looking for the file master-password in the node directory on the instance machine, and it is not there by default. Therefore, the subcommand fails.

You can use the change-master-password subcommand to make sure the correct password is used in this password file, as described in “Using start-instance and start-cluster With a Password File” on page 38.

**Note** – The start-instance and start-cluster subcommands do not include any other way for you to provide the password. If you change the master password and are not using a master password file, the start-instance and start-cluster subcommands are not able to determine the master password. In this case, you must start the instances locally by using start-local-instance.

**▼ Using start-instance and start-cluster With a Password File**

Assume that you have changed the master password on the DAS and you want to make the same change for all instances.

The start-instance and start-cluster subcommands automatically use the master password file if it exists in the instance filesystem. You can use the change-master-password subcommand to make sure the password file exists and that the correct password is used.
1. From the DAS, create a domain and set the master password.
   asadmin> create-domain --savemasterpassword true domain-name

2. Start the domain.
   asadmin> start-domain domain-name

3. Create a node that is enabled for communication over secure shell (SSH).
   asadmin> create-node-ssh --nodehost host-name --installdir /some-dir node-name

4. Create an instance on the node.
   asadmin> create-instance --node node-name instance-name

5. Before you start the instance, on the instance machine run change-master-password with the
   --savemasterpassword option to create a file called master-password in the agents directory
   to access the keystores. (The start-instance subcommand is looking for a file called
   master-password in the agents directory to access the stores.)
   asadmin> change-master-password --savemasterpassword true --nodedir /some-dir node-name

   You are prompted to enter the current and new master password:
   Enter the current master password>
   Enter the new master password>
   Enter the new master password again>
   Command change-master-password executed successfully.

   Remember that when you created the domain you specified a new master password. This
   master password was then used to encrypt the keystore and truststore for the DAS, and these
   stores were copied to the instance as a result of the create-instance subcommand.

   Therefore, enter the master password you set when you created the domain as both the current
   master password and again as the new master password. You enter it as the new master
   password because you do not want to change the master password for the instance and make it
   out of sync with the DAS.

6. Run start-instance from the DAS.
   asadmin> start-instance instance-name

   The master password file is associated with the node and not with an instance. After the master
   password file exists in the node directory on the instance machine, additional instances can be
   created, started and stopped from the DAS.
To Change an Administration Password

Use the change-admin-password subcommand in remote mode to change an administration password. The default administration password is admin. You are prompted for the old and new admin passwords, with confirmation. The passwords are not echoed to the display.

**Note** – If you accepted the default admin user with no password during zip installation, you can add a password to this user. If there is a single user called admin that does not have a password, you are not prompted for login information. Any other situation requires login.

Encrypting the admin password is strongly encouraged.

Before You Begin

If you want to change the admin password before creating an alias for the password (encrypting), you can use the set subcommand with syntax similar to the following:

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

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

2  **Change the admin password by using the change-admin-password(1) subcommand.**

3  **Enter the old and new admin passwords when prompted.**

4  **Restart GlassFish Server.**
See “To Restart a Domain” in Oracle GlassFish Server 3.1 Administration Guide.

Example 1–3  **Changing the Admin Password**

This example changes the admin password for user anonymous from admin to newadmin:

```
asadmin> change-admin-password --username anonymous
```

You are prompted to enter the old and the new admin passwords:

```
Enter admin password> admin
Enter new admin password> newadmin
Enter new admin password again> newadmin
```

Information similar to the following is displayed:

```
Command change-admin-password executed successfully.
```
You can also view the full syntax and options of the subcommand by typing `asadmin help change-admin-password` at the command line.

To Set a Password From a File

Instead of typing the password at the command line, you can access the password for a command from a file such as `passwords.txt`. The `--passwordfile` option of the `asadmin` utility takes the name of the file that contains the passwords. The entry for a password in the file must have the `AS_ADMIN_` prefix followed by the password name in uppercase letters.

The following other types of passwords can be specified:

- `AS_ADMIN_MASTERPASSWORD`
- `AS_ADMIN_USERPASSWORD`
- `AS_ADMIN_ALIASPASSWORD`

1. **Edit the password file.**
   
   For example, to specify the password for the domain administration server (DAS), add an entry similar to the following to the password file, where `adminadmin` is the administrator password:
   
   ```
   AS_ADMIN_PASSWORD=adminadmin
   ```

2. **Save the password file.**

   You can now specify the password file in an `asadmin` subcommand. In this example, `passwords.txt` is the file that contains the password:
   
   ```
   asadmin>delete-jdbc-resource --user admin --password passwords.txt jdbc/DerbyPool
   ```

**Troubleshooting**

If `AS_ADMIN_PASSWORD` has been exported to the global environment, specifying the `--passwordfile` option will produce a warning about using the `--passwordfile` option. To prevent this warning situation from happening, unset `AS_ADMIN_PASSWORD`.

Administering Password Aliases

A **password alias** is used to indirectly access a password so that the password itself does not appear in cleartext in the domain's `domain.xml` configuration file.

Storing passwords in cleartext format in system configuration files is common in many open source projects. In addition to GlassFish Server, Apache Tomcat, Maven, and Subversion, among others, store and pass passwords in cleartext format. However, storing and passing passwords in cleartext can be a security risk, and may violate some corporate security policies. In such cases, you can use password aliases.
To Create a Password Alias

Use the `create-password-alias` subcommand in remote mode to create an alias for a password in the domain's keystore. The password corresponding to the alias name is stored in an encrypted form in the domain configuration file. The `create-password-alias` subcommand takes both a secure interactive form, in which users are prompted for all information, and a more script-friendly form, in which the password is propagated on the command line.

You can also use the `set(1)` subcommand to remove and replace the password in the configuration file. For example:

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

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

2. **Go to the directory where the configuration file resides.**
   By default, the configuration file is located in `domain-dir/config`.

3. **Create the password alias by using the `create-password-alias(1)` subcommand.**

4. **Type the password for the alias when prompted.**

5. **Add the alias to a password file.**
   For example, assume the use of a password file such as `passwords.txt`. Assume further that you want to add an alias for the `AS_ADMIN_USERPASSWORD` entry that is read by the `create-file-user(1)` subcommand. You would add the following line to the password file:
   `AS_ADMIN_USERPASSWORD=${ALIAS=user-password-alias}`, where `user-password-alias` is the new password alias.

6. **To continue the example of the previous step, you would then run the `create-file-user(1)` subcommand.**
   You could use this method to create several users (`user1, user2, and so forth`), all with the same password.
Creating a Password Alias

This example creates the new jms-password alias for the admin user:

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

You are prompted to type the password for the alias:

```
Please enter the alias password>
```

```
secret-password
```

```
Please enter the alias password again>
```

```
secret-password
```

Command create-password-alias executed successfully.

You can also view the full syntax and options of the subcommand by typing asadmin help create-password-alias at the command line.

To List Password Aliases

Use the list-password-aliases subcommand in remote mode to list existing the password aliases.

1 Ensure that the server is running.
Remote subcommands require a running server.

2 List password aliases by using the list-password-aliases(1) subcommand.

Listing Password Aliases

This example lists the existing password aliases:

```
asadmin> list-password aliases
jmspassword-alias
```

Command list-password-aliases executed successfully

You can also view the full syntax and options of the subcommand by typing asadmin help list-password-aliases at the command line.

To Delete a Password Alias

Use the delete-password-alias subcommand in remote mode to delete an existing password alias.
1 Ensure that the server is running.
   Remote subcommands require a running server.

2 List all aliases by using the `list-password-aliases(1)` subcommand.

3 Delete a password alias by using the `list-password-aliases(1)` subcommand.

**Example 1–6 Deleting a Password Alias**

This example deletes the password alias `jmspassword-alias`:

```
asadmin> delete-password-alias jmspassword-alias
Command list-password-aliases executed successfully
```

**See Also** You can also view the full syntax and options of the subcommand by typing `asadmin help delete-password-alias` at the command line.

**To Update a Password Alias**

Use the `update-password-alias` subcommand in remote mode to change the password for an existing password alias. The `update-password-alias` subcommand takes both a secure interactive form, in which the user is prompted for all information, and a more script-friendly form, in which the password is propagated on the command line.

1 Ensure that the server is running.
   Remote subcommands require a running server.

2 Update an alias by using the `update-password-alias(1)` subcommand.

3 Type the password when prompted.

**Example 1–7 Updating a Password Alias**

This example updates the password for the `jmspassword-alias` alias:

```
asadmin> update-password-alias jmspassword-alias
You are prompted to type the new password for the alias:

Please enter the alias password> new-secret-password
Please enter the alias password again> new-secret-password
Command update-password-alias executed successfully
```
You can also view the full syntax and options of the subcommand by typing `asadmin help update-password-alias` at the command line.

Administering Audit Modules

The following topics are addressed here:

- “To Create an Audit Module” on page 45
- “To List Audit Modules” on page 45
- “To Delete an Audit Module” on page 46

▼ To Create an Audit Module

Use the `create-audit-module` subcommand in remote mode to create an audit module for the add-on component that implements the audit capabilities.

1 Ensure that the server is running.
Remote subcommands require a running server.

2 Create an audit module by using the `create-audit-module(1)` subcommand.
Information about properties for this subcommand is included in this help page.

Example 1–8 Creating an Audit Module

This example creates an audit module named `sampleAuditModule`:

```bash
asadmin> create-audit-module
    --classname com.sun.appserv.auditmodule --property defaultuser=admin:Password=admin
    sampleAuditModule
Command create-audit-module executed successfully.
```

See Also You can also view the full syntax and options of the subcommand by typing `asadmin help create-audit-module` at the command line.

▼ To List Audit Modules

Use the `list-audit-modules` subcommand in remote mode to list the audit modules on one of the following targets:

- Server instance, `server` (the default)
- Specified server instance
Administering Audit Modules

- Specified configuration

1. Ensure that the server is running.
   Remote subcommands require a running server.

2. List the audit modules by using the `list-audit-modules(1)` subcommand.

Example 1–9
Listing Audit Modules
This example lists the audit modules on localhost:

```
asadmin> list-audit-modules
audit-module : default
audit-module : sampleAuditModule
Command list-audit-modules executed successfully.
```

See Also
You can also view the full syntax and options of the subcommand by typing `asadmin help list-audit-modules` at the command line.

▼ To Delete an Audit Module

Use the `delete-audit-module` subcommand in remote mode to delete an existing audit module.

1. Ensure that the server is running.
   Remote subcommands require a running server.

2. List the audit modules by using the `list-audit-modules(1)` subcommand.

3. Delete an audit module by using the `delete-audit-module(1)` subcommand.

Example 1–10
Deleting an Audit Module
This example deletes `sampleAuditModule`:

```
asadmin> delete-audit-module sampleAuditModule
Command delete-audit-module executed successfully.
```
Administering JSSE Certificates

In the developer profile, the GlassFish Server 3.1 uses the JSSE format on the server side to manage certificates and key stores. In all profiles, the client side (appclient or stand-alone) uses the JSSE format.

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

The following topics are addressed here:

- “To Generate a Certificate by Using keytool” on page 47
- “To Sign a Certificate by Using keytool” on page 49
- “To Delete a Certificate by Using keytool” on page 52

▼ To Generate a Certificate by Using keytool

By default, the keytool utility creates a keystore file in the directory where the utility is run.

Before You Begin

To run the keytool utility, your shell environment must be configured so that the Java SE /bin directory is in the path, otherwise the full path to the utility must be present on the command line.

1 Change to the directory that contains the keystore and truststore files.

Always generate the certificate in the directory containing the keystore and truststore files. The default is domain-dir/config.

2 Generate the certificate in the keystore file, keystore.jks, using the following command format:

   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 the default (changeit), substitute the new password for changeit. The default key password alias is s1as.

   A prompt appears that asks for your name, organization, and other information.
3 Export the generated certificate to the server.cer file (or client.cer if you prefer), using the following command format:

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

4 If a certificate signed by a certificate authority is required, see “To Sign a Certificate by Using keytool” on page 49.

5 Create the cacerts.jks truststore file and add the certificate to the truststore, using the following command format:

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

   If you have changed the keystore or private key password from the default (changeit), substitute the new password.

   Information about the certificate is displayed and a prompt appears asking if you want to trust the certificate.

6 Type yes, then press Enter.

   Information similar to the following is displayed:

   Certificate was added to keystore
   [Saving cacerts.jks]

7 To apply your changes, restart GlassFish Server. See “To Restart a Domain” in Oracle GlassFish Server 3.1 Administration Guide.

Example 1–11 Creating a Self-Signed Certificate in a JKS Keystore by Using an RSA Key Algorithm

   RSA is public-key encryption technology developed by RSA Data Security, Inc.

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

Example 1–12 Creating a Self-Signed Certificate in a JKS Keystore by Using a Default Key Algorithm

   keytool -genkey -noprompt -trustcacerts -alias ${cert.alias} -dname
   ${dn.name} -keypass ${key.pass} -keystore ${keystore.file} -storepass
   ${keystore.pass}
**To Sign a Certificate by Using keytool**

After creating a certificate, the owner must sign the certificate 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).

**Note** – If authentication is not a concern, for example if private secure communications are all that is required, you can save the time and expense involved in obtaining a CA certificate by using a self-signed certificate.

1. **Delete the default self-signed certificate:**
   ```bash
   keytool -delete -alias sqlas -keystore keystore.jks -storepass <store_passwd>
   ```
   where `<store_passwd>` is the password for the keystore. For example, "mypass". Note that `sqlas` is the default alias of the GlassFish Server keystore.

2. **Generate a new key pair for the application server:**
   ```bash
   keytool -genkeypair -keyalg <key_alg> -keystore keystore.jks
   -validity <val_days> -alias sqlas
   ```
   where `<key_alg>` is the algorithm to be used for generating the key pair, for example RSA, and `<val_days>` is the number of days that the certificate should be considered valid. For example, 365.

   In addition to generating a key pair, the command wraps the public key into a self-signed certificate and stores the certificate and the private key in a new keystore entry identified by the alias.

   For HTTPS hostname verification, it is important to ensure that the name of the certificate (CN) matches the fully-qualified hostname of your site (fully-qualified domain name). If the
names do not match, clients connecting to the server will see a security alert stating that the name of the certificate does not match the name of the site.

3 Generate a Certificate Signing Request (CSR):

   keytool -certreq -alias s1as -file <certreq_file> -keystore keystore.jks
   -storepass <store_passwd>

   where <certreq_file> is the file in which the CSR is stored (for example, s1as.csr) and <store_passwd> is the password for the keystore. For example, change it.

4 Submit the CSR to a Certificate Authority such as VeriSign http://www.verisign.com/ssl/buy-ssl-certificates/index.html. In response, you should receive a signed server certificate. Make sure to import into your browser the CA certificate of the CA (if not already present) and any intermediate certificates indicated by the CA in the reply.

5 Store the signed server certificate from the CA, including the markers -----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----, into a file such as s1as.cert. Download the CA certificate and any intermediate CA certificates and store them in local files.

6 Import the CA certificate (if not already present) and any intermediate CA certificates (if not already present) indicated by the CA into the truststore cacerts.jks:

   keytool -import -v -trustcacerts -alias <CA-Name> -file ca.cert
   -keystore cacerts.jks -storepass <store_passwd>

7 Replace the original self-signed certificate with the certificate you obtained from the CA, as stored in a file such as s1as.cert:

   keytool -import -v -trustcacerts -alias s1as -file s1as.cert
   -keystore keystore.jks -storepass <store_passwd>

   When you import the certificate using the same original alias s1as, keytool treats it as a command to replace the original certificate with the certificate obtained as a reply to a CSR.

   After running the command, you should see that the certificate s1as in the keystore is no longer the original self-signed certificate, but is now the response certificate from the CA.

   Consider the following example that compares an original s1as certificate with a new s1as certificate obtained from VeriSign:

   Original s1as (self-signed):

   Owner: CN=FQDN, OU=Sun Java System Application Server, O=Sun Microsystems, L=Santa Clara, ST=California, C=US
   Issuer: CN=KUMAR, OU=Sun Java System Application Server, O=Sun Microsystems, L=Santa Clara, ST=California, C=US
   Serial number: 472acd34
Valid from: Fri Nov 02 12:39:40 GMT+05:30 2007 until: Mon Oct 30 12:39:40 GMT+05:30 2017

New s1as (contains signed cert from CA):

Owner: CN=FQDN, OU=Terms of use at www.verisign.com/cps/test ca (c)05, OU=Sun Java System Application Server, O=Sun Microsystems, L=Santa Clara, ST=California, C=US
Issuer: CN=VeriSign Trial Secure Server Test CA, OU=Terms of use at https://www.verisign.com/cps/testca (c)05, OU="For Test Purposes Only. No assurances.", O=VeriSign, Inc., C=US
Serial number: 1375de18b223908c2cb0123059d5c440
Valid from: Sun Nov 11 05:30:00 GMT+05:30 2007 until: Mon Nov 26 05:29:59 GMT+05:30 2007

8 To apply your changes, restart GlassFish Server.
See “To Restart a Domain” in Oracle GlassFish Server 3.1 Administration Guide.

Example 1–15 Importing an RFC/Text-Formatted Certificate Into a JKS Keystore

Certificates are often stored using the printable encoding format defined by the Internet Request for Comments (RFC) 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}
```

Example 1–16 Exporting a Certificate From a JKS Keystore 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}
```

Example 1–17 Exporting a Certificate From a JKS Keystore in RFC/Text Format

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

See Also For more information about keytool, see the keytool reference page (http://download.oracle.com/docs/cd/E17409_01/javase/6/docs/technotes/tools/solaris/keytool.html).
**To Delete a Certificate by Using keytool**

Use the keytool -delete command to delete an existing certificate.

- Delete a certificate using the following command format:

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

---

**Example 1–18 Deleting a Certificate From a JKS Keystore**

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

**See Also**

For more information about keytool, see the keytool reference page (http://download.oracle.com/docs/cd/E17409_01/javase/6/docs/technotes/tools/solaris/keytool.html).

---

**Administering JACC Providers**

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

GlassFish Server includes Administration Console support and subcommands to support JACC providers, as follows:

- create create-jacc-provider
- delete delete-jacc-provider
- list list-jacc-providers

The default GlassFish Server installation includes two JACC providers, named *default* and *simple*. You should not delete these default providers. Any JACC providers you create with the create-jacc-provider subcommand are in addition to these two default providers.

The GlassFish Server creates a JSR-115-compliant JACC provider that you can use with third-party authorization modules for applications running in GlassFish Server. The JACC provider is created as a jacc-provider element within the security-service element in the domain's domain.xml file.
Administering JACC Providers From the Administration Console

To use the Administration Console to administer JACC providers, perform the following steps:

1. Select Configurations and expand the entry.
2. Select the server configuration for which you want to administer JACC providers and expand the entry.
3. Select Security and expand the entry.
4. Select JACC Providers. The JACC Providers page is displayed. The existing JACC providers are shown on this page.

To create a new provider, click New.
Enter the Name, Policy Configuration (the class that implements the policy configuration factory) and the Policy Provider (the class that implements the policy factory) for the new JACC provider. You can also enter optional properties (name/value) for the provider.

To delete an existing JACC provider, select that provider and click Delete.
Administering JACC Providers from the Command Line

To use the command line to administer JACC providers, perform the following steps:

1. To create a JACC provider, use the `create-jacc-provider` subcommand. The following example shows how to create a JACC provider named `testJACC` on the default server target.

   ```
   asadmin> create-jacc-provider
   --policyproviderclass com.sun.enterprise.security.provider.PolicyWrapper
   --policyconfigfactoryclass com.sun.enterprise.security.provider.PolicyConfigurationFactoryImpl
   testJACC
   ```

2. To delete a JACC provider, use the `delete-jacc-provider` subcommand. The following example shows how to delete a JACC provider named `testJACC` from the default domain:

   ```
   asadmin> delete-jacc-provider testJACC
   ```

3. To list the available providers, use the `list-jacc-providers` subcommand. The following example shows how to list JACC providers for the default domain:

   ```
   asadmin> list-jacc-providers
   default
   simple
   Command list-jacc-providers executed successfully.
   ```
This chapter provides instructions for administering user security in the Oracle GlassFish Server environment by using the \texttt{ashadmin} command-line utility. GlassFish Server enforces its authentication and authorization policies upon realms, users, and groups. This chapter assumes that you are familiar with security features such as authentication, authorization, and certificates. If you are not, see Chapter 1, “Administering System Security.”

The following topics are addressed here:

- “Administering Authentication Realms” on page 56
- “To Create an Authentication Realm” on page 57
- “To List Authentication Realms” on page 58
- “To Update an Authentication Realm” on page 58
- “To Delete an Authentication Realm” on page 59
- “To Configure a JDBC or Digest Authentication Realm” on page 59
- “To Configure LDAP Authentication with OID” on page 61
- “To configure LDAP Authentication with OVD” on page 62
- “To Enable LDAP Authentication on the GlassFish Server DAS” on page 63

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

**Administering Authentication Realms**

The following topics are addressed here:
Overview of Authentication Realms

An authentication realm, also called a security policy domain or security domain, is a scope over which the GlassFish Server defines and enforces a common security policy. GlassFish Server is preconfigured with the file, certificate, and administration realms. In addition, you can set up LDAP, JDBC, digest, Oracle Solaris, or custom realms. An application can specify which realm to use in its deployment descriptor. If the application does not specify a realm, GlassFish Server uses its default realm (file).

File realm
GlassFish Server stores user credentials locally in a file named keyfile. The file realm is the initial default realm.

Administration realm
The administration realm is also a file realm and stores administrator user credentials locally in a file named admin-keyfile.

Certificate realm
GlassFish 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.

LDAP realm
GlassFish Server can get user credentials from a Lightweight Directory Access Protocol (LDAP) server such as Oracle Virtual Directory (OVD), Oracle Internet Directory (OID), and Oracle Directory Server Enterprise Edition. 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.

See “To Configure LDAP Authentication with OID” on page 61 for instructions on configuring GlassFish Server to work with an OVD/OID LDAP provider.

JDBC realm
GlassFish Server gets user credentials from a database. The server uses the database information and the enabled JDBC realm option in the configuration file.

Digest realm
Digest Authentication authenticates a user based on a user name and a password. However, the authentication is performed by transmitting the password in an encrypted form.

Oracle Solaris realm
GlassFish Server gets user credentials from the Oracle Solaris operating system. This realm is supported on the Oracle Solaris 9 and Oracle Solaris 10 operating systems. Consult your Oracle Solaris documentation for information about managing users and groups in the Oracle Solaris realm.

PAM realm
A Pluggable Authentication Module (PAM) realm allows applications deployed on GlassFish Server to authenticate users against a native Unix (Solaris/Linux/Mac OS) users list. PAM realms
use the class name
com.sun.enterprise.security.auth.realm.pam.PamRealm and
the JAAS Context pamRealm.

This realm is supported on all Unix Operating Systems, including the
Oracle Solaris 9 and Oracle Solaris 10 operating systems

Custom realm

You can create other repositories for user credentials, such as a
relational database or third-party components. For more
information about custom realms, see the Administration Console
online help. For instructions on creating a custom realm, see
“Creating a Custom Realm” in Oracle GlassFish Server 3.1
Application Development Guide.

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

▼ To Create an Authentication Realm

Use the create-auth-realm subcommand in remote mode to create an authentication realm.

1 Ensure that the server is running.
Remote subcommands require a running server.

2 Create a realm by using the create-auth-realm subcommand.
Information about properties for this subcommand is included in this help page.

Example 2–1 Creating a Realm

This example creates a realm named db.

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

See Also You can also view the full syntax and options of the subcommand by typing asadmin help create-auth-realm at the command line.

For information on creating a custom realm, see “Creating a Custom Realm” in Oracle GlassFish Server 3.1 Application Development Guide.
To List Authentication Realms

Use the `list-auth-realms` subcommand in remote mode to list the existing authentication realms.

1. Ensure that the server is running.
   Remote subcommands require a running server.

2. List realms by using the `list-auth-realms` subcommand.

Example 2–2 Listing Realms

This example lists the authentication realms on `localhost`.

```bash
asadmin> list-auth-realms
db
certificate
db
certificate
admin-realm
admin-realm
Command list-auth-realms executed successfully.
```

See Also You can also view the full syntax and options of the subcommand by typing `asadmin help list-auth-realms` at the command line.

To Update an Authentication Realm

Use the `set` subcommand to modify an existing authentication realm.

Note – A custom realm does not require server restart.

1. List realms by using the `list-auth-realms` subcommand.

2. Modify the values for the specified thread pool by using the `set` subcommand.
   The thread pool is identified by its dotted name.

3. To apply your changes, restart GlassFish Server.
   See “To Restart a Domain” in Oracle GlassFish Server 3.1 Administration Guide.
To Delete an Authentication Realm

Use the `delete-auth-realm` subcommand in remote mode to delete an existing authentication realm.

1. Ensure that the server is running.
   Remote subcommands require a running server.

2. List realms by using the `list-auth-realms(1)` subcommand.

3. If necessary, notify users that the realm is being deleted.

4. Delete the realm by using the `delete-auth-realm(1)` subcommand.

5. To apply your changes, restart GlassFish Server. See “To Restart a Domain” in Oracle GlassFish Server 3.1 Administration Guide.

Example 2–3

Deleting a Realm

This example deletes an authentication realm named `db`.

```
asadmin> delete-auth-realm db
Command delete-auth-realm executed successfully.
```

See Also

You can also view the full syntax and options of the subcommand by typing `asadmin help delete-auth-realm` at the command line.

To Configure a JDBC or Digest Authentication Realm

GlassFish 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 type realm instead of the connection pool prevents other applications from browsing the database tables for user 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 user credentials for the realm.
   How you create the database tables depends on the database that you are using.
2 **Add user credentials to the database tables that you created.**
   How you add user credentials to the database tables depends on the database that you are using.

3 **Create a JDBC connection pool for the database.**
   See “To Create a JDBC Connection Pool” in Oracle GlassFish Server 3.1 Administration Guide.

4 **Create a JDBC resource for the database.**
   “To Create a JDBC Resource” in Oracle GlassFish Server 3.1 Administration Guide.

5 **Create a realm.**
   For instructions, see “To Create an Authentication Realm” on page 57.

   **Note** – The JAAS context should be jdbcDigestRealm for digest authentication or jdbcRealm for other authentication types.

6 **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 Oracle GlassFish Server 3.1 Application Development Guide.

7 **Assign security roles 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.

8 **Verify that the database is running.**
   If needed, see “To Start the Database” in Oracle GlassFish Server 3.1 Administration Guide.

9 **To apply the authentication, restart the server.**
   See “To Restart a Domain” in Oracle GlassFish Server 3.1 Administration Guide.
Example 2–4  Assigning a Security Role

This 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>

▼ To Configure LDAP Authentication with OID

This procedure explains how to configure GlassFish Server to use LDAP authentication with Oracle Internet Directory.

1 Install Oracle Enterprise Manager 11g and the latest Enterprise Manager patches, if they are not installed already.
   Instructions for installing Oracle Enterprise Manager are provided in the Oracle Enterprise Manager documentation set.

2 Install the Oracle Identity Management Suite (IDM) 11g and Patch Set 2 or later, if they are not installed already.
   Instructions for installing the Oracle Identity Management suite are provided in the Oracle Fusion Middleware Installation Guide for Oracle Identity Management.

3 Configure SSL for Oracle Internet Directory (OID), if it is not configured already. Configure the OID instance in the server authentication mode and with the protocol version set to SSLv3.
   Instructions for configuring SSL for OID are provided in the SSL chapter of the Oracle Internet Directory Administrator’s Guide.

4 Using Oracle Wallet Manager, export an SSL self-signed certificate you want to use with GlassFish Server.
   Instructions for using Oracle Wallet Manager to create and export SSL certificates are provided in the Configure Oracle Internet Directory for SSL section of the SSL chapter in the Oracle Internet Directory Administrator’s Guide.

5 On the GlassFish Server side, use the keytool command import the certificate you exported with Oracle Wallet Manager.
   The keytool command is available in the $JAVA_HOME/bin directory. Use the following syntax:
   
   keytool -importcert -alias "alias-name" -keystore domain-dir/config/cacerts.jks -file cert-name
   
   where the variables are defined as follows:
To configure LDAP Authentication with OVD

This procedure explains how to configure GlassFish Server to use LDAP authentication with Oracle Virtual Directory.

1 Create the OVD adapter, as described in the Creating and Configuring Oracle Virtual Directory Adapters chapter of the Administrator's Guide for Oracle Virtual Directory (http://download.oracle.com/docs/cd/E12839_01/oid.1111/e10046/toc.htm).

2 Configure SSL for Oracle Virtual Directory (OVD), if it is not configured already. For instructions on configuring SSL for OVD, see the section "Enable SSL for Oracle Virtual Directory Using Fusion.
Middleware Control" in SSL Configuration in Oracle Fusion Middleware (http://download.oracle.com/docs/cd/E12839_01/core.1111/e10105/sslconfig.htm#ASADM1800).

Also, configure the SSL for the OVD listener in server authentication mode.

3 Export the certificate from JKS keystore you want to use with GlassFish Server. See Exporting a Keystore Using Fusion Middleware Control (http://download.oracle.com/docs/cd/E16764_01/core.1111/e10105/wallets.htm#CIHECAIB) for information.

4 On the GlassFish Server side, use the keytool command to import the certificate you exported from the JKS keystore.

The keytool command is available in the $JAVA_HOME/bin directory. Use the following syntax:

```
keytool -importcert -alias "alias-name" -keystore domain-dir/config/cacerts.jks -file cert-name
```

where the variables are defined as follows:

- **alias-name** Name of an alias to use for the certificate
- **domain-dir** Name of the domain for which the certificate is used
- **cert-name** Path to the certificate that you exported from the keystore.

For example, to import a certificate named ovd.cer for a GlassFish Server domain in /glassfishv3/glassfish/domains/domain1, using an alias called "OVD self-signed certificate," you would use the following command:

```
keytool -importcert -alias "OVD self signed certificate" -keystore /glassfishv3/glassfish/domains/domain1/config/cacerts.jks -file ovd.cer
```

5 Restart the GlassFish Server domain.

See "To Restart a Domain" in Oracle GlassFish Server 3.1 Administration Guide.

**To Enable LDAP Authentication on the GlassFish Server DAS**

This procedure explains how to enable LDAP authentication for logins to the GlassFish Server Domain Administration Server (DAS). Logging in to the DAS is typically only performed by GlassFish Server administrators who want to use the GlassFish Server Administration Console or asadmin command. See "To Configure LDAP Authentication with OID" on page 61 for instructions on enabling general LDAP authentication for GlassFish Server.

**Before You Begin**

Ensure that you have followed the configuration instructions in "To Configure LDAP Authentication with OID" on page 61.
Use the `asadmin configure-ldap-for-admin` subcommand to enable user authentication to the GlassFish Server DAS.

Use the following syntax:

```
asadmin configure-ldap-for-admin --basedn "dn-list" --url [ldap|ldaps]://ldap-url --ldap-group group-name
```

where the variables are defined as follows:

- **dn-list**: basedn parameters
- **ldap-url**: URL and port number for the LDAP server; can use standard (ldap) or secure (ldaps) protocol
- **group-name**: LDAP group name for allowed users, as defined on the LDAP server.

For example:

```
asadmin configure-ldap-for-admin --basedn "dc=red,dc=iplanet,dc=com" --url ldap://interopoel54-1:3060 --ldap-group sqestaticgroup
```

```
asadmin configure-ldap-for-admin --basedn "dc=red,dc=iplanet,dc=com" --url ldaps://interopoel54-1:7501 --ldap-group sqestaticgroup
```

See also: See `configure-ldap-for-admin(1)` for more information about the `configure-ldap-for-admin` subcommand.

---

### Administering File Users

A user is an individual (or application program) identity that is defined in GlassFish Server. A user who has been authenticated is sometimes called a principal.

As the administrator, you are responsible for integrating users into the GlassFish 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 topics are addressed here:

- "To Create a File User" on page 65
- "To List File Users" on page 65
- "To List File Groups" on page 66
- "To Update a File User" on page 67
- "To Delete a File User" on page 67
To Create a File User

Use the create-file-user subcommand in remote mode to create a new user by adding a new entry to the keyfile. The entry includes the user name, password, and any 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 subcommands require a running server.

2 If the user will belong to a particular group, see the current groups by using the list-file-groups(1) subcommand.

3 Create a file user by using the create-file-user(1) subcommand.

Example 2–5 Creating a User

This example create user Jennifer on the default realm file (no groups are specified).

The asadmin --passwordfile option specifies the name of a file that contains the password entries in a specific format. The entry for a password must have the AS_ADMIN_ prefix followed by the password name in uppercase letters, an equals sign, and the password. See asadmin(1M) for more information.

```
asadmin> create-file-user --user admin
   --passwordfile=c:\tmp\asadminpassword.txt Jennifer
Command create-file-user executed successfully.
```

See Also You can also view the full syntax and options of the subcommand by typing asadmin help create-file-user at the command line.

To List File Users

Use the list-file-users subcommand in remote mode to list the users that are in the keyfile.

1 Ensure that the server is running.
   Remote subcommands require a running server.

2 List users by using the list-file-users(1) subcommand.
**Example 2–6  Listing File Users**

This example lists file users on the default file realm file.

```
asadmin> list-file-users
Jennifer
Command list-file-users executed successfully.
```

**See Also** You can also view the full syntax and options of the subcommand by typing `asadmin help list-file-users` at the command line.

**To List File Groups**

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, and the big spenders might also 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 and realm. A user can be associated with multiple groups of users.

A group is different from a role 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 all employees (the employee role). In addition, each user has its own designation that defines an additional level of employment.

Use the `list-file-groups` subcommand in remote mode to list groups for a file user, or all file groups if the `--name` option is not specified.

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

2. **List file groups by using the `list-file-groups` subcommand.**

**Example 2–7 Listing Groups for a User**

This example lists the groups for user joesmith.

```
asadmin> list-file-groups --name joesmith
staff
manager
Command list-file-groups executed successfully
```
To Update a File User

Use the `update-file-user` subcommand in remote mode to modify the information in the keyfile for a specified user.

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

2. **Update the user information by using the `update-file-user(1)` subcommand.**

3. **To apply your changes, restart GlassFish Server.**
   See “To Restart a Domain” in *Oracle GlassFish Server 3.1 Administration Guide.*

**Example 2–8**  
Updating a User

The following subcommand updates the groups for user Jennifer.

```
$ asadmin> update-file-user --passwordfile c:\tmp\asadminpassword.txt --groups staff:manager:engineer Jennifer
Command update-file-user executed successfully.
```

**See Also**  
You can also view the full syntax and options of the subcommand by typing `asadmin help update-file-user` at the command line.

To Delete a File User

Use the `delete-file-user` subcommand in remote mode to remove a user entry from the keyfile by specifying the user name. You cannot delete yourself, that is, the user you are logged in as cannot be deleted during your session.

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

2. **List users by using the `list-file-users(1)` subcommand.**

3. **Delete the user by using the `delete-file-user(1)` subcommand.**

**Example 2–9**  
Deleting a User

This example deletes user Jennifer from the default file realm.
asadmin> delete-file-user Jennifer
Command delete-file-user executed successfully.

See Also  You can also view the full syntax and options of the subcommand by typing asadmin help delete-file-user at the command line.
This chapter provides information and procedures on configuring the message layer security for web services in the GlassFish Server environment.

>Note – Message security (JSR 196) is supported only in the Full Platform Profile of GlassFish Server, not in the Web Profile.

The following topics are addressed here:
- “About Message Security in GlassFish Server” on page 69
- “Enabling Default Message Security Providers for Web Services” on page 75
- “Configuring Message Protection Policies” on page 76
- “Administering Non-default Message Security Providers” on page 80
- “Enabling Message Security for Application Clients” on page 83
- “Additional Information About Message Security” on page 84

Some of the material in this chapter assumes a basic understanding of security and web services concepts. For more information about security, see “About System Security in GlassFish Server” on page 17.

Instructions for accomplishing the tasks in this chapter by using the Administration Console are contained in the Administration Console online help.

**About Message Security in GlassFish Server**

*Message security* enables a server to perform end-to-end authentication of web service invocations and responses at the message layer. Security information is inserted into messages so that it travels through the networking layers and arrives with the intact message at the message destination(s). Message security differs from transport layer security in that message security can be used to decouple message protection from message transport so that messages remain protected after transmission.
Web services deployed on GlassFish Server are secured by binding SOAP layer message security providers and message protection policies to the containers in which the applications are deployed, or to web service endpoints served by the applications. SOAP layer message security functionality is configured in the client-side containers of GlassFish Server by binding SOAP layer message security providers and message protection policies to the client containers or to the portable service references declared by client applications.

Message-level security can be configured for the entire GlassFish Server or for specific applications or methods. Configuring message security at the application level is discussed in the Oracle GlassFish Server 3.1 Application Development Guide.

The following topics are addressed here:

- “Security Tokens and Security Mechanisms” on page 70
- “Authentication Providers” on page 71
- “Message Protection Policies” on page 72
- “Application-Specific Web Services Security” on page 72
- “Message Security Administration” on page 73
- “Sample Application for Web Services” on page 74

Security Tokens and Security Mechanisms

WS-Security is a specification that provides a communications protocol for applying security to web services. The security mechanisms implement the specification. Web Services Interoperability Technologies (WSIT) implements WS-Security so as to provide interoperable message content integrity and confidentiality, even when messages pass through intermediary nodes before reaching their destination endpoint. WS-Security as provided by WSIT is in addition to existing transport-level security, which can still be used.

The Simple Object Access Protocol (SOAP) layer message security providers installed with GlassFish Server can be used to employ username/password and X.509 certificate security tokens to authenticate and encrypt SOAP web services messages.

- **Username Tokens.** GlassFish Server uses username tokens in SOAP messages to authenticate the message sender. The recipient of a message containing a username token (within embedded password) validates that the message sender is authorized to act as the user (identified in the token) by confirming that the sender knows the password of the user. When using a username token, a valid user database must be configured on GlassFish Server.

- **Digital Signatures.** GlassFish Server uses XML digital signatures to bind an authentication identity to message content. Clients use digital signatures to establish their caller identity. Digital signatures are verified by the message receiver to authenticate the source of the message content (which might be different from the sender of the message.) When using digital signatures, valid keystore and truststore files must be configured on GlassFish Server.
**Encryption.** The purpose of encryption is to modify the data so that it can only be understood by its intended audience. This is accomplished by substituting an encrypted element for the original content. When based on public key cryptography, encryption can be used to establish the identity of the parties who are authorized to read a message. When using encryption, a Java Cryptography Extension (JCE) provider that supports encryption must be installed.

### Authentication Providers

The authentication layer is the message layer on which authentication processing must be performed. GlassFish Server enforces web services message security at the SOAP layer. The types of authentication that are supported include the following:

- **Sender authentication**, including username-password authentication
- **Content authentication**, including XML digital signatures

GlassFish Server invokes authentication providers to process SOAP message layer security. The message security providers provide information such as the type of authentication that is required for the request and response messages. The following message security providers are included with GlassFish Server:

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

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

- **Server-side Provider.** A server-side provider establishes its container as an authorized recipient of a received request (by successfully decrypting it), and validates passwords or signatures in the request to authenticate the source identity associated with the request. A server-side provider also establishes (by signature or username/password) the source identity of response messages and/or protects (by encryption) response messages such that they can only be viewed by their intended recipients. Server-side providers are only invoked by server-side containers.

  The *default server provider* is used to identify the server—side provider to be invoked for any application for which a specific server provider has not been bound.
Message Protection Policies

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

Message protection policies are defined for request message processing and response message processing. The policies are expressed in terms of requirements for source and/or recipient authentication. The providers apply specific message security mechanisms to cause the message protection policies to be realized in the context of SOAP web services messages.

- **Source Authentication Policy.** A source authentication policy represents a requirement that the identity of the entity that sent a message or that defined the content of a message be established in the message such that it can be authenticated by the message receiver.

- **Recipient Authentication Policy.** A recipient authentication policy represents a requirement that the message be sent such that the identity of the entities that can receive the message can be established by the message sender.

Request and response message protection policies are defined when a security provider is configured into a container. Application-specific message protection policies (at the granularity of the web service port or operation) can also be configured within the GlassFish Server deployment descriptors of the application or application client. In any situation where message protection policies are defined, the request and response message protection policies of the client must be equivalent to the request and response message protection policies of the server.

For more information about defining application-specific message protection policies, see Chapter 4, “Securing Applications,” in Oracle GlassFish Server 3.1 Application Development Guide.

Application-Specific Web Services Security

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

For information about defining application-specific message protection policies, see Chapter 4, “Securing Applications,” in Oracle GlassFish Server 3.1 Application Development Guide.
Message Security Administration

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

GlassFish Server administrative interfaces can be used as follows:

- To modify the message protection policies enforced by the providers
- To bind the existing providers for use by the server-side containers of GlassFish Server
- To create new security provider configurations with alternative message protection policies

Analogous administrative operations can be performed on the SOAP message layer security configuration of the application client container. If you want web services security to protect all web services applications deployed on GlassFish Server. See “Enabling Message Security for Application Clients” on page 83.

By default, message layer security is disabled on GlassFish Server. To configure message layer security for the GlassFish Server see “Enabling Default Message Security Providers for Web Services” on page 75.

In most cases, you must restart GlassFish Server after performing administrative tasks. This is especially true if you want the effects of the administrative change to be applied to applications that were already deployed on GlassFish Server at the time the operation was performed.

Message Security Tasks

The general implementation tasks for message security include some or all of the following:

1. If you are using a version of the Java SDK prior to version 1.5.0, and using encryption technology, configuring a JCE provider
2. If you are using a username token, verifying that a user database is configured for an appropriate realm
   When using a username/password token, an appropriate realm must be configured and a user database must be configured for the realm.
3. Managing certificates and private keys, if necessary
4. Enabling the GlassFish Server default providers
5. Configuring new message security providers
Message Security Roles
In GlassFish Server, the administrator and the application deployer are expected to take primary responsibility for configuring message security. In some situations, the application developer might also contribute.

System Administrator
The system administrator is responsible for the following message security tasks:
- Administering server security settings and certificate databases
- Administering keystore and truststore files
- Configuring message security providers on GlassFish Server
- Turning on message security
- (If needed) Installing the samples server

Application Deployer
The application deployer is responsible for the following message security tasks:
- Specifying (at application reassembly) any required application-specific message protection policies if such policies have not already been specified by the developer/assembler.
- Modifying GlassFish Server deployment descriptors to specify application-specific message protection policies information (message-security-binding elements) to web service endpoint and service references.

Application Developer/Assembler
The application developer/assembler is responsible for the following message security tasks:
- Determining if an application-specific message protection policy is required by the application
  - If so, the developer ensures that the required policy is specified at application assembly time.
- Specifying how web services should be set up for message security
  - Message security can be set up by the administrator so that all web services are secured, or by the application deployer when the security provider or protection policy bound to the application must be different from that bound to the container.
- Turning on message security if authorized to do so by the administrator

Sample Application for Web Services
GlassFish Server includes a sample application named xms. The xms application features a simple web service that is implemented by both a Java EE EJB endpoint and a Java servlet endpoint. Both endpoints share the same service endpoint interface. The service endpoint
interface defines a single operation, sayHello, which takes a string argument, and returns a String composed by pre-pending Hello to the invocation argument.

The xms sample application is provided to demonstrate the use of GlassFish Server WS-Security functionality to secure an existing web services application. The instructions which accompany the sample describe how to enable the WS-Security functionality of GlassFish Server such that it is used to secure the xms application. The sample also demonstrates the binding of WS-Security functionality directly to the application as described in “Application-Specific Web Services Security” on page 72 application.

For information about compiling, packaging, and running the xms sample application, Chapter 4, “Securing Applications,” in Oracle GlassFish Server 3.1 Application Development Guide.

The xms sample application is installed in the following directory:
as-install/samples/webservices/security/ejb/apps/xms/

Enabling Default Message Security Providers for Web Services

By default, message security is disabled on GlassFish Server. Default message security providers have been created, but are not active until you enable them. After the providers have been enabled, message security is enabled.

The following topics are addressed here:

■ “To Enable a Default Server Provider” on page 75
■ “To Enable a Default Client Provider” on page 76

To Enable a Default Server Provider

To enable message security for web services endpoints deployed in GlassFish Server, you must specify a security provider to be used by default on the server side. If you enable a default provider for message security, you also need to enable providers to be used by clients of the web services deployed in GlassFish Server.

1 Specify the default server provider by using the set(1) subcommand.
Use the following syntax:

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

2 To apply your changes to applications that are already running, restart GlassFish Server.
See “To Restart a Domain” in Oracle GlassFish Server 3.1 Administration Guide.
To Enable a Default Client Provider

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

1. Specify the default client provider by using the `set(1)` subcommand.
   Use the following syntax:
   ```bash
   asadmin set --port admin-port
   server-config.security-service.message-security-config.SOAP.
   default_client_provider=ClientProvider
   ```

2. To apply your changes to applications that are already running, restart GlassFish Server.
   See “To Restart a Domain” in Oracle GlassFish Server 3.1 Administration Guide.

Configuring Message Protection Policies

Message protection policies are defined for request message processing and response message processing. The policies are expressed in terms of requirements for source and/or recipient authentication. The providers apply specific message security mechanisms to cause the message protection policies to be realized in the context of SOAP web services messages.

The following topics are addressed here:

- “Message Protection Policy Mapping” on page 76
- “To Configure the Message Protection Policies for a Provider” on page 78
- “Setting the Request and Response Policy for the Application Client Configuration” on page 78

Message Protection Policy Mapping

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

<table>
<thead>
<tr>
<th>Message Protection Policy</th>
<th>Resulting WS-Security SOAP message protection operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth-source=&quot;sender&quot;</td>
<td>The message contains a <code>wsse:Security</code> header that contains a <code>wsse:UsernameToken</code> (with password).</td>
</tr>
<tr>
<td>Message Protection Policy</td>
<td>Resulting WS-Security SOAP message protection operations</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td><code>auth-source=&quot;content&quot;</code></td>
<td>The content of the SOAP message Body is signed. The message contains a <code>wsse:Security</code> header that contains the message Body signature represented as a <code>ds:Signature</code>.</td>
</tr>
<tr>
<td><code>auth-source=&quot;sender&quot;</code></td>
<td>The content of the SOAP message Body is encrypted and replaced with the resulting <code>xenc:EncryptedData</code>. The message contains a <code>wsse:Security</code> header that contains a <code>wsse:UsernameToken (with password)</code> and an <code>xenc:EncryptedKey</code>. The <code>xenc:EncryptedKey</code> contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.</td>
</tr>
<tr>
<td><code>auth-recipient=&quot;before-content&quot;</code></td>
<td>The content of the SOAP message Body is encrypted and replaced with the resulting <code>xenc:EncryptedData</code>. The <code>xenc:EncryptedData</code> is signed. The message contains a <code>wsse:Security</code> header that contains an <code>xenc:EncryptedKey</code> and a <code>ds:Signature</code>. The <code>xenc:EncryptedKey</code> contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.</td>
</tr>
<tr>
<td><code>auth-source=&quot;content&quot;</code></td>
<td>The content of the SOAP message Body is signed, then encrypted, and then replaced with the resulting <code>xenc:EncryptedData</code>. The message contains a <code>wsse:Security</code> header that contains an <code>xenc:EncryptedKey</code> and a <code>ds:Signature</code>. The <code>xenc:EncryptedKey</code> contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.</td>
</tr>
<tr>
<td><code>auth-recipient=&quot;after-content&quot;</code></td>
<td>The content of the SOAP message Body is encrypted and replaced with the resulting <code>xenc:EncryptedData</code>. The message contains a <code>wsse:Security</code> header that contains an <code>xenc:EncryptedKey</code>. The <code>xenc:EncryptedKey</code> contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.</td>
</tr>
<tr>
<td><code>auth-recipient=&quot;before-content&quot;</code></td>
<td>The content of the SOAP message Body is encrypted and replaced with the resulting <code>xenc:EncryptedData</code>. The message contains a <code>wsse:Security</code> header that contains an <code>xenc:EncryptedKey</code>. The <code>xenc:EncryptedKey</code> contains the key used to encrypt the SOAP message body. The key is encrypted in the public key of the recipient.</td>
</tr>
</tbody>
</table>

No policy specified. No security operations are performed by the modules.
To Configure the Message Protection Policies for a Provider

Typically, you would not reconfigure a provider. However, if needed for your situation, you can modify a provider’s message protection policies by changing provider type, implementation class, and provider-specific configuration properties. To understand the results of different combinations, see Table 3–1.

Use the `set(1)` subcommand to set the response policy, then replace the word `request` in the following commands with the word `response`.

1. Add a request policy to the client and set the authentication source by using the `set(1)` subcommand.
   For example:
   ```
asadmin> set server-config.security-service.message-security-config.SOAP.
   provider-config.ClientProvider.request-policy.auth_source=[sender | content]
   ```

2. Add a request policy to the server and set the authentication source by using the `set` subcommand.
   For example:
   ```
asadmin> set server-config.security-service.message-security-config.SOAP.
   provider-config.ServerProvider.request-policy.auth_source=[sender | content]
   ```

3. Add a request policy to the client and set the authentication recipient by using the `set` subcommand:
   For example:
   ```
asadmin> set server-config.security-service.message-security-config.SOAP.
   provider-config.ClientProvider.request-policy.auth_recipient=[before-content | after-content]
   ```

4. Add a request policy to the server and set the authentication recipient by using the `set` subcommand:
   For example:
   ```
asadmin> set server-config.security-service.message-security-config.SOAP.
   provider-config.ServerProvider.request-policy.auth_recipient=[before-content | after-content]
   ```

Setting the Request and Response Policy for the Application Client Configuration

The request and response policies define the authentication policy requirements associated with request and response processing performed by the authentication provider. Policies are expressed in message sender order such that a requirement that encryption occur after content would mean that the message receiver would expect to decrypt the message before validating the signature.
To achieve message security, the request and response policies must be enabled on both the server and client. When configuring the policies on the client and server, make sure that the client policy matches the server policy for request/response protection at application-level message binding.

To set the request policy for the application client configuration, modify the GlassFish Server–specific configuration for the application client container as described in "Enabling Message Security for Application Clients" on page 83.

**EXAMPLE 3-1  Message Security Policy Setting for Application Clients**

In the application client configuration file, the request-policy and response-policy elements are used to set the request policy, as shown in the following code snippet. (Additional code in the snippet is provided as illustration and might differ slightly in your installation. Do not change the additional code.)

```xml
<client-container>
  <target-server name="your-host" address="your-host"
    port="your-port"/>
  <log-service file="" level="WARNING"/>
  <message-security-config auth-layer="SOAP"
    default-client-provider="ClientProvider">
    <provider-config class-name="com.sun.enterprise.security.jauth.ClientAuthModule"
      provider-id="ClientProvider" provider-type="client">
      <request-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
      <response-policy auth-source="sender | content"
        auth-recipient="after-content | before-content"/>
      <property name="security.config"
        value="as-install/lib/appclient/wss-client-config.xml"/>
    </provider-config>
  </message-security-config>
</client-container>
```

Valid values for auth-source include sender and content. Valid values for auth-recipient include before-content and after-content. A table describing the results of various combinations of these values can be found in "Configuring Message Protection Policies" on page 76.

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

```xml
<response-policy/>
```
Administering Non-default Message Security Providers

The following topics are addressed here:

- “To Create a Message Security Provider” on page 80
- “To List Message Security Providers” on page 81
- “To Update a Message Security Provider” on page 81
- “To Delete a Message Security Provider” on page 81
- “To Configure a Servlet Layer Server Authentication Module (SAM)” on page 82

▼ To Create a Message Security Provider

Use the create-message-security-provider subcommand in remote mode to create a new message provider for the security service. If the message layer does not exist, the message layer is created, and the provider is created under it.

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

2. **Create the message security provider by using the create-message-security-provider(1) subcommand.**
   Information about properties for this subcommand is included in the help page.

3. **(Optional) If needed, restart the server.**
   Some properties require server restart. See “Configuration Changes That Require Restart” in Oracle GlassFish Server 3.1 Administration Guide. If your server needs to be restarted, see “To Restart a Domain” in Oracle GlassFish Server 3.1 Administration Guide.

**Example 3–2 Creating a Message Security Provider**

This example creates the new message security provider mySecurityProvider.

asadmin> create-message-security-provider
--classname com.sun.enterprise.security.jauth.ClientAuthModule
--providertype client mySecurityProvider
Command create-message-security-provider executed successfully.

**See Also** You can also view the full syntax and options of the subcommand by typing asadmin help create-message-security-provider at the command line.
**To List Message Security Providers**

Use the `list-message-security-providers` subcommand in remote mode to list the message providers for the security layer.

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

2. **List the message security providers by using the `list-message-security-providers(1)` subcommand.**

```bash
asadmin> list-message-security-providers --layer SOAP
XWS_ClientProvider
ClientProvider
XWS_ServerProvider
ServerProvider
Command list-message-security-providers executed successfully.
```

**Example 3–3** Listing Message Security Providers

This example lists the message security providers for a message layer.

You can also view the full syntax and options of the subcommand by typing `asadmin help list-message-security-providers` at the command line.

**To Update a Message Security Provider**

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

2. **List the message security providers by using the `list-message-security-providers(1)` subcommand.**

3. **Modify the values for the specified message security provider by using the `set(1)` subcommand.**
   The message security provider is identified by its dotted name.

**To Delete a Message Security Provider**

Use the `delete-message-security-provider` subcommand in remote mode to remove a message security provider.

1. **Ensure that the server is running.**
   Remote subcommands require a running server.
2 List the message security providers by using the `list-message-security-providers(1)` subcommand.

3 Delete the message security provider by using the `delete-message-security-provider(1)` subcommand.

**Example 3–4 Deleting a Message Security Provider**

This example deletes the `myServerityProvider` message security provider.

```
asadmin> delete-message-security-provider --layer SOAP myServerityProvider
Command delete-message-security-provider executed successfully.
```

**See Also** You can also view the full syntax and options of the subcommand by typing `asadmin help delete-message-security-provider` at the command line.

**To Configure a Servlet Layer Server Authentication Module (SAM)**

You configure a JSR 196 Server Authentication Module (SAM) as an HttpServlet-layer message security provider, either through the Administration Console or with the `create-message-security-provider` subcommand.

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

2 **Create the message security provider by using the `create-message-security-provider(1)` subcommand.**
   Information about properties for this subcommand is included in the help page.

3 **Bind the message security provider for use with your application.**
   You do this by defining the `httpservlet-security-provider` attribute in the `glassfish-web.xml` file corresponding to your application. Set the value of the attribute to the provider name you assigned to the message security provider. For example, if you use MySAM when you create the message security provider the entry would be `httpservlet-security-provider="MySAM"`.

4 **(Optional) If needed, restart the server.**
   Some properties require server restart. See “Configuration Changes That Require Restart” in *Oracle GlassFish Server 3.1 Administration Guide*. If your server needs to be restarted, see “To Restart a Domain” in *Oracle GlassFish Server 3.1 Administration Guide*. 
Creating a Message Security Provider

This example creates the new message security provider mySAM.

```
asadmin> create-message-security-provider --layer=HttpServlet --classname com.sun.glassfish.oamsam.OAMAuthenticatorSAM --providertype server --property oam.resource.hostid.variation="your-host-system.com" mySAM
Creation of message security provider named mySAM completed successfully
Command create-message-security-provider executed successfully.
```

The subcommand results in the following `domain.xml` entry:

```xml
<message-security-config auth-layer="HttpServlet">
  <provider-config provider-type="server" provider-id="mySAM"
  class-name="com.sun.glassfish.oamsam.OAMAuthenticatorSAM">
    <property name="oam.resource.hostid.variation" value="your-host-system.com"></property>
    <request-policy></request-policy>
    <response-policy></response-policy>
  </provider-config>
</message-security-config>
```

To list the HttpServlet message security providers, use the `list-message-security-providers` subcommand:

```
asadmin> list-message-security-providers --layer HttpServlet
list-message-security-providers successful
GFConsoleAuthModule
mySAM
Command list-message-security-providers executed successfully.
```

See Also
You can also view the full syntax and options of the subcommand by typing asadmin help create-message-security-provider at the command line.

Enabling Message Security for Application Clients

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

To enable message security for client applications, modify the GlassFish Server specific configuration for the application client container. The process is analogous to the process in “Configuring Message Protection Policies” on page 76.
Additional Information About Message Security

For additional information about message security, see the following documentation:

- Chapter 39, “Introduction to Security in the Java EE Platform,” in *The Java EE 6 Tutorial*
- Chapter 4, “Securing Applications,” in *Oracle GlassFish Server 3.1 Application Development Guide*
This chapter describes important information about administering security in a cluster.

The following topics are described:

- “Configuring Certificates in Cluster Mode” on page 85
- “Dynamic Reconfiguration” on page 86
- “Understanding Synchronization” on page 87

This chapter assumes that you are familiar with security features such as authentication, authorization, and certificates. If you are not, see Chapter 1, “Administering System Security.”

Instructions for accomplishing the tasks specific to GlassFish Server by using the Administration Console are contained in the Administration Console online help.

### Configuring Certificates in Cluster Mode

The sections “Certificates and SSL” on page 28 and “Administering JSSE Certificates” on page 47 describe the relevant concepts and use of certificates in GlassFish Server.

By default, GlassFish Server uses self-signed certificates. The self-signed certificates that GlassFish Server uses might not be trusted by clients by default because a certificate authority does not vouch for the authenticity of the certificate.

You can instead use your own certificates, as described in “Using Your Own Certificates” on page 96.
Dynamic Reconfiguration

Administrative commands that you execute on the domain administration server (DAS) must either be replicated on the affected server instances, or on all server instances that are part of the cluster. GlassFish Server replicates the commands by sending the same administration command request that was sent to the DAS to the server instances. As a result of replicating the commands on the DAS and the individual instances, the DAS and the instances make the same changes to their respective copies of the domain’s configuration.

Note – Oracle recommends that you enable secure admin as described in Chapter 5, “Managing Administrative Security,” so that GlassFish Server securely transfers these files on the network.

Dynamic reconfiguration refers to using the --target operand to CLI subcommands to make a change to a server instance (if the user-specified target is a server instance), or all server instances that are part of the cluster (if the user-specified target is a cluster). For example:

```
asadmin create-jdbc-resource some-options --target some-target.
```

The --target operand allows the following values:

- **server** – Performs the command on the default server instance. This is the default value.
- **configuration_name** – Performs the command in the specified configuration.
- **cluster_name** – Performs the command on all server instances in the specified cluster.
- **instance_name** – Performs the command on a specified server instance.

If a command fails for a cluster, the status shows all server instances where dynamic reconfiguration failed, and suggests corrective next steps.

The command status also shows when a restart is required for each server instance.

The --target operand is supported for the following security-related CLI subcommands:

- `create-jacc-provider`
- `delete-jacc-provider`
- `list-jacc-providers`
- `create-audit-module`
- `create-auth-realm`
- `create-file-user`
- `delete-audit-module`
- `delete-auth-realm`
- `delete-file-user`
- `update-file-user`
- `create-message-security-provider`
- `delete-message-security-provider`
- `list-audit-modules`
Enabling Dynamic Configuration

Dynamic configuration is enabled by default and no additional action is required.

Use the following command to enable dynamic configuration from the command line:

```bash
asadmin --user user --passwordfile password-file set cluster-name-config.dynamic-reconfiguration-enabled=true.
```

To enable dynamic configuration from the Administration Console, perform the following steps:

1. Expand the Configurations node.
2. Click the name of the cluster's configuration.
3. On the Configuration System Properties page, check the Dynamic Reconfiguration Enabled box.
4. Click Save

Understanding Synchronization

As described in "Resynchronizing GlassFish Server Instances and the DAS" in Oracle GlassFish Server 3.1-3.1.1 High Availability Administration Guide, configuration data for a GlassFish Server instance is stored in the repository of the DAS and in a cache on the host that is local to the instance. The configuration data in these locations must be synchronized. The cache is synchronized only when a user uses the administration tools to start or restart an instance.

See "Resynchronizing GlassFish Server Instances and the DAS" in Oracle GlassFish Server 3.1-3.1.1 High Availability Administration Guide for information about default synchronization for files and directories, for the steps required to resynchronize an instance and the DAS, and for additional synchronization topics.
Managing Administrative Security

This chapter describes how to manage administrative security by using the secure administration feature.

This chapter assumes that you are familiar with security features such as authentication, authorization, and certificates. If you are not, first see Chapter 1, “Administering System Security.”

Instructions for accomplishing the tasks specific to GlassFish Server by using the Administration Console are contained in the Administration Console online help.

- “Secure Administration Overview” on page 89
- “How Secure Admin Works: The Big Picture” on page 90
- “Considerations When Running GlassFish Server With Default Security” on page 101
- “Running Secure Admin” on page 101
- “Additional Considerations When Creating Local Instances” on page 104
- “Secure Admin Use Case” on page 105
- “Upgrading an SSL-Enabled Secure GlassFish Installation to Secure Admin” on page 105

Secure Administration Overview

The secure administration feature allows an administrator to secure all administrative communication between the domain administration server (DAS), any remote instances, and administration clients such as the asadmin utility, the administration console, and REST clients.

In addition, secure administration helps to prevent DAS-to-DAS and instance-to-instance traffic, and carefully restricts administration-client-to-instance traffic.

The secure administration feature, which is henceforth referred to as secure admin, provides a secure environment, in which you can be confident that rogue users or processes cannot intercept or corrupt administration traffic or impersonate legitimate GlassFish Server components.
When you install GlassFish Server or create a new domain, secure admin is disabled by default. When secure admin is disabled, GlassFish Server does not encrypt administrative communication among the system components and does not accept administrative connections from remote hosts.

The following subcommands enable and disable secure admin:

- **enable-secure-admin**—The `enable-secure-admin` subcommand turns on secure admin. GlassFish Server uses SSL encryption to protect subsequent administrative traffic and will accept remote administrative connections. Enabling secure admin affects the entire domain, including the DAS and all instances. The DAS must be running, and not any instances, when you run `enable-secure-admin`. You must restart the DAS immediately after enabling secure admin, and then start any instances you want to run.

- **disable-secure-admin**—The `disable-secure-admin` subcommand turns off secure admin. GlassFish Server no longer encrypts administrative messages and will no longer accept remote administration connections. Disabling secure admin affects the entire domain, including the DAS and all instances. The DAS must be running, and not any instances, when you run `disable-secure-admin`. You must restart the DAS immediately after disabling secure admin, and then start any instances you want to run.

If secure admin is not enabled, this subcommand has no effect.

This section describes how to use these commands to run secure admin, and the implications of doing so.

---

**How Secure Admin Works: The Big Picture**

Secure admin is a domain-wide setting. It affects the DAS and all instances and all administration clients. This section describes the following topics:

- "Functions Performed by Secure Admin" on page 90
- "Which Administration Account is Used?" on page 91
- "What Authentication Methods Are Used for Secure Administration?" on page 92
- "Understanding How Certificate Authentication is Performed" on page 93
- "What Certificates Are Used?" on page 93
- "An Alternate Approach: Using Distinguished Names to Specify Certificates" on page 97
- "Guarding Against Unwanted Connections" on page 100

---

**Functions Performed by Secure Admin**

The `enable-secure-admin` subcommand performs the following functions. Subsequent sections describe these functions in more detail.

- Enables the secure admin behavior, optionally setting which aliases are to be used for identifying the DAS and instance certificates.
Adjusts all configurations in the domain, including default-config.

Adjusts Grizzly settings:
- SSL/TLS is enabled in the DAS's admin listener and the instances' admin listeners.
- Port unification (that is, HTTP and HTTPS are handled by the same port), http—to—https redirection, and client authentication (client-auth=want) are enabled.
- Configures SSL to use the administration truststore.
- Configures SSL to use the administration keystore and the correct alias (for the self-signed cert) for authenticating itself. (You can use your own certificate instead, as described in “Using Your Own Certificates” on page 96.

The Grizzly configuration on the DAS and each instance is identical, with the exception that the DAS uses the s1as alias for SSL/TLS authentication and the instances use the glassfish-instance alias. (These alias names are the default, and you can change them.)

A server restart is required to change the Grizzly adapter behavior.

The restart also synchronizes the restarted instances. When you start the instances, the DAS delivers the updated configuration to the instances.

Which Administration Account is Used?

If only one administration account exists in the realm, GlassFish Server treats that account as the current default administration account. In this case, when you run an asadmin command, you do not need to specify the username. If a password for that username is required, you need to specify it, typically by using the --passwordfile option or by letting asadmin prompt you for it.

By default, GlassFish Server includes a single account for user "admin" and an empty password. Therefore, if you make no other changes before you enable secure admin, "admin" is the initial default username and no password is required. You need to decide whether enabling secure admin without also requiring a password makes sense in your environment.

If multiple admin accounts exist, then GlassFish Server does not recognize any admin username as the default. You must then specify a valid username via the --user option when you use the asadmin command (or by or defining the AS_ASDMIN_USER environment variable), and its associated password (if the associated password is not empty).

The username and password used for a login attempt must match the username and password (if required) for an account defined in the realm, and you must have set up the account as a member of the admin group.
What Authentication Methods Are Used for Secure Administration?

The secure admin feature enforces security via the following authentication methods:

- The DAS and instances authenticate to each other via mutual (two-way) SSL/TLS certificate authentication. The DAS authenticates to clients via one-way SSL/TLS certificate authentication.

  The domain creation process creates a default keystore and truststore, plus a default private key for the DAS. Secure admin uses this initial configuration to set up the truststore so that the DAS and instances always trust each other.

- Remote administration clients (asadmin, administration console, browsers, and IDEs) must accept the public certificate presented by the DAS. If accepted, remote administration clients then send a user name and password (HTTP Basic authentication) in the HTTP Authorization header. The receiving DAS or instance makes sure those credentials are valid in its realm, and authenticates and authorizes the user.

- A locally-running asadmin (that is, connecting to an instance on the same host) authenticates and authorizes to the co-located instance using a locally-provisioned password.

- Credentials or other sensitive information sent over the network are always encrypted if secure admin is enabled. No credentials are sent in the clear if secure admin is enabled. (If secure admin is disabled, credentials are sent in the clear.) Messages between administration clients and the DAS, between the DAS and remote instances, and between local administration clients and instances are encrypted using SSL/TLS. This is true even if you explicitly set the asadmin --secure option to false.

Table 5–1 shows which authentication methods are employed when secure admin is enabled or disabled.

<table>
<thead>
<tr>
<th>Access Method</th>
<th>When Secure Admin is Disabled</th>
<th>When Secure Admin is Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote administration access to the DAS</td>
<td>Rejected.</td>
<td>Username/password authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Client must also accept server certificate.)</td>
</tr>
<tr>
<td>Communication between DAS and instances</td>
<td>Cleartext messages. No mutual authentication.</td>
<td>SSL-encrypted messages. SSL mutual authentication using certificates.</td>
</tr>
<tr>
<td>Communication between administration clients and DAS</td>
<td>Cleartext messages. No DAS authentication.</td>
<td>SSL-encrypted messages. DAS uses SSL certificate server authentication.</td>
</tr>
<tr>
<td>Local asadmin client to instance on same node</td>
<td>Cleartext messages. Locally-provisioned password mechanism is used.</td>
<td>SSL-encrypted messages. Locally-provisioned password mechanism is used.</td>
</tr>
</tbody>
</table>
Understanding How Certificate Authentication is Performed

The domain creation process creates a primary (private) key and a self-signed certificate for the DAS, and a separate private key and self-signed certificate for remote instances.

Then, when you enable secure admin, the following actions are performed:

- Both private keys are stored in the domain-wide DAS keystore file, keystore.jks.
- Both public certificates are stored in the domain-wide DAS truststore file, cacerts.jks.

When the DAS sends a message to an instance:

1. SSL on the instance asks the DAS to provide an SSL/TLS certificate.
2. The DAS sends the certificate with the alias you specified using the --adminalias option when you ran the enable-secure-admin subcommand.
3. SSL on the instance makes sure the certificate is valid and GlassFish Server makes sure that the security Principal associated with the incoming request (provided automatically by Grizzly and the SSL/TLS Java implementation) matches the Principal associated with the adminalias from the instance’s truststore.

What Certificates Are Used?

When you enable secure admin, you can optionally set the --adminalias and --instancealias options that tell secure admin which aliases to use for the DAS and instance certificates.

The DAS uses the alias associated with the --instancealias option to check incoming requests that use SSL/TLS cert authentication. Conversely, instances use the alias associated with the --adminalias option to check incoming requests with certificate authentication.

By default, --adminalias of the enable-secure-admin subcommand uses the s1as alias, and the --instancealias option uses the glassfish-instance alias, both of which identify the default self-signed certificates.

You can use your tool of choice, such as keytool, to list the default self-signed certificates in the keystore, similar to the following:

```
keytool.exe -list -keystore keystore.jks
```

**Note** – You can list the contents of the keystore without supplying a password. However, for a request that affects the private key, such as the keytool.exe --certreq option, the keystore password is required. This is the master password and has a default value of changeit unless you change it with the change-master-password subcommand.
Enter keystore password:

************************ WARNING WARNING WARNING ****************************
* The integrity of the information stored in your keystore *
* has NOT been verified! In order to verify its integrity, *
* you must provide your keystore password. *
************************ WARNING WARNING WARNING ****************************

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 2 entries

glassfish-instance, Jan 3, 2011, PrivateKeyEntry,
s1as, Jan 3, 2011, PrivateKeyEntry,

The --adminalias and --instancealias values are maintained. Because of this design, normal instance creation operations (create-instance over SSH and create-local-instance) apply the up-to-date keystore, truststore, and configuration to each instance.

**Self-Signed Certificates and Trust**

The self-signed certificates that GlassFish Server uses might not be trusted by clients by default because a certificate authority does not vouch for the authenticity of the certificate. If you enable secure admin and then contact the DAS using an administration client, that client will detect whether the certificate is automatically trusted.

Browsers will warn you, let you view the certificate, and ask you to reject the certificate, accept it once, or accept it indefinitely, as shown in Figure 5–1.
Similarly, the first time asadmin receives an untrusted certificate, it displays the certificate and lets you accept it or reject it, as follows: (If you accept it, asadmin also accepts that certificate in the future.)

D:\glassfish3\glassfish\bin>asadmin enable-secure-admin
Command enable-secure-admin executed successfully.

D:\glassfish3\glassfish\bin>asadmin stop-domain domain1
Waiting for the domain to stop ........
Command stop-domain executed successfully.

D:\glassfish3\glassfish\bin>asadmin start-domain domain1
Waiting for domain1 to start .........................
Successfully started the domain : domain1
Domain Location: D:\glassfish3\glassfish\domains\domain1
Log File: D:\glassfish3\glassfish\domains\domain1\logs\server.log
Admin Port: 4848
Command start-domain executed successfully.

D:\glassfish3\glassfish\bin>asadmin list-domains
[
][
Version: V3
Subject: CN=machine.oracle.com, OU=GlassFish, O=Oracle Corporation, L=San Clara, ST=California, C=US
Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5
Key: Sun RSA public key, 1024 bits
  modulus: 916043595073784449632358756374297330881618062298549101872702252458856
  740796563583225688000354850721926291808643119248249383195045822088563459253216383
  21100660819657284757523896415668324714995640712267224780056407183213862976797465
  6245850519956376352782895837519504394674686082145398885236913866246525691704749
  public exponent: 65537

Chapter 5 • Managing Administrative Security 95
Validity: [From: Tue Jan 04 14:30:08 EST 2011, To: Fri Jan 01 14:30:08 EST 2021]

Issuer: CN=machine.oracle.com, OU=GlassFish, O=Oracle Corporation, L=Santa Clara, ST=California, C=US

SerialNumber: [4d237540]

Certificate Extensions: 1

[1]: ObjectId: 2.5.29.14 Criticality=false

SubjectKeyIdentifier [KeyIdentifier [0000: AF 8B 90 1E 51 9A 80 1B EB A4 D9 C6 01 8A A0 FD ....Q...........
0010: DE EC 83 8A ....]]

Algorithm: [SHA1withRSA]

Signature:

0000: 3F 2B 30 CE 97 0B 5E F3 72 0E 60 18 8D 3B 04 DC ?+0...^,.r'.;,
0010: 26 E6 7A 6F D0 19 CC 26 1D 90 C0 DE 33 4E 53 FB &..zo...&....NS.
0020: 0E 7E 8E 57 36 D4 3E 9B FB ...x......,W6>....
0030: 0F 38 EF 72 27 D9 4F 79 1F 89 91 BF 96 26 33 64 ...r'.Oy......G3d
0040: 8F 04 9B 04 48 83 B9 BF 4D 54 B4 8F 75 17 1A 51 BD .K.K...MT...u...Q.
0050: F3 69 94 CE 90 95 08 55 2C 07 D2 23 AC AE EC 6D .,...U,...#...m
0060: 84 B6 3D 00 FB FE 92 50 37 1A 20 00 F1 21 5C E6 ..=....P7.-!/.m
0070: 1F 39 26 B2 50 C1 FD CB B1 4F CC EE 26 84 B8 B5 .96...O...&...

Do you trust the above certificate [y/N] -->

asadmin saves certificates you accept in the file .asadmintruststore in your log-in default directory. You do not generally need to work with the file directly, but if you delete or move the file, asadmin will prompt you again when it receives untrusted certificates.

Some asadmin commands such as run-script can contact an instance directly to retrieve information (but not to make configuration changes). The instances do not use the same certificate as the DAS, so in these cases asadmin then prompts you to accept or reject the instance certificate.

### Using Your Own Certificates

By default, --adminalias of the enable-secure-admin subcommand uses the s1as alias, and the --instancealias option uses the glassfish-instance alias, both of which identify the default self-signed certificates.

You can instead have GlassFish Server use your own certificates for this purpose by first adding your certificates to the keystore and truststore, and then running enable-secure-admin and specifying the aliases for your certificates.

It is also possible to use s1as and glassfish-instance as the alias names for your own certificates. A benefit of doing so is that you would not have to specify alias names with the enable-secure-admin subcommand.
In addition, your own certificate identified by the s1as alias would be used in all other cases within the domain where the s1as alias is used (by default), such as in the SSL configuration of the IIOP and http-listener-2 listeners, and as the encryption.key.alias and signature.key.alias used for provider configuration in the SOAP authentication layer for Message Security configuration.

You may find the wide-reaching effect of using the s1as alias with your own certificate to be either a useful feature or an unintended consequence. Therefore, you should understand the implications of using the s1as alias before doing so.

If you decide to use the s1as and glassfish-instance aliases with your own certificates, you will first need to disable secure admin (if enabled) and then change or delete the exiting s1as alias from both the keystore.jks keystore and cacerts.jks truststore for the DAS. You can use the --changealias or --delete option of keytool to accomplish this. Then, import your own certificates.

When you enable secure admin, the DAS and the instances then have copies of the same keystore and truststore.

**An Alternate Approach: Using Distinguished Names to Specify Certificates**

By default, the DAS uses the alias associated with the --instance.alias option to check incoming requests that use SSL/TLS cert authentication. Conversely, instances use the alias associated with the --admin.alias option to check incoming requests with certificate authentication.

The `enable-secure-admin-principal(1)` subcommand provides an alternate approach. `enable-secure-admin-principal` instructs GlassFish Server to accept admin requests when accompanied by an SSL certificate with the specified distinguished name (DN).

**Note** – Any certificate you specify with `enable-secure-admin-principal` must either be issued by a trusted certificate authority or, if it is self-signed, must already be in the GlassFish Server truststore.

For example, assume that you write your own admin client that uses the REST interface. When your client establishes the connection, it can choose which certificate to use for its client cert. You would then specify the DN of this certificate to `enable-secure-admin-principal`.

You must specify either the DN or the --alias option of the `enable-secure-admin-principal` subcommand.
If you specify the DN, GlassFish Server records the value you specify as the DN. You specify the DN as a comma-separated list in quotes. For example, "CN=system.amer.oracle.com,OU=GlassFish,O=Oracle Corporation,L=Santa Clara,ST=California,C=US".

**Note**—The `enable-secure-admin-principal` subcommand accepts the string you enter and does not immediately validate it. However, secure admin must be able to match the DN you specify in order to use it.

If you have sufficient privileges to view the content of the keystore, you can use keytool to display the DN of a certificate:

```
keytool.exe -v -list -keystore keystore.jks
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 2 entries

Alias name: glassfish-instance
Creation date: Jul 7, 2011
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
    Owner: CN=systemname.amer.oracle.com-instance, OU=GlassFish, O=Oracle Corporation, L=Santa Clara, ST=California, C=US
    Issuer: CN=systemname.amer.oracle.com-instance, OU=GlassFish, O=Oracle Corporation, L=Santa Clara, ST=California, C=US
    Serial number: 4e15d6e7
    Certificate fingerprints:
        Signature algorithm name: SHA1withRSA
        Version: 3

Extensions:
    #1: ObjectId: 2.5.29.14 Criticality=false
        SubjectKeyIdentifier [KeyIdentifier]
            0000: 96 99 36 86 CF 60 1E 8A AE 25 75 4E C8 34 AA AB ..6..'...
        0010: 9D 03 03 CF 03 ...%

If you use the "- -alias aliasname" form, then GlassFish Server looks in its truststore for a certificate with the specified alias and uses the DN associated with that certificate. `alias-name` must be an alias associated with a certificate currently in the truststore. Therefore, you may find it most useful for self-signed certificates for which you know the alias.
If you have sufficient privileges to view the contents of the truststore, you can use keytool to display the alias of a certificate:

```
keytool.exe -v -list -keystore cacerts.jks
Enter keystore password: : 
Alias name: glassfish-instance 
Creation date: Jul 7, 2011 
Entry type: trustedCertEntry 
Owner: CN=systemname.amer.oracle.com-instance, OU=GlassFish, O=Oracle Corporation, L=Santa Clara, ST=California, C=US 
Issuer: CN=systemname.amer.oracle.com-instance, OU=GlassFish, O=Oracle Corporation, L=Santa Clara, ST=California, C=US 
Serial number: 4e15d6e7 
Certificate fingerprints: 
Signature algorithm name: SHA1withRSA 
Version: 3 
```

Extensions:

```
#1: ObjectId: 2.5.29.14 Criticality=false 
SubjectKeyIdentifier [ 
  KeyIdentifier [ 
    0000: 96 99 36 B6 CF 60 1E 8A AE 25 75 4E C8 34 AA AB ..6..%uN.4.. 
    0010: E1 3B CF 03 .;.. ] ] ] 
When you run enable-secure-admin, GlassFish Server automatically records the DNs for the admin alias and the instance alias, whether you specify those values or use the defaults. You do not need to run enable-secure-admin-principal yourself for those certificates.

Other than these certificates, you must run enable-secure-admin-principal for any other DN that GlassFish Server should authorize to send admin requests. This includes DNs corresponding to trusted certificates (those with a certificate chain to a trusted authority.)

You can run enable-secure-admin-principal multiple times so that GlassFish Server accepts admin requests from a client sending a certificate with any of the DNs you specify.

The following example shows how to specify a DN for authorizing access in secure administration:

```
asadmin> enable-secure-admin-principal "CN=system.amer.oracle.com,OU=GlassFish, O=Oracle Corporation,L=Santa Clara,ST=California,C=US"
```

Command enable-secure-admin-principal executed successfully.

You can use the disable-secure-admin-principal(1) subcommand to disable a specific certificate for authenticating and authorizing access in secure admin. You must specify either
the DN or the --alias option of the disable-secure-admin-principal subcommand. To disable multiple certificates for authenticating and authorizing access in secure admin, run the disable-secure-admin-principal subcommand multiple times.

You can use the list-secure-admin-principals(1) subcommand to list the certificates for which GlassFish Server accepts admin requests from clients.

Guarding Against Unwanted Connections

Secure admin guards against unwanted connections in several ways:

- **DAS-to-DAS, instance-to-instance:**
  - The DAS and the instances have copies of the same truststore, which contains the public certificate of the DAS and the separate public certificate that is used by all instances. In addition, GlassFish Server includes a unique, generated "domain ID" that servers use to ensure that admin requests from other GlassFish Servers originate from the correct domain.
  - DAS-to-other-DAS communication is not authenticated because each different DAS will have its own self-signed certificate that is not in the truststore of the other DAS.
  - DAS-to-itslf communication is unlikely unless you were to misconfigure the admin listener port for an instance on the same host so it is the same as for the DAS. Similarly, instance-to-instance traffic is unlikely unless you were to misconfigure listener ports for instances on the same host.

To prevent both of these situations, both cases are handled by making sure that the connecting Principal (alias) is not the running Principal. secure admin ensures that if the client has authenticated using SSL/TLS client authentication that the Principal associated with the remote client is not the same as the current process. That is, the DAS makes sure that the Principal is not itself. Similarly, each instance ensures that the client is not an instance. (The instances share the same self-signed certificate and therefore are mapped to the same Principal.)

- **Remote client-to-instance:**
  Remote asadmin clients are unable to connect directly to instances. If the user on host "test1" runs a local command but specifies a remote instance on host "test2," asadmin on test1 will read and send that locally-provisioned password. The instance on "test2" will have a different locally-provisioned password and so the authentication attempt will fail.

Therefore, a user on "test1" will not be able to run a remote command targeting an instance on "test2."
Considerations When Running GlassFish Server With Default Security

In GlassFish Server, the default admin account is username “admin” with an empty password. Admin clients provide empty credentials or none at all, and all are authenticated and authorized as that default admin user. None of the participants (clients, DAS, or instances) encrypts network messages.

If this level of security is acceptable in your environment, no changes are needed and you do not need to enable secure administration. Imposing a heightened level of security is optional.

However, consider Table 5-2, which shows which operations are accepted and rejected when secure admin is disabled.

Note – When secure admin is disabled, GlassFish Server does allow remote monitoring (read-only) access via the REST interface.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Run From Same System as DAS</th>
<th>Run From Remote System</th>
</tr>
</thead>
<tbody>
<tr>
<td>start-local-instance</td>
<td>Functions as expected</td>
<td>Cannot sync with DAS. The instance starts but cannot communicate with the DAS. DAS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>will not see the instance.</td>
</tr>
<tr>
<td>Any other asadmin subcommand</td>
<td>Functions as expected</td>
<td>Rejected. A user sees the username/password prompt, but even correct entries are rejected.</td>
</tr>
<tr>
<td>Commands that use SSH.</td>
<td>Functions as expected; requires</td>
<td>Functions as expected; requires prior SSH configuration.</td>
</tr>
<tr>
<td></td>
<td>prior SSH configuration.</td>
<td></td>
</tr>
</tbody>
</table>

Running Secure Admin

This section describes how to run secure admin. The section begins with prerequisites for running secure admin.

Prerequisites for Running Secure Admin

Before running GlassFish Server with secure admin enabled, you must make sure that:

1. The DAS is installed, initialized, and running.
2. If one or more remote instances are installed and initialized, they must not be running.
3. Any administration clients you require are installed.

4. The DAS communicates on the `--admin-port` you configure when you create the domain, and defaults to 4848. An instance communicates on the `ASADMIN_LISTENER_PORT` system property you specify for the instance.

5. The user name and password sent by remote administration clients (`asadmin`, administration console, browsers, and IDEs) must exist in the realm and be in the admin group.

6. The keystore and truststore for the domain exist. (They are created by default when you create the domain or install GlassFish Server.)

   If you are not using the default self-signed certificates, you must add your own valid certificates and CA root in the keystore and truststore, respectively.

7. If you are not using the default self-signed certificates, create two aliases corresponding to certificates in the keystore and truststore: one that the DAS will use for authenticating itself in administration traffic, and one that the instances will use for authenticating itself in administration traffic.

---

**An Alternate Approach: Using A User Name and Password for Internal Authentication and Authorization**

By default, secure admin uses the GlassFish Server self-signed certificates, via the aliases corresponding to these certificates, to authenticate the DAS and instances with each other and to authorize secure admin operations. Specifically, the DAS uses the (`s1as`) alias for authenticating itself and authorizing access in administration traffic, and instances use the (`glassfish-instance`) alias for authenticating themselves and authorizing access in secure admin traffic.

As described in “Using Your Own Certificates” on page 96, you can instead use your own certificates and their associated aliases for authenticating and authorizing the DAS and instances in administration traffic.

As an alternative to this certificate-based authentication and authorization, you can instead use the `enable-secure-admin-internal-user(1)` subcommand to instruct all servers in the domain to authenticate to each other, and to authorize admin operations submitted to each other, using an existing admin user name and password rather than SSL certificates.

---

**Note** – If secure admin is enabled, all GlassFish Server processes continue to use SSL encryption to secure the content of the admin messages, regardless of how they authenticate to each other.
You might want to use the `enable-secure-admin-internal-user(1)` subcommand if your use case favors the use of a user name and password combination over the use of SSL certificates and aliases.

This generally means that you must:

1. Create a valid admin user.
   ```
   asadmin> create-file-user --authrealmname admin-realm --groups admin newAdminUsername
   ```

2. Create a password alias for the just-created password.
   ```
   asadmin> create-password-alias passwordAliasName
   ```

3. Use that username and password for inter-process authentication and admin authorization.
   ```
   asadmin> enable-secure-admin-internal-user --passwordalias passwordAliasName newAdminUsername
   ```

The following example allows secure admin to use a user name and password alias for authentication and authorization between the DAS and instances, instead of certificates.

```
asadmin> enable-secure-admin-internal-user --passwordalias passwordAliasName newAdminUsername
``` 

If GlassFish Server finds at least one secure admin internal user, then if secure admin is enabled GlassFish Server processes will not use SSL authentication and authorization with each other and will instead use user name password pairs.

Most users who use this subcommand will need to set up only one secure admin internal user. If you set up more than one secure admin internal user, you should not make any assumptions about which user name and password pair GlassFish Server will choose to use for any given admin request.

As a general practice, you should not use the same user name and password pair for internal admin communication and for admin user login. That is, create at least one admin account specifically for internal admin communication.

You can use the `disable-secure-admin-internal-user(1)` subcommand to disable secure admin from using the user name (instead of SSL certificates) to authenticate the DAS and instances with each other and to authorize admin operations. To disable multiple user names for authenticating and authorizing access in secure admin, run the `disable-secure-admin-internal-user` subcommand multiple times.

You can use the `list-secure-admin-internal-users(1)` subcommand to list the user names for which GlassFish Server authenticate the DAS and instances with each other and authorizes admin operations.
Example of Running enable-secure-admin

The following example shows how to enable secure admin for a domain using the default admin alias and the default instance alias. You must restart the DAS immediately after enabling secure admin.

```
asadmin> enable-secure-admin
Command enable-secure-admin executed successfully.
```

**Note** – The only indicator that secure admin is enabled is the successful status from the `enable-secure-admin` subcommand. When secure admin is running, the DAS and instances do not report the secure admin status.

```
asadmin> enable-secure-admin --adminalias adtest --instancealias intest
```

The following example shows how to disable secure admin:

```
asadmin> disable-secure-admin
Command disable-secure-admin executed successfully.
```

You can use the following command to see the current state of secure admin in a domain:

```
asadmin> get secure-admin.enabled
secure-admin.enabled=false
Command get executed successfully.
```

Additional Considerations When Creating Local Instances

If you use `xxx-local-instance` commands to set up local instances, either leave secure admin disabled, or enable it before you create or start the instances and leave it that way.

However, if you use `xxx-instance` commands over SSH to manage remote instances, you can enable and disable secure admin, although this is not recommended because it can result in an inconsistent security model.
Secure Admin Use Case

This section describes a simple secure admin use case.

In the `asadmin --secure=false --user me --passwordfile myFile.txt cmd ...` use case, the user submits a command with `--secure` set to false, and supplies password credentials.

The important concept to note is that `asadmin` uses HTTPS because of the DAS redirection, even though the command sets `--secure` to false. `asadmin` sends the HTTP Authorization header along with the redirected request.

In addition to the flow described here, certificate authentication is also performed as described in **Table 5–3**. Also, the credentials that the user supplies are assumed to be valid administrator credentials for the DAS.

<table>
<thead>
<tr>
<th>TABLE 5–3</th>
<th>asadmin --secure=false, With Username and Password</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>asadmin</td>
</tr>
<tr>
<td>asadmin</td>
<td>Sends HTTP request, no</td>
</tr>
<tr>
<td></td>
<td>authorization header (because the transport is not</td>
</tr>
<tr>
<td></td>
<td>secure).</td>
</tr>
<tr>
<td></td>
<td>Returns 3xx status and redirects HTTP to HTTPS.</td>
</tr>
<tr>
<td></td>
<td>Follows redirection, this time</td>
</tr>
<tr>
<td></td>
<td>adding the Authorization header</td>
</tr>
<tr>
<td></td>
<td>(because transport is now HTTPS).</td>
</tr>
<tr>
<td></td>
<td>Authenticates admin user and</td>
</tr>
<tr>
<td></td>
<td>password from HTTP Authorization header in the</td>
</tr>
<tr>
<td></td>
<td>realm.</td>
</tr>
<tr>
<td></td>
<td>Executes command, and responds with success status.</td>
</tr>
</tbody>
</table>

Upgrading an SSL-Enabled Secure GlassFish Installation to Secure Admin

If you enable secure admin on an SSL-enabled GlassFish Server installation, secure admin uses the existing `<ssl cert-nickname>` value as the DAS adminalias for secure admin.
This chapter describes important information about running GlassFish Server in a secure environment.

This chapter assumes that you are familiar with security features such as authentication, authorization, and certificates. If you are not, see Chapter 1, “Administering System Security.”

Instructions for accomplishing the tasks specific to GlassFish Server by using the Administration Console are contained in the Administration Console online help.

The chapter describes the following topics:

- “Determining Your Security Needs” on page 107
- “Installing GlassFish Server in a Secure Environment” on page 109
- “Remove Unused Components” on page 109
- “Run on the Web Profile if Possible” on page 112
- “Securing the GlassFish Server Host” on page 113
- “Securing GlassFish Server” on page 116
- “Securing Applications” on page 119

### Determining Your Security Needs

Before you deploy GlassFish Server and your Java EE applications into a production environment, determine your security needs and make sure that you take the appropriate security measures, as described in the following sections:

- “Understand Your Environment” on page 108
- “Hire Security Consultants or Use Diagnostic Software” on page 108
- “Read Security Publications” on page 108
Understand Your Environment

To better understand your security needs, ask yourself the following questions:

- Which resources am I protecting?
  
  Many resources in the production environment can be protected, including information in databases accessed by GlassFish Server and the availability, performance, applications, and the integrity of the Web site. Consider the resources you want to protect when deciding the level of security you must provide.

- From whom am I protecting the resources?
  
  For most Web sites, resources must be protected from everyone on the Internet. But should the Web site be protected from the employees on the intranet in your enterprise? Should your employees have access to all resources within the GlassFish Server environment? Should the system administrators have access to all GlassFish Server resources? Should the system administrators be able to access all data? You might consider giving access to highly confidential data or strategic resources to only a few well-trusted system administrators. Perhaps it would be best to allow no system administrators access to the data or resources.

- What will happen if the protections on strategic resources fail?
  
  In some cases, a fault in your security scheme is easily detected and considered nothing more than an inconvenience. In other cases, a fault might cause great damage to companies or individual clients that use the Web site. Understanding the security ramifications of each resource will help you protect it properly.

Hire Security Consultants or Use Diagnostic Software

Whether you deploy GlassFish Server on the Internet or on an intranet, it is a good idea to hire an independent security expert to go over your security plan and procedures, audit your installed systems, and recommend improvements. Oracle On Demand offers services and products that can help you to secure a GlassFish Server production environment. See the Oracle On Demand page at Oracle on Demand (http://www.oracle.com/us/products/ondemand/index.html).

Read Security Publications

Read about security issues:

- For the latest information about securing Web servers, Oracle recommends the “Security Practices & Evaluations” information available from the CERT Coordination Center operated by Carnegie Mellon University at http://www.cert.org (http://www.cert.org/).
Register your Oracle GlassFish Server installation with My Oracle Support. By registering, Oracle Support will notify you immediately of any security updates that are specific to your installation. You can create a My Oracle Support account by visiting http://www.oracle.com/support/index.html.

For security advisories, refer to the Critical Patch Updates and Security Alerts page at the following location:

Critical Patch Updates and Security Alerts (http://www.oracle.com/technetwork/topics/security/alerts-086861.html)

---

**Installing GlassFish Server in a Secure Environment**

This section describes recommendations for installing GlassFish Server in a secure environment. The following topic is described:

- “Enable the Secure Administration Feature” on page 109

**Enable the Secure Administration Feature**

The secure administration feature allows an administrator to secure all administrative communication between the domain administration server (DAS), any remote instances, and administration clients such as the asadmin utility, the administration console, and REST clients. In addition, secure administration helps to prevent DAS-to-DAS and instance-to-instance traffic, and carefully restricts administration-client-to-instance traffic.

When you install GlassFish Server or create a new domain, secure admin is disabled by default. GlassFish Server does not encrypt administrative communication among the system components and does not accept administrative connections from remote hosts. Imposing a heightened level of security is optional.

See Chapter 5, “Managing Administrative Security,” for information on enabling the secure administration feature.

**Remove Unused Components**

Minimize the GlassFish Server installation by removing components that you are not using and do not intend to use.

The Update Tool is a standalone graphical tool bundled with GlassFish Server that you can use to find, install, and remove updates and add-ons on a deployed server instance.
The `pkg` command is the command-line equivalent to Update Tool. Most of the tasks that can be performed with the graphical Update Tool can be performed from a command line using the `pkg` tool.

To update or remove installed add-on components, use one of the following commands:

- `install-dir/bin/updatetool`, which starts the Update Tool graphical utility.
- `install-dir/bin/pkg`, a command-line version of the Update Tool.

### Removing Installed Components

This section describes how to use the `pkg` utility to remove an installed component. You can also use the Update Tool to perform this task.

▼ **Procedure To Remove an Installed Component**

1. **Stop GlassFish Server.**
   - See “To Stop a Domain” in Oracle GlassFish Server 3.1 Administration Guide.

2. To ensure that the `pkg` command can locate the application image, change to the base installation directory for GlassFish Server.
   ```
   cd install-dir
   ```

3. Obtain a list of all your installed components. (The following list is for example purposes only and might not match your installed components.)
   ```
   install-dir/bin/pkg list
   ```

<table>
<thead>
<tr>
<th>NAME</th>
<th>VERSION</th>
<th>STATE</th>
<th>UFIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>felix</td>
<td>3.0.7-0</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-appclient</td>
<td>3.1-39</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-bundled-jdk (release.release.sun.com)</td>
<td>1.6.0.23-5.1</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-cluster</td>
<td>3.1-39</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-cmp</td>
<td>3.1-39</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-common</td>
<td>3.1-39</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-common-full</td>
<td>3.1-39</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-corba</td>
<td>3.1.0-23</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-corba-base</td>
<td>3.1.0-23</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-ejb</td>
<td>3.1-39</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-ejb-lite</td>
<td>3.1-39</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-full-incorporation</td>
<td>3.1-39</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-full-profile</td>
<td>3.1-39</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-grizzly</td>
<td>1.9.28-1</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-grizzly-full</td>
<td>1.9.28-1</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-gui</td>
<td>3.1-39</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-ha</td>
<td>3.1-39</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-hk2</td>
<td>3.1-39</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-javahelp</td>
<td>2.0.2-1</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-jca</td>
<td>3.1-39</td>
<td>installed</td>
<td>----</td>
</tr>
<tr>
<td>glassfish-jcdi</td>
<td>3.1-39</td>
<td>installed</td>
<td>----</td>
</tr>
</tbody>
</table>
Remove Unused Components

4 Uninstall the component that you want to remove from your system.

    pkg uninstall package-name

For example:

    pkg uninstall metro

5 Start GlassFish Server.

    See “To Start a Domain” in Oracle GlassFish Server 3.1 Administration Guide.

Remove Services You Are Not Using

Consider removing services that you are not using. For example, if applications are not using messaging, then consider removing the JMS from the server. Also consider removing EJB Container, JCA, and so forth.
**Note** – There is always a potential of making mistakes when deleting components from the GlassFish Server installation. Therefore, Oracle recommends testing your changes in a secure development environment before implementing them in a production environment.

The Updatetool and the Administration Console both provide descriptions of each installed component. In addition, the Updatetool also describes dependencies. You can use this information to decide whether you need to keep these components installed.

Before you remove a component, use the `asadmin list-<component>-resources` subcommand or the Administration Console to make sure that resources of a given type, for example JMS, are not in use. For example, you might use the `asadmin list-jms-resources` subcommand to make sure that JMS resources are not currently in use:

```bash
D:\glassfish3\glassfish\bin>asadmin list-jms-resources
Nothing to list
Command list-jms-resources executed successfully.
```

## Run on the Web Profile if Possible

If your applications can run on the Web Profile, use that instead of the Full Platform.

Java EE 6 introduced the concept of profiles. A profile is a collection of Java EE technologies and APIs that address specific developer communities and application types.

The following profiles are implemented through the distributions of GlassFish Server:

- **Full Platform** – The full Java EE platform is designed for developers who require the full set of Java EE APIs for enterprise application development, and is installed when you install GlassFish Server. This profile is also installed as part of the Java EE 6 SDK installation.

- **Web Profile** – This profile contains Web technologies that are a subset of the full Java platform, and is designed for developers who do not require the full set of Java EE APIs. This profile is also installed with Java EE 6 Web Profile SDK.

For the list of APIs in each profile, see “Java EE 6 Standards Support” in *Oracle GlassFish Server 3.1-3.1.1 Release Notes*. 
Securing the GlassFish Server Host

A GlassFish Server production environment is only as secure as the security of the machine on which it is running. It is important that you secure the physical machine, the operating system, and all other software that is installed on the host machine.

The following are recommendations for securing a GlassFish Server host in a production environment. Also check with the manufacturer of the machine and operating system for recommended security measures.

**Note** – The domain and server configuration files should be accessible only by the operating system users who configure or execute GlassFish Server.

<table>
<thead>
<tr>
<th>Security Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physically secure the hardware.</td>
<td>Keep your hardware in a secured area to prevent unauthorized operating system users from tampering with the deployment machine or its network connections.</td>
</tr>
<tr>
<td>Log out of the Administration Console before navigating to a non-secure site.</td>
<td>If you are logged on to the Administration Console, be sure to log out completely before browsing to an unknown or non-secure Web site.</td>
</tr>
<tr>
<td>Secure networking services that the operating system provides.</td>
<td>Have an expert review network services such as e-mail programs or directory services to ensure that a malicious attacker cannot access the operating system or system-level commands. The way you do this depends on the operating system you use. Sharing a file system with other machines in the enterprise network imposes risks of a remote attack on the file system. Be certain that the remote machines and the network are secure before sharing the file systems from the machine.</td>
</tr>
<tr>
<td>Use a file system that can prevent unauthorized access.</td>
<td>Make sure that the file system on each GlassFish Server host can prevent unauthorized access to protected resources. For example, on a Windows computer, use only NTFS.</td>
</tr>
<tr>
<td>Security Action</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Set file access permissions for data stored on disk. | Set operating system file access permissions to restrict access to data stored on disk. This data includes, but is not limited to, the following:  
The database files. GlassFish Server includes an implementation of Java DB (formerly known as Derby), however, you can use any JDBC-compliant database.  
The directory and filename location of a private keystore, such as keystore.jks  
The directory and filename location of a Root Certificate Authority (CA) keystore, such as cacerts.jks.  
For example, operating systems provide utilities such as umask and chmod to set the file access permissions. At a minimum, consider using "umask 066", which denies read and write permission to Group and Others. |
| Limit the number of user accounts on the host machine. | Avoid creating more user accounts than you need on host machines, and limit the file access privileges granted to each account. On operating systems that allow more than one system administrator user, the host machine should have two user accounts with system administrator privileges and one user with sufficient privileges to run GlassFish Server. Having two system administrator users provides a back up at all times. The GlassFish Server user should be a restricted user, not a system administrator user. One of the system administrator users can always create a new GlassFish Server user if needed.  
Important: Domain and server configuration files should be accessible only by the operating system users who configure or execute GlassFish Server.  
Review active user accounts regularly and when personnel leave.  
Background Information: Configuration data and some URL (Web) resources, including Java Server Pages (JSPs) and HTML pages, are stored in clear text on the file system. A sophisticated user or intruder with read access to files and directories might be able to defeat any security mechanisms you establish with authentication and authorization schemes. |
TABLE 6–1 Securing the GlassFish Server Host

<table>
<thead>
<tr>
<th>Security Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>For your system administrator user accounts, choose names that are not obvious.</td>
<td>For additional security, avoid choosing an obvious name such as “system,” “admin,” or “administrator” for your system administrator user accounts.</td>
</tr>
<tr>
<td>Safeguard passwords.</td>
<td>The passwords for user accounts on production machines should be difficult to guess and should be guarded carefully.</td>
</tr>
<tr>
<td></td>
<td>Set a policy to expire passwords periodically.</td>
</tr>
<tr>
<td></td>
<td>Never code passwords in client applications.</td>
</tr>
<tr>
<td></td>
<td>Do not deploy an application that can be accessed with the default username admin and no password.</td>
</tr>
<tr>
<td>Safeguard password files</td>
<td>The -passwordfile option of the asadmin command specifies the name of a file that contains password entries in a specific format. These password entries are stored in clear text in the password file, and rely on file system mechanisms for protection.</td>
</tr>
<tr>
<td></td>
<td>To provide additional security, create a password alias.</td>
</tr>
<tr>
<td>Use a password alias</td>
<td>A password alias stores a password in encrypted form in the domain keystore, providing a clear-text alias name to use instead of the password.</td>
</tr>
<tr>
<td></td>
<td>To provide additional security, use the create-password-alias subcommand to create an alias for the password. The password for which the alias is created is stored in an encrypted form.</td>
</tr>
<tr>
<td></td>
<td>Then, specify the alias in the entry for the password in the password file as follows:</td>
</tr>
<tr>
<td></td>
<td>In password files and the domain configuration file, use the form ${alias=alias-name} to refer to the encrypted password.</td>
</tr>
<tr>
<td>Do not run GlassFish Server as root</td>
<td>GlassFish Server should run only as an unprivileged user, never as root.</td>
</tr>
<tr>
<td></td>
<td>The directory structure in which GlassFish Server is located, including all files, should be protected from access by unprivileged users.</td>
</tr>
<tr>
<td></td>
<td>Taking these steps helps ensure that unprivileged users cannot insert code that can potentially be executed by GlassFish Server.</td>
</tr>
</tbody>
</table>
Securing GlassFish Server

GlassFish Server provides a powerful and flexible set of software tools for securing the subsystems and applications that run on a server instance. The following table provides a checklist of essential features that Oracle recommends you use to secure your production environment.

<table>
<thead>
<tr>
<th>Security Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consider use PAM Realm</td>
<td>The use of a PAM Realm requires GlassFish Server to run as an account that has read-access to a shadow password file or the equivalent, and therefore may not be suitable in your environment.</td>
</tr>
<tr>
<td>Do not develop on a production machine.</td>
<td>Develop first on a development machine and then move code to the production machine when it is completed and tested. This process prevents bugs in the development environment from affecting the security of the production environment.</td>
</tr>
<tr>
<td>Do not install development or sample software on a production machine.</td>
<td>Do not install development tools on production machines. Keeping development tools off the production machine reduces the leverage intruders have should they get partial access to a production machine.</td>
</tr>
<tr>
<td>Enable security auditing.</td>
<td>If the operating system on which GlassFish Server runs supports security auditing of file and directory access, Oracle recommends using audit logging to track any denied directory or file access violations. Administrators should ensure that sufficient disk space is available for the audit log.</td>
</tr>
<tr>
<td>Consider using additional software to secure your operating system.</td>
<td>Most operating systems can run additional software to secure a production environment. For example, an Intrusion Detection System (IDS) can detect attempts to modify the production environment. Refer to the vendor of your operating system for information about available software.</td>
</tr>
<tr>
<td>Apply operating system patch sets and security patches.</td>
<td>Refer to the vendor of your operating system for a list of recommended patch sets and security-related patches.</td>
</tr>
<tr>
<td>Apply the latest maintenance packs and critical patch updates.</td>
<td>Refer to the vendor of your operating system for a list of maintenance packs and critical patch updates.</td>
</tr>
</tbody>
</table>

**TABLE 6–1** Securing the GlassFish Server Host *(Continued)*
### Table 6-2: Securing GlassFish Server

<table>
<thead>
<tr>
<th>Security Action</th>
<th>Description</th>
</tr>
</thead>
</table>
| Enable Secure Admin.                                 | The secure administration feature allows an administrator to secure all administrative communication between the domain administration server (DAS), any remote instances, and administration clients such as the asadmin utility, the administration console, and REST clients. In addition, secure administration helps to prevent DAS-to-DAS and instance-to-instance traffic, and carefully restricts administration-client-to-instance traffic. The secure administration feature provides a secure environment, in which you can be confident that rogue users or processes cannot intercept or corrupt administration traffic or impersonate legitimate GlassFish Server components.
|                                                       | See Chapter 5, "Managing Administrative Security"                                                                                                                                                                                                                                                                                                                                                     |
| Protect the .asadminpass file                         | If you create a domain with the `--save-login` option, `create-domain` saves the administration user name and password in the .asadminpass file in the user's home directory. Make sure that this file remains protected. Information stored in this file will be used by asadmin commands to manage this domain.                                                                                                                                                                                  |
| Deploy production-ready security providers to the security realm. | 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 GlassFish Server provides a simple, file-based authorization engine that complies with the JACC specification. You can also specify additional third-party JACC providers.
<p>|                                                       | If you have purchased or written your own security providers, make sure that you have deployed and configured them properly.                                                                                                                                                                                                                           |</p>
<table>
<thead>
<tr>
<th>Security Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use SSL, but do not use the self-signed certificates in a production environment.</td>
<td>To prevent sensitive data from being compromised, secure data transfers by using HTTPS. By default, GlassFish Server uses self-signed certificates. The self-signed certificates that GlassFish Server uses might not be trusted by clients by default because a certificate authority does not vouch for the authenticity of the certificate. You can instead use your own certificates, as described in &quot;Using Your Own Certificates&quot; on page 96.</td>
</tr>
<tr>
<td>Restrict the size and the time limit of requests on external channels to prevent Denial of Service attacks.</td>
<td>To prevent some Denial of Service (DoS) attacks, restrict the size of a message as well as the maximum time it takes a message to arrive. The default setting for maximum post size is 2097152 bytes and 900 seconds for the request timeout.</td>
</tr>
<tr>
<td>Enable authentication and authorization auditing.</td>
<td>Auditing is the process of recording key security events in your GlassFish Server environment. You use audit modules to develop an audit trail of all authentication and authorization decisions. To enable audit logging, two steps are required: 1. On the Security page, select the Audit Logging Enabled checkbox to enable audit logging. 2. Set the auditOn property for the active audit module to true. Review the auditing records periodically to detect security breaches and attempted breaches. Noting repeated failed logon attempts or a surprising pattern of security events can prevent serious problems.</td>
</tr>
<tr>
<td>Set logging for security and SSL messages.</td>
<td>Consider setting module log levels for table.javax.enterprise.system.ssl.security and javax.enterprise.system.core.security. You can set a level from Severe to Finest (the default is Info), but be aware that the finer logging levels may produce a large log file. By default, GlassFish Server logging messages are recorded in the server log, and you can set the file rotation limit, as described in rotate-log(1)</td>
</tr>
<tr>
<td>Ensure that you have correctly assigned users to the correct groups.</td>
<td>Make sure you have assigned the desired set of users to the right groups. In particular, make sure that users assigned to the asadmin group need to be members of that group.</td>
</tr>
</tbody>
</table>
### TABLE 6–2  Securing GlassFish Server (Continued)

<table>
<thead>
<tr>
<th>Security Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create no fewer than two user accounts in the asadmin group.</td>
<td>The user admin is created when you install GlassFish Server. For production environments, create at least one other account in the asadmin group in case one account password is compromised. When creating asadmin users give them unique names that cannot be easily guessed.</td>
</tr>
<tr>
<td>Assign a password to the admin account.</td>
<td>By default, GlassFish Server includes a single account for user “admin” and an empty password. For production environments this default is inherently unsecure, and you should set a password for admin.</td>
</tr>
</tbody>
</table>

### Securing Applications

Although much of the responsibility for securing the GlassFish Server resources in a domain fall within the scope of the server, some security responsibilities lie within the scope of individual applications. For some security options, GlassFish Server enables you to determine whether the server or individual applications are responsible. For each application that you deploy in a production environment, review the items in the following table to verify that you have secured its resources.

### TABLE 6–3  Securing Applications

<table>
<thead>
<tr>
<th>Security Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use JSP comment tags instead of HTML comment tags.</td>
<td>Comments in JSP files that might contain sensitive data and or other comments that are not intended for the end user should use the JSP syntax of <code>&lt;%/* xxx */%&gt;</code> instead of the HTML syntax <code>&lt;!-- xxx --&gt;</code>. The JSP comments, unlike the HTML comments, are deleted when the JSP is compiled and therefore cannot be viewed in the browser.</td>
</tr>
<tr>
<td>Do not install uncompiled JSPs and other source code on the production machine.</td>
<td>Always keep source code off of the production machine. Getting access to your source code allows an intruder to find security holes. Consider precompiling JSPs and installing only the compiled JSPs on the production machine. To do this, set the <code>deploy</code> subcommand <code>-precompilejsp</code> option to true for the component. When set to true, the <code>deploy</code> and <code>redeploy</code> subcommands <code>-precompilejsp</code> option compiles JSPs during deploy time. If set to false (the default), JSPs are compiled during runtime.</td>
</tr>
</tbody>
</table>
### TABLE 6–3  Securing Applications (Continued)

<table>
<thead>
<tr>
<th>Security Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure your applications to use SSL.</td>
<td>Set the transport-guarantee to CONFIDENTIAL in the user-data-constraint element of the web.xml file whenever appropriate.</td>
</tr>
<tr>
<td>Examine applications for security.</td>
<td>There are instances where an application can lead to a security vulnerability. Of particular concern is code that uses Java native interface (JNI) because Java positions native code outside of the scope of Java security. If Java native code behaves errantly, it is only constrained by the operating system. That is, the Java native code can do anything GlassFish Server itself can do. This potential vulnerability is further complicated by the fact that buffer overflow errors are common in native code and can be used to run arbitrary code.</td>
</tr>
<tr>
<td>If your applications contain untrusted code, enable the Java security manager.</td>
<td>The Java security manager defines and enforces permissions for classes that run within a JVM. In many cases, where the threat model does not include malicious code being run in the JVM, the Java security manager is unnecessary. However, when third parties use GlassFish Server and untrusted classes are being run, the Java security manager may be useful. See &quot;Enabling and Disabling the Security Manager&quot; in Oracle GlassFish Server 3.1 Application Development Guide.</td>
</tr>
<tr>
<td>Replace HTML special characters when servlets or JSPs return user-supplied data.</td>
<td>The ability to return user-supplied data can present a security vulnerability called cross-site scripting, which can be exploited to steal a user's security authorization. For a detailed description of cross-site scripting, refer to &quot;Understanding Malicious Content Mitigation for Web Developers&quot; (a CERT security advisory) at <a href="http://www.cert.org/tech_tips/malicious_code_mitigation.html">http://www.cert.org/tech_tips/malicious_code_mitigation.html</a>. To remove the security vulnerability, before you return data that a user has supplied, scan the data for HTML special characters. If you find any such characters, replace them with their HTML entity or character reference. Replacing the characters prevents the browser from executing the user-supplied data as HTML.</td>
</tr>
</tbody>
</table>
This chapter provides instructions for integrating Oracle Access Manager (OAM) in the Oracle GlassFish Server environment via the OAM Security Provider.

This chapter assumes that you are familiar with security features such as authentication, authorization, and certificates. If you are not, see Chapter 1, “Administering System Security.”

The chapter additionally assumes that you are familiar with Oracle Access Manager features, concepts, and components, and it summarizes the Oracle Access Manager configuration steps that you must perform. See the Oracle Access Manager Installation Guide (http://download.oracle.com/docs/cd/B28196_01/idmanage.1014/b25353/toc.htm) for detailed instructions on Access Manager tasks.

The following topics are addressed here:

- “About OAM Security Provider for Glassfish” on page 122
- “About Oracle Access Manager” on page 123
- “Understanding OAM Security Provider Use Cases” on page 125
- “Configuring the OAM Security Provider” on page 130
- “Determining Which Authentication Method is Used” on page 132
- “Integrating OAM Security Provider with Oracle Access Manager 10g” on page 133
- “Integrating OAM Security Provider with Oracle Access Manager 11g” on page 147
- “Integrating OAM Security Provider with Oracle Access Manager 11g and WebGate” on page 164

Instructions for accomplishing the tasks specific to GlassFish Server by using the Administration Console are contained in the Administration Console online help.
About OAM Security Provider for Glassfish

The OAM Security Provider for GlassFish Server is a JSR 196 Server Authentication Module (SAM) that provides authentication, authorization (optional), and single sign-on across JavaEE Web applications that are deployed on GlassFish Server. (See JSR 196: JavaTM Authentication Service Provider Interface for Containers (http://www.jcp.org/en/jsr/detail?id=196) for the specification and the related Javadocs.)

The OAM Security Provider enables GlassFish Server administrators to use Oracle Access Manager to control user access to business applications. The OAM Security Provider enables the following Oracle Access Manager functions for GlassFish Server users:

- **Authenticator** – This security provider uses Oracle Access Manager authentication services to authenticate users who access JavaEE applications deployed on GlassFish Server. Users are authenticated based on their credentials, such as user name and password, or client certificate.

  Authentication occurs when GlassFish Server directly receives unauthenticated requests from end-user clients for applications deployed on GlassFish Server. The OAM Security Provider challenges the user, collects the credentials and sends them to Oracle Access Manager via the configured AccessGate. The form of challenge issued (BASIC, FORM, Client-Certificate) and the security characteristics of the transport (SSL or PlainText) depend on the policies configured at Oracle Access Manager for the resource being accessed.

  Oracle Access Manager authenticates the credentials and, upon successful authentication, create an OBSSOSession. The OAM Security Provider receives this session and sets its identifier as an ORA_GF_ObSSOCookie in the response. This allows subsequent requests from the client for other resources protected by the same policy-domain to not require authentication, thereby achieving single sign-on (SSO) access to resources.

- **Identity Asserter** – The security provider uses Oracle Access Manager authentication services to validate already-authenticated Oracle Access Manager users using the OAM_REMOTE_USER header and creates an authenticated session.

  Identity assertion occurs when GlassFish Server is behind a proxy Oracle HTTP Server (OHS) that has an installed WebGate. The WebGate is responsible for challenging the user requests for resources. OHS then passes the OAM_REMOTE_USER header to the GlassFish Server.

  The OAM Security Provider tries to assert the identity of the user that was sent in the OAM_REMOTE_USER header.

- **Authorizer** – In this mode the OAM Security Provider additionally uses Oracle Access Manager authorization services to authorize users who are accessing a protected resource. The authorization is based on Oracle Access Manager policies.
Obtaining Oracle Access Manager Group Information

In the authentication function, the OAM Security Provider also tries to obtain group membership information for the authenticated user from the Oracle Access Manager backend. This information is set in a SecurityContext, which is then passed to the GlassFish Server authorization system to determine if access to the resource should be allowed.

The backend query is integrated into the JSR-196 CallerPrincipal Callback handling in the container and is transparent to the OAM Security Provider.

To obtain the group information, you configure an LDAPRealm in GlassFish Server to point to the OAM backend where the group information is stored. For example, you can use the Administration Console or the `create-auth-realm` command to add the following entry to the domain-specific domain.xml file:

```xml
<auth-realm
classname="com.sun.enterprise.security.auth.realm.ldap.LDAPRealm"
name="ldaprealm">
  <property name="jaas-context" value="ldapRealm" />
  <property name="base-dn" value="o=company,c=us" />
  <property name="directory" value="ldap://140.87.134.98:1389" />
  <property name="search-bind-dn" value="cn=Directory Manager" />
  <property name="search-bind-password" value="welcome1" />
</auth-realm>
```

About Oracle Access Manager

Oracle Access Manager allows users of your applications to log in once and gain access to a broad range of resources. Oracle Access Manager provides an identity management and access control system that is shared by all of your applications. The result is a centralized and automated single sign-on (SSO) solution for managing who has access to what information across your entire IT infrastructure.

To integrate the OAM Security Provider with Oracle Access Manager, you need to understand the Oracle Access Manager component concepts shown in Table 7-1.

Note – The OAM Security Provider integrates with versions 10g and 11g of Oracle Access Manager. Between these two versions of Oracle Access Manager, many component names remain the same. However, there are several important changes that you should know about, as described in Product and Component Name Changes (http://download.oracle.com/docs/cd/E14571_01/doc.1111/e15478/whatsnew.htm#CJADIIBF) in the Oracle Fusion Middleware Administrator's Guide for Oracle Access Manager.
## Table 7-1  Oracle Access Manager Concepts

<table>
<thead>
<tr>
<th>Oracle Access Manager 10g</th>
<th>Oracle Access Manager 11g</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access System</td>
<td>Access System</td>
<td>The Access System is the access-control system that provides single sign-on across any Web application. It supports a variety of access policies, and is fully integrated with the Identity System so that changes in user profiles are instantly reflected in the Access System's policy enforcement. The Access System consists of the Policy Manager, Access Server, and WebGate.</td>
</tr>
<tr>
<td>Access Server</td>
<td>OAM Server</td>
<td>Access Server receives requests and then queries authentication, authorization, and auditing rules in the directory server. Based on the information in the directory server, the Access Server sends the authentication scheme, validates user credentials, authorizes the user, audits, and manages the session.</td>
</tr>
<tr>
<td>Policy Manager</td>
<td>OAM Administration Console</td>
<td>The Policy Manager provides a Web-based interface where administrators can create and manage access policies. The Policy Manager also communicates with the directory server to write policy data, and communicates with the Access Server when certain policy modifications are made.</td>
</tr>
<tr>
<td>WebGate</td>
<td>OAM Agent</td>
<td>WebGate is a server plug-in Oracle Access Manager access client that intercepts HTTP requests for Web resources and forwards them to the Access Server for authentication and authorization. A WebGate is shipped out-of-the-box with Oracle Access Manager.</td>
</tr>
<tr>
<td>AccessGate</td>
<td>OAM Agent</td>
<td>An AccessGate is a custom access client that is specifically developed using the Software Development Kit (SDK) and Oracle Access Manager APIs, either by the customer or by Oracle. An AccessGate processes requests for Web and non-Web resources (that is, non-HTTP) from users or applications.</td>
</tr>
</tbody>
</table>

The component names described in this section reflect the 10g version of Oracle Access Manager.
Understanding OAM Security Provider Use Cases

With the OAM Security Provider, you can use Oracle Access Manager to protect JavaEE Web Application resources. You can configure single sign-on for Web applications, such that a user who has authenticated to GlassFish Server can access Oracle Access Manager-protected resources without re-authentication. You can also configure single sign-on such that a user who has authenticated to Oracle Access Manager can access GlassFish Server Web resources without re-authentication. The OAM Security Provider can also be optionally configured to perform resource authorization using policies configured at the Oracle Access Manager Policy Manager.

You can use the provider in the following ways:

- To provide authentication for Web resources via an AccessGate.
- To provide authentication (Identity Assertion) for Web resources using a proxy server with a WebGate installed.
- To indicate that authorization checks are based on Policy Manager.

Use cases for each scenario are described next.

Use Case: Authentication for Web Resources Via Access Gate

The OAM Security Provider uses Oracle Access Manager authentication services to authenticate users who access JavaEE applications deployed on GlassFish Server. This method can be used to protect HTTP resources deployed on GlassFish Server. Users are authenticated based on their credentials, such as user name and password (BASIC/Form) and client certificate.

The OAM Security Provider uses an AccessGate to consult an AccessServer configured at the Oracle Access Manager instance to do authentication. Upon successful authentication, the OAM Security Provider sets a cookie (ORA_GF_ObSSOCookie) to represent the single sign-on (SSO) session created by Oracle Access Manager.

Consider the flow of control shown in Figure 7–1.
The flow of control shown in Figure 7.1 is as follows:

1. A user attempts to access an Oracle Access Manager-protected GlassFish Server resource.
2. The GlassFish Server challenges the user for a username and password using a predefined login form in the case of FORM authentication (or a challenge in the case of BASIC authentication) because the application’s deployment descriptor requires authentication from the container. You can use your own login form, which can be specified as an option to the OAM Security Provider.
3. The GlassFish Server forwards the username and password to the OAM Security Provider for authentication and authorization.
4. The OAM Security Provider uses the AccessGate to communicate with the Access Server to verify the user’s identity. If authentication is successful, the Access Server generates a session token with a URL that contains the ORA_GF_ObSSOCookie.
5. If authentication is successful, the OAM Security Provider creates a Subject with the authenticated principal and also sets the cookie in the response.
   The ORA_GF_ObSSOCookie is set so that when the user attempts to access additional Oracle Access Manager-protected non-GlassFish Server resources, authentication is not performed. That is, if the ORA_GF_ObSSOCookie is already set and the user has logged in using form-based authentication, the user is logged in without being challenged.
6. The control returns to the GlassFish Server authorization mechanism, which checks if the user has permission to access the requested resource. The policies that protect resources are specified in the Policy Manager application in Oracle Access Manager. Policies that are
defined in web.xml and glassfish-web.xml are honored by default and authorization based on policies at the Policy Manager can be configured as an option. When the OAM Security Provider is configured to do additional authorization checks based on policies at the Policy Manager, the resource access is disallowed if the policies defined in web.xml are not also satisfied.

7. If authorization is successful, the GlassFish Server allows the user to access the requested resource.

**Use Case: Identity Assertion for Web Resources via WebGate**

This use case occurs when GlassFish Server is behind an Oracle HTTP Server proxy that has an installed WebGate, and there is a connector to the GlassFish Server Web container from the proxy.

The WebGate is responsible for challenging the user requests for resources, and it is assumed that all requests to the Web container have to come through the configured proxy. The Web resources are protected using authentication schemes such as FORM, BASIC, or CLIENT-CERT supported by the WebGate. The WebGate protects only Web resources (HTTP).

Consider the flow of control shown in Figure 7–2.
The flow of control shown in Figure 7–2 is as follows:

1. A user attempts to access an Oracle Access Manager-protected Web application that is deployed on the GlassFish Server. (The WebGate protects only Web resources (HTTP).)

2. The WebGate intercepts the request and queries the Access Server to check if the resource is protected.

3. If the resource is protected, the WebGate challenges the user for credentials based on the type of Oracle Access Manager authentication scheme configured for the resource, which is CLIENT CERT in this use case.

4. The user presents a certificate.

5. If the user authenticates successfully, the Oracle Access Manager sets the OAM_REMOTE_USER header in the request. Note the following cookie and header use:
   - For a 10g WebGate:
     - The ObSSOCookie is always propagated.
     - The OAM_REMOTE_USER header is always set by an OAM 11g server.
   - For an 11g Webgate:
     - A cookie is not propagated.
     - The OAM_REMOTE_USER header is always set by an OAM 11g server

6. The request is redirected to the GlassFish Server via the Oracle HTTP Server (OHS) redirect directive.
7. The OAM Security Provider looks at the OAM_REMOTE_USER header and gets the authenticated user identity.

8. The OAM Security Provider creates a Subject with the authenticated principal.

9. Authorization based on policies at the Policy Manager in Oracle Access Manager can be configured as an option.

10. (Not shown.) If authorization is successful, the GlassFish Server enables the user to access the requested resource.

**Use Case: Authorization Checks Based on Policy Manager**

The oam.check.resource.access parameter to the OAM Security Provider (see Table 7-2) indicates whether Oracle Access Manager should perform authorization checks for the resource.

This authorizer function is optional.

By default, GlassFish Server uses its container authorization mechanisms to make the final resource access decision. GlassFish Server uses the Subject set by the OAM Security Provider (after successful authentication) to perform the authorization step.

If you set the oam.check.resource.access parameter for the OAM Security Provider to true, the Oracle Access Manager policies that protect the resources are invoked. (You specify these policies in the Policy Manager application in Oracle Access Manager.)

**Note** – Even when the OAM Security Provider is configured to do additional authorization checks based on policies at the Policy Manager, resource access is still disallowed if the policies defined in web.xml are not also satisfied.

The authorization flow of control is as follows:

1. A user attempts to access an Oracle Access Manager-protected GlassFish Server resource.

2. The GlassFish Server challenges the user for a username and password using a predefined login form in the case of FORM authentication (or a challenge in the case of BASIC authentication) because the application’s deployment descriptor requires authentication from the container. You can use your own login form, which can be specified as an option to the OAM Security Provider.

3. The GlassFish Server forwards the username and password to the OAM Security Provider for authentication and authorization.
4. The OAM Security Provider uses the AccessGate to communicate with the Access Server to verify the user's identity and to see if the user has access to the protected resource, as determined by a policy in the Policy Manager. (The policies that protect resources are specified in the Policy Manager application in Oracle Access Manager.) If authentication is successful and the user has access to the resource, the Access Server generates a session token with a URL that contains the ORA_GF_ObSSOCookie.

5. If authentication is successful, the OAM Security Provider creates a Subject with the authenticated principal and also sets the cookie in the response. The ORA_GF_ObSSOCookie is set so that when the user attempts to access additional Oracle Access Manager-protected resources, authentication is not performed. That is, if the ORA_GF_ObSSOCookie is already set and the user has logged in using form-based authentication, the user is logged in without being challenged.

6. The control returns to the GlassFish Server authorization mechanism, which checks whether the user has permission to access the requested resource. When the OAM Security Provider is configured to do additional authorization checks based on policies at the Policy Manager, final access to a resource is allowed only when Policy Manager policies and web.xml policies are both satisfied.

7. If authorization is successful, the GlassFish Server allows the user to access the requested resource.

**Configuring the OAM Security Provider**

The OAM Security Provider is implemented via the OAMAuthenticatorSAM class.

You configure the OAM Security Provider as a message-security-provider (HttpServlet layer interception point), either through the Administration Console or the create-message-security-provider command.

The following message-security-provider syntax configures an OAM Security provider:

```xml
<message-security-config auth-layer="HttpServlet">
  <provider-config provider-type="server" provider-id="MySAM" class-name="com.sun.glassfish.oamsam.OAMAuthenticatorSAM">
    <property name="oam.resource.hostid.variation" value="your-host-system.com" />
  </provider-config>
</message-security-config>
```

You must set the provider id to a unique value. The class name must be `com.sun.glassfish.oamsam.OAMAuthenticatorSAM`.

You can optionally set the properties shown in Table 7–2 for the OAM Security Provider.
### TABLE 7–2  OAM Security Provider Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPSERVER_SSL_MUTUAL_AUTH_PORT</td>
<td>Option to the SAM indicating the SSL Mutual Authentication port for the Application Server.</td>
</tr>
<tr>
<td>APPSERVER_SSL_PORT</td>
<td>Option to the SAM indicating the SSL port for the Application Server.</td>
</tr>
<tr>
<td>cookie.httponly</td>
<td>Option to the SAM indicating if the httponly flag of the ORA_GF_ObSSOCookie should be set.</td>
</tr>
<tr>
<td>cookie.path</td>
<td>Option to the SAM indicating the value of the path parameter of the ORA_GF_ObSSOCookie. By default the path parameter is not set.</td>
</tr>
<tr>
<td>cookie.secure</td>
<td>Option to the SAM indicating if the isSecure() bit of the Cookie should be set to true/false when the ORA_GF_ObSSOCookie is created after successful authentication.</td>
</tr>
<tr>
<td>error.page</td>
<td>Option to the SAM indicating the custom Error Page to use for Authentication/Authorization failures. Should be a path relative to current context root, and must begin with a “/”.</td>
</tr>
<tr>
<td>form.login.page</td>
<td>Option to the SAM indicating the custom FORM Login Page to use for FORM based login. Must be a path relative to the domain’s docroot. A default login FORM is used when this option is absent, or when its value is set to ”” (empty string) or to the value “default”.</td>
</tr>
<tr>
<td>oam.auth.cert.param</td>
<td>Option to the SAM indicating the parameter name for certificate credential passed to create ObUserSession.</td>
</tr>
<tr>
<td>oam.auth.hdr.external</td>
<td>Option to the SAM whose value indicates the name of Request Header that contains the Userld/Cookie corresponding to the ObUserSession.</td>
</tr>
<tr>
<td>oam.auth.password.param</td>
<td>Option to the SAM indicating the parameter name for password credential passed to create ObUserSession.</td>
</tr>
<tr>
<td>oam.auth.userid.param</td>
<td>Option to the SAM indicating the parameter name for username credential passed to create ObUserSession.</td>
</tr>
<tr>
<td>oam.check.resource.access</td>
<td>Option to the SAM indicating if OAM should also perform the authorization check for the resource. The value is a flag that indicates if OAM should check (authenticate and authorize) the resource access. By default OAM would only be consulted for authentication of any credentials and establishing the SSO token.</td>
</tr>
<tr>
<td>Note</td>
<td>The GlassFish Server authorization has the final say as to whether the access to the resource is finally allowed, primarily based on the group and principal of the user being in the correct JavaEE-defined Role to access the resource.</td>
</tr>
<tr>
<td>oam.header.type</td>
<td>Option to the SAM indicating if the External Header in the Request is to be interpreted as a Cookie or as the User Principal of an Authenticated user when the module is acting as an Identity Asserter.</td>
</tr>
</tbody>
</table>
Determining Which Authentication Method is Used

The JavaEE Web applications you want to protect should be configured with deployment descriptors containing required `<security-constraint>` and associated `<auth-constraint>` specifying the roles. The descriptors should not contain `<login-config>` elements that specify the JavaEE supported authentication methods.

Instead, the OAM Security Provider determines the authentication mechanism to used based on the challenge method of the Authentication Scheme you configure for the resource in Oracle Access Manager. For example consider the Authentication Scheme shown in Figure 7–3, which uses the BASIC challenge method.
The authentication mechanisms supported by the OAM Security Provider are BASIC, FORM or Client-Cert. The default is BASIC. The Authentication Schemes supported at Oracle Access Manager include BASIC, FORM and Client-Cert. The use of the SSL transport is optional for BASIC and FORM.

**Integrating OAM Security Provider with Oracle Access Manager 10g**

This section describes how to integrate the OAM Security Provider with Oracle Access Manager 10g.
Integrating OAM Security Provider with Oracle Access Manager 10g

This procedure explains how to integrate the OAM Security Provider with Oracle Access Manager 10g.

See the Oracle Access Manager Installation Guide (http://download.oracle.com/docs/cd/B28196_01/idmanage.1014/b25353/toc.htm) for detailed instructions on Access Manager tasks.

The procedure was tested on Microsoft Windows and the file names are those of the Windows version.

1 Install and configure Oracle Access Manager.
   To do this, install Oracle Identity Server, the Policy Manager, and the Oracle Access Server component. You need to obtain all of the required Oracle Access Manager components before you begin the installation procedures.

2 Log in to Administration Console.
   After the successful installation, access the Administration Console from http://host:port/access/oblix and log in.

   The administrative interface for managing core access server components, policy manager, and identity console is displayed.

3 Create users.
   From the User Manager tab of the Identity System Console, create the users who will have access to the protected resource. For example, you might create the user Glassfish, as shown in Figure 7–4.
Createthegroup.
FromtheGroupManagertaboftheIdentitySystemConsole,createthegrouptowhichyouwill
grantaccessandadduserstothegroup.Forexample,youmightcreatethegroup
Glassfish_group,andaddtheuserGlassfishtothisgroup,asshownin
Figure7–5.

4 Create the group.
From the Group Manager tab of the Identity System Console, create the group to which you will
grant access and add users to the group. For example, you might create the group
Glassfish_group, and add the user Glassfish to this group, as shown in Figure 7–5.
5 **Create Access Server configuration.**

From the Access System Configuration tab of the Access System Console, create an Access Server Configuration.

Provide the required parameters including Name and Hostname, and enable it.

When you configure an Access Server, you specify the transport security mode to use between an Access Gate/WebGate and this Access Server, from the following choices:

- Open mode means no encryption.
- Simple mode means encryption through SSL and a public key certificate provided by Oracle.
- Cert mode means encryption through SSL and a public key certificate provided by an external CA.

For example, a sample Access Server configuration, GlassfishAS, is shown in Figure 7-6.
Create Access Gate configuration.

From the Access System Configuration tab of the Access System Console, create an Access Gate configuration.

Provide the required parameters including Name, Hostname (name of the system that will host the AccessGate), and enable it. For example, a sample AccessGate configuration, GlassfishAG, is shown in Figure 7–7.

<table>
<thead>
<tr>
<th>Details for Access Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Hostname</td>
</tr>
<tr>
<td>Port</td>
</tr>
<tr>
<td>Debug</td>
</tr>
<tr>
<td>Debug File Name</td>
</tr>
<tr>
<td>Transport Security</td>
</tr>
<tr>
<td>Maximum Client Session Time (hours)</td>
</tr>
<tr>
<td>Number of Threads</td>
</tr>
<tr>
<td>Access Management Service</td>
</tr>
<tr>
<td>Audit to Database (on/off)</td>
</tr>
<tr>
<td>Audit to File (on/off)</td>
</tr>
<tr>
<td>Audit File Name</td>
</tr>
<tr>
<td>Audit File Size (bytes)</td>
</tr>
<tr>
<td>Buffer Size (bytes)</td>
</tr>
<tr>
<td>File Rotation Interval (seconds)</td>
</tr>
<tr>
<td>Engine Configuration Refresh Period (seconds)</td>
</tr>
<tr>
<td>URL Prefix Reload Period (seconds)</td>
</tr>
<tr>
<td>Password Policy Reload Period (seconds)</td>
</tr>
</tbody>
</table>

6 Create Access Gate configuration.

From the Access System Configuration tab of the Access System Console, create an Access Gate configuration.

Provide the required parameters including Name, Hostname (name of the system that will host the AccessGate), and enable it. For example, a sample AccessGate configuration, GlassfishAG, is shown in Figure 7–7.
Associate AccessGate and Access Server.

From the Access System Configuration tab of the Access System Console, associate the Access Gate and the Access Server.

To do this, from the AccessGate configuration page, click List Access Servers and then add the Access Server you previously created.
8 Add Host Identifiers.

From the Access System Configuration tab of the Access System Console, add one or more host identifiers.

You use Host identifiers to simplify the identification of a Web server that hosts resources you want to protect with Access Manager.

For example, when you added the configuration for the AccessGate, assume that you used the hostname of the system hosting the AccessGate. You might want to add a host identifier for the IP address or the fully-qualified domain name of the system hosting the AccessGate.
Create an Authentication Scheme.

From the Access System Configuration tab of the Access System Console, create an authentication scheme for each login mechanism.

Create an authentication scheme for each login mechanism (BASIC, Form, client cert) that you want to use to protect the resource, save it, and then add a credential mapping plug-in (from the Plugins tab) for the authentication scheme.

For BASIC authentication, the credential mapping plug-in maps the LDAP attributes of the user (shown in the figure as genuserid=%userid% and obCredentialPassword) with the challenge parameters (shown in the figure as realm:LDAP User Name/Password) of the authentication scheme, and they must match.

Figure 7–10 and Figure 7–11 show a sample BASIC over LDAP authentication scheme, and the respective plug-ins.
For client cert authentication, a sample authentication scheme is shown in Figure 7-12.

**FIGURE 7-10** BASIC Authentication Scheme

<table>
<thead>
<tr>
<th>Details for Authentication Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Level</td>
</tr>
<tr>
<td>Challenge Method</td>
</tr>
<tr>
<td>Challenge Parameter</td>
</tr>
<tr>
<td>SSL Required</td>
</tr>
<tr>
<td>Challenge Redirect</td>
</tr>
<tr>
<td>Enabled</td>
</tr>
</tbody>
</table>

**FIGURE 7-11** Plug-Ins for BASIC Authentication Scheme

<table>
<thead>
<tr>
<th>Plugins for Authentication Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugin Name</td>
</tr>
<tr>
<td>-----------------------------------</td>
</tr>
<tr>
<td>credential_mapping</td>
</tr>
<tr>
<td>validate_password</td>
</tr>
</tbody>
</table>

For client cert authentication, a sample authentication scheme is shown in Figure 7-12.
In this mapping, the LDAP attribute `employeetype` is mapped to the certificate subject `certSubject.CN`.

**FIGURE 7–12  Client Cert Authentication Scheme**

<table>
<thead>
<tr>
<th>Details for Authentication Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Level</strong></td>
</tr>
<tr>
<td><strong>Challenge Method</strong></td>
</tr>
<tr>
<td><strong>Challenge Parameter</strong></td>
</tr>
<tr>
<td><strong>SSL Required</strong></td>
</tr>
<tr>
<td><strong>Challenge Redirect</strong></td>
</tr>
<tr>
<td><strong>Enabled</strong></td>
</tr>
</tbody>
</table>

For client cert authentication, the associated plug-in configuration is shown in Figure 7–13.
Create a Policy Domain.

From the Policy Manager, create a policy domain.

The Policy Manager application allows you to create, remove and manage policies and resources and test policy enforcement.

Before a policy domain can control access to your resources, you must:

a. **Add a resource to be protected.**
   
   From the Resources tab, click Add. Select the host identifier for the system that hosts the AccessGate and the resource to be protected.

   The URL prefixes provide the pathway to your Web resources. A policy domain can contain one or more URL prefixes. The available URL prefixes depend on the resource type you selected. You can also specify a unique URL prefix in the associated text field; for example, glassfish-default.

   If you want to use the Policy Manager authorization function to protect resources, you can configure the resources individually. If you want to use GlassFish Server for authorization, and use Oracle Access Manager only for authentication, you can use a proxy resource.

b. **Add an authentication rule.**

   From the Default Rules tab, select the Authentication Rule tab and add enter an authentication rule. Choose the authentication scheme you configured in Step 10. For example, Basic Over LDAP.

c. **Create one or more authorization rules.**

   From the Authorization Rules tab, click Add and enter an authorization rule. Save the change. Then, from the Allow Access tab, add the user or group you created in Step 4 and Step 5.

d. **Create an authorization expression.**

   From the Default Rules tab, select the Authorization Expression tab. Click Add to add an authorization expression.
e. Optionally, create one or more policies to fine-tune access control for the resource.
   Provide a name, choose the operations for the policy, and choose the configured resource (Host Identifier and URL Prefix).

11 Download and unzip the Oracle Access Server SDK.
You can download the SDK from Oracle Identity Management 10g (10.1.4.x) Downloads (http://www.oracle.com/technetwork/middleware/ias/downloads/101401-099957.html).

Select Oracle Access Manager Core Components (10.1.4.3.0) (Disk1 (http://download.oracle.com/otn/nt/middleware/11g/ofm_oam_core_win_10.1.4.3.0_disk1_1of1.zip)).

The Access Server SDK allows Java, C++, and C applications to use the services of the Oracle Access Manager, including user authentication and authorization of user requests to access resources protected by Oracle Access Manager.

The following components are installed from the zip file:
   - oamcfgtool_10_1_4_3_0.zip
   - Oracle_Access_Manager10_1_4_3_0_Win32-dotnet20_AccessServerSDK.exe
   - Oracle_Access_Manager10_1_4_3_0_Win32_AccessServerSDK.exe
   - Oracle_Access_Manager10_1_4_3_0_Win32_Access_Server.exe
   - Oracle_Access_Manager10_1_4_3_0_Win32_Identity_Server.exe
   - Oracle_Access_Manager10_1_4_3_0_Win32_OHS11g_Policy_Manager.exe
   - Oracle_Access_Manager10_1_4_3_0_Win32_OHS11g_WebPass.exe
   - Oracle_Access_Manager10_1_4_3_0_Win32_Snmp_Agent.exe

12 Run the installation wizard.
Run Oracle_Access_Manager10_1_4_3_0_Win32_AccessServerSDK.exe on the system from which you intend to run the GlassFish Server hosted application and follow the installation wizard.

   **Note** – Depending on the platform, additional steps may be required to configure the AccessServerSDK, such as setting LD_LIBRARY_PATH and so forth. Please consult the ASDK_README.htm file under installdirectory\AccessServerSDK for more details.

13 Configure the AccessServer SDK.
Configure the AccessServer SDK to facilitate the handshake between the AccessGate you created and Oracle Access Manager.

To do this, run the utility configureAccessGate.exe, located in
installdirectory\AccessServerSDK\oblx\tools\configureAccessGate, where installdirectory is the root folder for your Access Manager SDK installation.

```
configureAccessGate -i <AccessServerSDK-install-dir>
-t AccessGate -w GlassfishAG -m open -h access-server-host-name -p 6021 -a GlassfishAS
```
Please enter the Password for this AccessGate:

Preparing to connect to Access Server. Please wait.

AccessGate installed Successfully.

The command line parameters shown are as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Installation directory for AccessServerSDK.</td>
</tr>
<tr>
<td>-t</td>
<td>AccessGate keyword. Enter as shown.</td>
</tr>
<tr>
<td>-w</td>
<td>AccessGate name you specified when you created the AccessGate entry in Oracle Access Manager.</td>
</tr>
<tr>
<td>-m</td>
<td>Oracle Access Manager intercomponent mode. This is the security mode you specified between the WebGate/Access client and the Access Server. The valid values are open, simple, or cert. Open mode means no encryption. Simple mode means encryption through SSL and a public key certificate provided by Oracle. Cert mode means encryption through SSL and a public key certificate provided by an external CA.</td>
</tr>
<tr>
<td>-h</td>
<td>Fully-qualified name of the host on which you installed the Access Server.</td>
</tr>
<tr>
<td>-p</td>
<td>Port on which the Access Server is running.</td>
</tr>
<tr>
<td>-a</td>
<td>The Access Server name you created in Oracle Access Manager.</td>
</tr>
</tbody>
</table>

You may also need to enter an Access Server password (-r AccessServerpassphrase) and an AccessGate password (-P AccessGatePwd) if you created them in Oracle Access Manager.

14 Add JVM Options.

In the GlassFish Server JVM options, add the following entry:

```
<jvm-options>-DJOBACCESS_INSTALL_DIR=<AccessServerSDK-install-dir></jvm-options>
```

You can use the create-jvm-options command to do this. Consider the following example:

```
asadmin --port 4848 --host localhost create-jvm-options
-DJOBACCESS_INSTALL_DIR=D:\"program files\"\NetPoint\AccessServerSDK
```

15 Copy jobaccess.jar.

Copy jobaccess.jar from from installdirectory\AccessServerSDK\oblix\lib into <Glassfish>\lib.

16 Verify the location of the OAM Security Provider.

Ensure that the OAM Security Provider, oam-integration.jar, is in <Glassfish>\lib.
17 Configure the OAM Security Provider.

a. Define your SAM as a message-security-provider.

You can use either the Administration Console or the create-message-security-provider command to do this. Consider the following example:

```xml
<message-security-config auth-layer="HttpServlet">
  <provider-config provider-type="server" provider-id="MySAM"
  class-name="com.sun.glassfish.oamsam.OAMAuthenticatorSAM">
    <property name="oam.resource.hostid.variation" value="your-system.com" />
    <!--property name="form.login.page" value="" /-->
  </provider-config>
</message-security-config>
```

b. Bind the OAM Security Provider.

Bind the OAM Security Provider for use with your application. You do this by defining the httpservlet-security-provider attribute in the glassfish-web.xml file corresponding to your application.

Set the value of the attribute to the name you assigned to the OAM Security Provider in Step a. For example, MySAM.

The group name you specify must match the group name you specified in Oracle Access Manager.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Application Server 9.0 Servlet 2.5//EN" "http://www.sun.com/software/appserver/dtds/sun-web-app_2_5-0.dtd">
<sun-web-app error-url="" httpservlet-security-provider="MySAM">
  <context-root>/BasicAuthen</context-root>
  <security-role-mapping>
    <role-name>glassfish</role-name>
    <group-name>Glassfish_group</group-name>
    <group-name>cn=Glassfish_group,ou=groups,ou=myrealm,dc=oam_domain</group-name>
  </security-role-mapping>
  <class-loader delegate="true"/>
  <jsp-config>
    <property name="keepgenerated" value="true"/>
    <description>Keep a copy of the generated servlet class' java code.</description>
  </jsp-config>
</sun-web-app>
```

18 Deploy the application.

Deploy the application you want to protect in GlassFish Server.

19 Add SDK_install_dir\oblix\lib to the Path.
20 Configure LDAPRealm.
   Configure the LDAPRealm in GlassFish Server to point to Oracle Access Manager backend, as described in "Obtaining Oracle Access Manager Group Information" on page 123. For example, you can use the Administration Console or the `create-auth-realm` command to add the following entry:
   ```
   <auth-realm
      className="com.sun.enterprise.security.auth.realm.ldap.LDAPRealm" name="ldaprealm">
      <property name="jaas-context" value="ldapRealm" />
      <property name="base-dn" value="o=company,c=us" />
      <property name="directory" value="ldap://140.87.134.98:1389" />
      <property name="search-bind-dn" value="cn=Directory Manager" />
      <property name="search-bind-password" value="welcome1" />
   </auth-realm>
   ```
   MaketheLDAPRealmthedefaultrealminGlassFishServer.
   ```
   <security-service default-realm="ldaprealm">
   </security-service>
   ```
   RestartGlassFishServer.
   Restart GlassFish Server and access the protected resource from a browser, such as http://hostname.domainname:8080/BasicAuthen/SecureServlet.

Integrating OAM Security Provider with Oracle Access Manager 11g

This section describes how to integrate the OAM Security Provider with Oracle Access Manager 11g.

▼ Integrating OAM Security Provider with Oracle Access Manager 11g

This procedure explains how to integrate the OAM Security Provider with Oracle Access Manager 11g.


The procedure was tested on Microsoft Windows and the file names are those of the Windows version.

This procedure focuses on LDAP authentication. See “Additional Considerations for Certificate Authentication” on page 162 for information on X509 authentication.

1 Install and configure Oracle Identity and Access Management 11g.
2 Create OAM Server instance.
   If you have not already done so, create an OAM Server instance.
   Under System Configuration, from the left navigation pane select Server Instances and then 
click the Create command button in the tool bar. The Create: OAM Server page appears.
   When you create an OAM Server instance, you specify the transport security mode to use 
between an OAM Agent and this OAM Server instance, from the following choices. See 
“Securing Communication Between OAM 11g Servers and WebGates” in the Access Manager 
online help for a description of these modes and how to use them.
   - Open
   - Simple
   - Cert

   For example, consider the OAM Server instance shown in Figure 7–25.

3 Create a 10g Webgate.
   Under System Configuration, from the left navigation pane select SSO Agents and then OAM 
Agents. Click Create 10g Webgate. The Create OAM 10g Webgate screen appears.
   The security mode you choose must match that of the OAM Server instance.
   The Create OAM 10g Webgate screen is shown in Figure 7–15.
Create one or more Host Identifiers.

You use Host identifiers to simplify the identification of a Web server that hosts resources you want to protect with Access Manager.

From the Policy Configuration tab, expand the Shared Components node, and then select Host Identifiers. Click the Create command button in the tool bar.

Figure 7–16 shows an example host identifier GlassfishTestClient to be used for systems that require BASIC and FORM authentication.
Create a User Identity Store

From the System Configuration tab, select Data Sources. Expand Data Sources and click User Identity Stores. Click the Create command button in the tool bar.

The User Identity Store specifies the LDAP provider it is associated with, as shown in Figure 7–17. This LDAP provider can be the Oracle WebLogic embedded LDAP provider, or another supported provider from the drop-down menu that you have previously configured.
For LDAP providers other than the embedded LDAP provider, when you create a User Identify Store based on one of these LDAP providers, you specify identifying properties, such as the User Search Base and the User Name Attribute that you will need later in this procedure to configure the GlassFish Server LDAP Realm. For example, Figure 7–18 shows the User Identity Store screen for the Oracle Internet Directory LDAP Provider.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>weblogic</td>
<td>User</td>
</tr>
<tr>
<td>Alice</td>
<td>User</td>
</tr>
<tr>
<td>Administrators</td>
<td>Group</td>
</tr>
</tbody>
</table>
For the embedded LDAP provider, the identifying properties are not available from the Access Manager user interface. You therefore need to use some other LDAP tool to configure these properties when you create a user in the next step.

6 Create a group and user in the LDAP provider.

Create a group and user in the LDAP provider who will have access to the protected resource.

Later in this procedure you need to configure the GlassFish Server LDAPRealm, and to do this you will need identifying LDAP properties that are not available for the embedded LDAP provider from the Access Manager user interface.

If you want to use the embedded LDAP provider, you will therefore need to use some other LDAP tool to configure these properties. For example, you might use LDAP commands to add the user and group, and to search the LDAP database for them:

```
ldapadd -x -h cieqlnx01.us.oracle.com -p 7001 -D"cn=Admin" -w welcome1 -f gadd.ldif
ldapadd -x -h cieqlnx01.us.oracle.com -p 7001 -D"cn=Admin" -w welcome1 -f uadd.ldif
ldapsearch -x -h cieqlnx01.us.oracle.com -p 7001 -D"cn=Admin" -w welcome1 -s sub -b "ou=people,ou=myrealm,dc=oam_domain"
ldapsearch -x -h cieqlnx01.us.oracle.com -p 7001 -D"cn=Admin" -w welcome1 -s sub -b "ou=groups,ou=myrealm,dc=oam_domain"
```
In this example, the contents of the "user add" file `uadd.ldif` that adds the user Glassfish might be as follows:

```ldif
# GlassFish, people, myrealm, oam_domain
dn: uid=GlassFish,ou=people,ou=myrealm,dc=oam_domain
objectclass: top
objectclass: person
objectclass: organizationalPerson
    objectclass: inetOrgPerson
objectclass: wlsUser
cn: GlassFish
sn: GlassFish
uid: GlassFish
description: GlassFish User
wlsMemberOf: cn=GlassFish_Group,ou=groups,ou=myrealm,dc=oam_domain
userpassword: GlassFish
```

The contents of the "group add" file `gadd.ldif` that adds a GlassFish_Group might be as follows:

```ldif
dn: cn=GlassFish_Group,ou=groups,ou=myrealm,dc=oam_domain
objectclass: top
objectclass: groupOfURLs
objectclass: groupOfUniqueNames
cn: GlassFish Group
description: OAM GlassFish Group
memberURL: ldap://ou=people,ou=myrealm,dc=oam_domain??
sub?((&objectclass=person)(wlsMemberOf=cn=GlassFish_Group,ou=groups,ou=myrealm,
dc=oam_domain))
```

Make a note of sufficient properties to uniquely identify the user, such as `wlsMemberOf`, as you will need them later in this procedure when you configure the LDAPRealm.

7 Create an Authentication Module.

From the System Configuration tab, select Authentication Modules. Expand Authentication Modules and select LDAP Authentication Modules. Click the Create command button in the tool bar.

Specify a name for this Authentication Module, and select the User Identity Store you previously created, as shown in Figure 7–19.
Create an Authentication Scheme.

Create an authentication scheme for each challenge method (BASIC, Form, and X.509) that you want to use to protect the resource.

From the Policy Configuration tab, expand the Shared Components node, and then select Authentication Schemes. Click the Create command button in the tool bar.

The Authentication Module must already exist.

Figure 7–20 shows an example BASIC over LDAP authentication scheme.
9 Create one or more Application Domains.

From the Policy Configuration tab, select Application Domains. Click the Create command button in the tool bar to create an Application Domain. An example Application Domain is shown in Figure 7–21.

Before an application domain can control access to your resources, you must now identify those resources and specify how you want to protect them:

a. Add a resource to be protected.

From the Policy Configuration tab, expand Application Domains and select the domain you just created. Select Resources, and click the Create command button in the tool bar. On the Create resource page:

- Select HTTP as the resource type.
- Select the host identifier (described in Step 4) for the system that hosts the resource to be protected.
- Specify the resource URL for the Web resource. The URL value of a resource must begin with a forward slash (/), for example /BasicAuthen/SecureServlet, and must match a resource value for the chosen host identifier. To protect all of the resource URLs, append "/*". For example, /BasicAuthen/SecureServlet/*.

Figure 7–22 shows an example of creating a resource that uses a host identifier named GlassfishTestClient.
b. **Add an Authentication Policy.**

From the Policy Configuration tab, expand Application Domains and select the domain you just created. Select Authentication Policies, and click the Create command button in the tool bar. On the Create Authentication Policies page:

- Name this Authentication Policy.
- Select a named authentication scheme you previously configured.
- Select the resources you want to protect with this Authentication Policy.

*Figure 7–23* shows an example of creating an Authentication Policy that uses the BasicScheme authentication scheme.
c. Optionally, create one or more Authorization Policies.

**Note** – The OAM 11g default behavior is to deny access when a resource is not protected by a policy that explicitly allows access. In contrast, the OAM 10g default behavior allowed access when a resource was not protected by a rule or policy that explicitly denied access to limit the number of WebGate queries to the Access Server.

From the Policy Configuration tab, expand Application Domains and select the domain you just created. Select Authorization Policies, and click the Create command button in the tool bar. On the Create Authorization Policies page:

- Name this Authorization Policy.
- Select the resource to be protected by this Authorization Policy.
- Set the Use Implied Restraints control, or explicitly set constraints on the Constraints tab.

*Figure 7–24* shows an example Authorization Policy with two protected resources.
10 **Download and unzip the Oracle Access Server SDK.**

You can download the SDK from Oracle Identity Management 10g (10.1.4.x) Downloads ([http://www.oracle.com/technetwork/middleware/ias/downloads/101401-099957.html](http://www.oracle.com/technetwork/middleware/ias/downloads/101401-099957.html)).

Select Oracle Access Manager Core Components (10.1.4.3.0) (Disk1 ([http://download.oracle.com/otn/nt/middleware/11g/ofm_oam_core_win_10.1.4.3.0_disk1_1of1.zip](http://download.oracle.com/otn/nt/middleware/11g/ofm_oam_core_win_10.1.4.3.0_disk1_1of1.zip))).

The Access Server SDK allows Java, C++, and C applications to use the services of the Oracle Access Manager, including user authentication and authorization of user requests to access resources protected by Oracle Access Manager.

The following components are available from the zip file:

- `oamcfgtool_10_1_4_3_0.zip`
- `Oracle_Access_Manager10_1_4_3_0_Win32-dotnet20_AccessServerSDK.exe`
- `Oracle_Access_Manager10_1_4_3_0_Win32_AccessServerSDK.exe`
- `Oracle_Access_Manager10_1_4_3_0_Win32_Access_Server.exe`
- `Oracle_Access_Manager10_1_4_3_0_Win32_Identity_Server.exe`
- `Oracle_Access_Manager10_1_4_3_0_Win32_OHS11g_Policy_Manager.exe`
- `Oracle_Access_Manager10_1_4_3_0_Win32_OHS11g_WebPass.exe`
- `Oracle_Access_Manager10_1_4_3_0_Win32_Snmp_Agent.exe`

11 **Run the installation wizard.**

`Oracle_Access_Manager10_1_4_3_0_Win32_AccessServerSDK.exe` on the system from which you intend to run the GlassFish Server hosted application and follow the installation wizard.
12 Configure the AccessServer SDK.
Configure the AccessServer SDK to facilitate the handshake between the WebGate you created and Oracle Access Manager.

To do this, run the utility configureAccessGate.exe, located in installdirectory\AccessServerSDK\oblix\tools\configureAccessGate, where installdirectory is the root folder for your Access Manager SDK installation.

configureAccessGate -i <AccessServerSDK-install-dir>  
-t AccessGate -w GlassfishAG -m open -h access-server-host-name -p 5575 -a oam_server1

Please enter the Password for this AccessGate:
Preparing to connect to Access Server. Please wait.
AccessGate installed Successfully.

The command line parameters shown are as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Installation directory for AccessServerSDK.</td>
</tr>
<tr>
<td>-t</td>
<td>AccessGate keyword. Enter as shown.</td>
</tr>
<tr>
<td>-w</td>
<td>Name you specified when you created the WebGate entry in Oracle Access Manager.</td>
</tr>
<tr>
<td>-m</td>
<td>Oracle Access Manager intercomponent mode. This is the security mode you specified between the WebGate/Access client and the OAM Server instance. The valid values are open, simple, or cert.</td>
</tr>
<tr>
<td></td>
<td>Open mode means no encryption. Simple mode means encryption through SSL and a public key certificate provided by Oracle. Cert mode means encryption through SSL and a public key certificate provided by an external CA.</td>
</tr>
<tr>
<td>-h</td>
<td>Fully-qualified name of the host on which you installed the OAM Server instance.</td>
</tr>
<tr>
<td>-p</td>
<td>Port on which the OAM Server instance is running.</td>
</tr>
<tr>
<td>-a</td>
<td>The OAM Server instance name you created in Oracle Access Manager.</td>
</tr>
</tbody>
</table>

You may also need to enter an OAM Server password (-r AccessServerpassphrase) and an AccessGate password (-P AccessGatePwd) if you created them in Oracle Access Manager.

13 Update JVM options
In the GlassFish Server JVM options, add the following entry:

<jvm-options>-DJOBACCESS_INSTALL_DIR=<AccessServerSDK-install-dir></jvm-options>
You can use the create-jvm-options command to do this. Consider the following example:

```bash
asadmin --port 4848 --host localhost create-jvm-options
   -DJOBACCESS_INSTALL_DIR=D:\"program files\\NetPoint\\AccessServerSDK
```

14 **Copy jobaccess.jar.**

Copy jobaccess.jar from from `installdirectory\AccessServerSDK\oblix\lib` into `<Glassfish>\lib`.

15 **Verify the location of the OAM Security Provider.**

Ensure that oam-integration.jar is in `<Glassfish>\lib`.

16 **Configure the OAM Security Provider.**

a. **Define your SAM as a message-security-provider.**

You can use either the Administration Console or the create-message-security-provider command to do this. Consider the following examples:

For BASIC authentication

```xml
<message-security-config auth-layer="HttpServlet">
   <provider-config provider-type="server" provider-id="MySAM"
   class-name="com.sun.glassfish.oamsam.OAMAuthenticatorSAM">
      <property name="oam.resource.hostid.variation" value="your-host-name" />
   </provider-config>
</message-security-config>
```

For Form authentication, you might want to set the `form.login.page`.

```xml
<message-security-config auth-layer="HttpServlet">
   <provider-config provider-type="server" provider-id="MySAM"
   class-name="com.sun.glassfish.oamsam.OAMAuthenticatorSAM">
      <property name="oam.resource.hostid.variation" value="your-host-name" />
      <property name="form.login.page" value="" />
   </provider-config>
</message-security-config>
```

b. **Bind the OAM Security Provider for use with your application.**

You do this by defining the `httpservlet-security-provider` attribute in the `glassfish-web.xml` file corresponding to your application.

Set the value of the attribute to the name you assigned to the OAM Security Provider in Step a. For example, `MySAM`.

The group name you specify must match the group name you specified in the LDAP provider. For example, GlassFish_Group.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app PUBLIC "-//Sun Microsystems, Inc./DTD Application..."/>
```
Deploy the application you want to protect in GlassFish Server.

Add `SDK_install_dir\oblix\lib` to the Path.

Configure the LDAPRealm.

Configure the LDAPRealm in GlassFish Server to point to the Access Manager LDAP provider and group you specified to authenticate the user.

The properties you enter here must match that of the LDAP provider.

For example, you can use the Administration Console or the `create-auth-realm` command to add the following entry:

```xml
<auth-realm classname="com.sun.enterprise.security.auth.realm.ldap.LDAPRealm" name="ldaprealm">
    <property name="jaas-context" value="ldapRealm"/>
    <property name="base-dn" value="ou=people,ou=myrealm,dc=oam_domain"/>
    <property name="directory" value="ldap://cieqalnx01.us.oracle.com:7001"/>
    <property name="group-search-filter" value="cn=%s"/>
    <property name="group-target" value="wlsMemberOf"/>
    <property name="search-bind-dn" value="cn=Admin"/>
    <property name="search-bind-password" value="welcome1"/>
</auth-realm>
```

Make the LDAPRealm the default realm in GlassFish Server.

Restart GlassFish Server.

Restart GlassFish Server and access the protected resource, such as `http://hostname.domainname:8080/BasicAuthen/SecureServlet`. 
Addtional Considerations for Certificate Authentication

In addition to the procedure described in “Integrating OAM Security Provider with Oracle Access Manager 11g” on page 147, there are additional steps required if you want to use certificate authentication to protect your resource.

- The X.509 root certificate and the CA certificate you plan to use for Authentication to the OAM Server must be in the GlassFish Server trust store for the domain.

- The X.509 root certificate and the CA certificate you plan to use for Authentication to the OAM Server must also be in the OAM Server keystore. See Importing CA-Signed Certificates Into the Keystore ([http://download.oracle.com/docs/cd/E21764_01/doc.1111/e15478/keytool.htm#BHBFJIFC](http://download.oracle.com/docs/cd/E21764_01/doc.1111/e15478/keytool.htm#BHBFJIFC)) for instructions on how to do this.

- The X.509 certificate you use for authentication must map to a valid user. To do this, create a user in the Access Manager LDAP provider whose CN attribute matches that of the subject.CN from the certificate.

  Consider the following example, which adds the user Alice. The subject.CN from the certificate must therefore be Alice.

  ```
  #ldapadd -x -h cieqalnx01.us.oracle.com -p 7001 -D "cn=Admin" -w welcome1 -f uadd.ldif
  ```

  In this example, the contents of uadd.ldif are as follows:

  ```
  #contents of uaddalice.ldif
  # GlassFish, people, myrealm, oam_domain
  dn: uid=Alice,ou=people,ou=myrealm,dc=oam_domain
  objectclass: top
  objectclass: person
  objectclass: organizationalPerson
  objectclass: inetOrgPerson
  objectclass: wlsUser
  cn: Alice
  sn: Alice
  uid: Alice
  description: GlassFish End User
  wlsMemberOf: cn=GlassFish_Group,ou=groups,ou=myrealm,dc=oam_domain
  userpassword: Alice
  ```

- Create an X509 Authentication Module. Make sure that the Match LDAP Attribute is set to cn, and that the X509 Cert Attribute is Subject.CN. Set CertValidationEnabled to true.

  Consider the following example.
Create an Authentication Scheme that uses the X509 Challenge Method, and the X509 Authentication Module you just created. Remember that the WebGate challenges the user for credentials based on the type of OAM Server authentication scheme configured for the resource, which is X509 in this case. Set the Challenge URL to the SSL port of OAM server. For example, https://<oam_server>:<ssl_port>/oam/CredCollectServlet/X509.

Note – To work around a known problem, the name of the Authentication Scheme must contain either or both of the (case sensitive) words “X509” and “Certificate”.

Consider the following example.
Configure the OAM Security Provider. To do this, define your SAM as a message-security-provider. You can use either the Administration Console or the create-message-security-provider command to do this. Consider the following example:

For Certificate Auth:

```xml
<message-security-config auth-layer="HttpServlet">
  <provider-config provider-type="server" provider-id="MySAM"
  class-name="com.sun.glassfish.oamsam.OAMAuthenticatorSAM">
    <property description="" name="oam.resource.hostid.variation"
    value="129.158.239.142"></property>
  </provider-config>
</message-security-config>
```

Modify the http-listener-2 configuration in domain.xml as follows:

```xml
<ssl classname="com.sun.enterprise.security.ssl.GlassfishSSLImpl"
ssl3-enabled="false" cert-nickname="slas" client-auth-enabled="true"></ssl>
```

### Integrating OAM Security Provider with Oracle Access Manager 11g and WebGate

This section describes how to integrate the OAM Security Provider with Oracle Access Manager 11g and a WebGate.

For your convenience, this self-contained section includes the steps required to configure Oracle Access Manager for use with the OAM Security Provider and a WebGate. If you have
already followed the steps described in “Integrating OAM Security Provider with Oracle Access Manager 11g” on page 147 to configure Oracle Access Manager, you need only tailor your configuration as described in this section.

In addition to having to configure the Oracle HTTP Server and the Oracle HTTP Server WebGate, the most significant differences in the procedure are the changes to the OAM Security Provider properties.

**Note** – You are not required to install the Oracle Access Server SDK or configure the Access Gate (on the system hosting GlassFish Server) in this use.

### Integrating OAM Security Provider with Oracle Access Manager 11g and WebGate

This procedure explains how to integrate the OAM Security Provider with Oracle Access Manager 11g and a WebGate.


The procedure was tested on Microsoft Windows and the file names are those of the Windows version.

1. **Install and configure Oracle Identity and Access Management 11g.**
2. **Install Oracle HTTP Server.**
   - If you have not already done so, install a version of Oracle HTTP Server (OHS) that is compatible with your Oracle Identity and Access Management installation.
   - OHS is available from Oracle Middleware Downloads page. Look for “Web Tier Utilities” under “Required Additional Software.” Note that you need to install the complete Web Tier Utilities bundle because OHS is not available separately.
3. **Create an OAM Server instance.**
   - If you have not already done so, create an OAM Server instance.
   - Under System Configuration, from the left navigation pane select Server Instances and then click the Create command button in the toolbar. The Create: OAM Server page appears.
   - When you create an OAM Server instance, you specify the transport security mode to use between an OAM Agent and this OAM Server instance, from the following choices. See “Securing Communication Between OAM 11g Servers and WebGates” in the Access Manager online help for a description of these modes and how to use them.
Open
Simple
Cert

For example, consider the OAM Server instance shown in Figure 7–25.

4 Create a User Identity Store

From the System Configuration tab, select Data Sources. Expand Data Sources and click User Identity Stores. Click the Create command button in the tool bar.

The User Identity Store specifies the LDAP provider it is associated with, as shown in Figure 7–26. This LDAP provider can be the Oracle WebLogic embedded LDAP provider, or another supported provider from the drop-down menu that you have previously configured.
For LDAP providers other than the embedded LDAP provider, when you create a User Identify Store based on one of these LDAP providers, you specify identifying properties, such as the User Search Base and the User Name Attribute that you will need later in this procedure to configure the GlassFish Server LDAPRealm. For example, Figure 7–27 shows the User Identity Store screen for the Oracle Internet Directory LDAP Provider.
For the embedded LDAP provider, the identifying properties are not available from the Access Manager user interface. You therefore need to use some other LDAP tool to configure these properties when you create a user in the next step.

5 **Create groups and users in the LDAP provider.**

Create a group and user in the LDAP provider who will have access to the protected resource.

Later in this procedure you need to configure the GlassFish Server LDAPRealm, and to do this you will need identifying LDAP properties that are not available for the embedded LDAP provider from the Access Manager user interface.

If you want to use the embedded LDAP provider, you will therefore need to use some other LDAP tool to configure these properties. For example, you might use LDAP commands to add the user and group, and to search the LDAP database for them:

```
ldapadd -x -h cieqalnx01.us.oracle.com -p 7001 -D "cn=Admin" -w welcome1 -f gadd.ldif
ldapadd -x -h cieqalnx01.us.oracle.com -p 7001 -D "cn=Admin" -w welcome1 -f uadd.ldif
ldapsearch -x -h cieqalnx01.us.oracle.com -p 7001 -D "cn=Admin" -w welcome1 -s sub -b "ou=people,ou=myrealm,dc=oam_domain"
ldapsearch -x -h cieqalnx01.us.oracle.com -p 7001 -D "cn=Admin" -w welcome1 -s sub -b "ou=groups,ou=myrealm,dc=oam_domain"
```
In this example, the contents of the "user add" file `uadd.ldif` that adds the user Glassfish might be as follows:

```
# GlassFish, people, myrealm, oam_domain
dn: uid=GlassFish,ou=people,ou=myrealm,dc=oam_domain
objectclass: top
objectclass: person
objectclass: organizationalPerson
    objectclass: inetOrgPerson
    objectclass: wlsUser
cn: GlassFish
sn: GlassFish
uid: GlassFish
description: GlassFish User
wlsMemberOf: cn=GlassFish_Group,ou=groups,ou=myrealm,dc=oam_domain
userpassword: GlassFish
```

The contents of the "group add" file `gadd.ldif` that adds a GlassFish_Group might be as follows:

```
dn: cn=GlassFish_Group,ou=groups,ou=myrealm,dc=oam_domain
objectclass: top
objectclass: groupOfURLs
objectclass: groupOfUniqueNames
cn: GlassFish Group
description: OAM GlassFish Group
memberURL: ldap:///ou=people,ou=myrealm,dc=oam_domain??
sub?(&(objectclass=person)(wlsMemberOf=cn=GlassFish_Group,ou=groups,ou=myrealm, dc=oam_domain))
```

Make a note of sufficient properties to uniquely identify the user, such as `wlsMemberOf`, as you will need them later in this procedure when you configure the LDAPRealm.

6 **Create an Authentication Module.**

From the System Configuration tab, select Authentication Modules. Expand Authentication Modules and select LDAP Authentication Modules. Click the Create command button in the tool bar.

Specify a name for this Authentication Module, and select the User Identity Store you previously created, as shown in Figure 7–28.
7 **Create an Authentication Scheme.**

Create an authentication scheme for each challenge method (BASIC, Form, and X.509) that you want to use to protect the resource.

From the Policy Configuration tab, expand the Shared Components node, and then select Authentication Schemes. Click the Create command button in the tool bar.

The Authentication Module must already exist.

**Figure 7–29** shows an example BASIC over LDAP authentication scheme.
8  **Create one or more Application Domains.**  

From the Policy Configuration tab, select Application Domains. Click the Create command button in the tool bar to create an Application Domain. An example Application Domain is shown in Figure 7–30.

![Application Domain](image)

Before an application domain can control access to your resources, you must now identify those resources and specify how you want to protect them:

a. **Add a resource to be protected.**  

From the Policy Configuration tab, expand Application Domains and select the domain you just created. Select Resources, and click the Create command button in the tool bar. On the Create resource page:

- Select HTTP as the resource type.
- Select the host identifier (described in Step 5) for the system that hosts the resource to be protected.
- Specify the resource URL for the Web resource. The URL value of a resource must begin with a forward slash (/), for example /BasicAuthen/SecureServlet, and must match a resource value for the chosen host identifier. To protect all of the resource URLs, append "/.../*". For example, /BasicAuthen/SecureServlet/.../*.

*Figure 7–31* shows an example of creating a resource that uses a host identifier named GlassfishTestClient.
b. **Add an Authentication Policy.**

From the Policy Configuration tab, expand Application Domains and select the domain you just created. Select Authentication Policies, and click the Create command button in the tool bar. On the Create Authentication Policies page:

- Name this Authentication Policy.
- Select a named authentication scheme you previously configured.
- Select the resources you want to protect with this Authentication Policy.

*Figure 7–32* shows an example of creating an Authentication Policy that uses the BasicScheme authentication scheme.
c. Optionally, create one or more Authorization Policies.

9 Configure the Oracle HTTP Server WebGate and Register the WebGate Agent.
You configure the Oracle HTTP Server WebGate on the system where the Oracle HTTP Server is installed.

Oracle HTTP Server WebGate is a Web server plug-in that is shipped out-of-the-box with Oracle Access Manager. The Oracle HTTP Server WebGate intercepts HTTP requests from users for Web resources and forwards them to the Access Server for authentication and authorization. Oracle HTTP Server WebGate installation packages are found on media and virtual media that are separate from the core components.

For complete instructions, see Installing and Configuring Oracle HTTP Server 11g Webgate for OAM (http://download.oracle.com/docs/cd/E21764_01/install.1111/e12002/webgate.htm#CACEAEIE).

10 Configure Oracle HTTP Server and Restart
Edit `mod_wl_ohs.conf` to configure the redirect host and port (WebLogicHost and WebLogicPort, respectively) to point to the GlassFish Server instance in which the secured application will run. Consider the following example:

```xml
<Location /BasicAuthen/SecureServlet>
SetHandler weblogic-handler
WebLogicHost host-name-where-GlassFish-running
WebLogicPort 8080
DebugConfigInfo on
</Location>
```
After you have edited the file, restart Oracle HTTP server.

11 Verify the location of the OAM Security Provider.
Ensure that oam-integration.jar is in <Glassfish>/lib.

12 Configure the OAM Security Provider.

a. Define your SAM as a message-security-provider.
You can use either the Administration Console or the create-message-security-provider(1) subcommand to do this. Consider the following example:

```xml
<message-security-config auth-layer="HttpServlet">
  <provider-config provider-type="server" provider-id="MySAM"
    class-name="com.sun.glassfish.oamsam.OAMAuthenticatorSAM">
    <property name="oam.base dn" value="ou=people,ou=myrealm,dc=oam_domain"></property>
    <property name="oam.ext.header" value="OAM_REMOTE_USER"></property>
    <property name="IdentityAsserterMode" value="true"></property>
  </provider-config>
</message-security-config>
```

The OAM Security Provider acts as an Identity Asserter only when IdentityAsserterMode is set to true. This example also shows the (OAM_REMOTE_USER) header. The value of the base-dn must point to the user's base-dn you configured in Step 5.

b. Bind the OAM Security Provider for use with your application.
You do this by defining the httpservlet-security-provider attribute in the glassfish-web.xml file corresponding to your application.

Set the value of the attribute to the name you assigned to the OAM Security Provider in Step a. For example, MySAM.

The group name you specify must match the group name you specified in the LDAP provider. For example, GlassFish_Group.

```xml
<security-role-mapping>
  <role-name>glassfish</role-name>
  <group-name>GlassFish_Group</group-name>
  <group-name>cn=GlassFish_Group,ou=groups,ou=myrealm,dc=oam_domain</group-name>
</security-role-mapping>
```

Integrating OAM Security Provider with Oracle Access Manager 11g and WebGate
Oracle GlassFish Server 3.1 Security Guide • August 2011

174
Deploy the application you want to protect in GlassFish Server.

Configure the LDAP Realm.
Configure the LDAP Realm in GlassFish Server to point to the Access Manager LDAP provider and group you specified to authenticate the user.

GlassFish Server directly contacts this LDAP provider for group information.

The properties you enter here must match that of the LDAP provider.

For example, you can use the Administration Console or the `create-auth-realm` command to add the following entry:

```xml
<auth-realm class="com.sun.enterprise.security.auth.realm.ldap.LDAPRealm" name="ldaprealm">
  <property name="jaas-context" value="ldapRealm"></property>
  <property name="base-dn" value="/ou=people,ou=myrealm,dc=oam_domain"></property>
  <property name="directory" value="ldap://cieqalnx01.us.oracle.com:7001"></property>
  <property name="group-search-filter" value="cn=%s"></property>
  <property name="group-target" value="wlsMemberOf"></property>
  <property name="search-bind-dn" value="cn=Admin"></property>
  <property name="search-bind-password" value="welcome1"></property>
</auth-realm>
```

Make the LDAP Realm the default realm in GlassFish Server.

Restart GlassFish Server.

Restart GlassFish Server and access the protected resource, such as `http://hostname.domainname:8080/BasicAuthen/SecureServlet`.

**Additional Considerations for Certificate Authentication With a WebGate**

In addition to the procedure described in “Integrating OAM Security Provider with Oracle Access Manager 11g and WebGate” on page 165, there are additional steps required if you want to use certificate authentication to protect your resource.

As described in “Use Case: Identity Assertion for Web Resources via WebGate” on page 127, the WebGate challenges the user for credentials based on the type of Oracle Access Manager authentication scheme configured for the resource, which is client-cert in this use case. In response, the client application needs to send its own certificate.
The underlying WebLogic Server implementation is the Web container and must trust the application’s certificate. WebLogic Server can trust this certificate either because the certificate is explicitly in the truststore, or because the certificate has a valid cert path to a trusted certificate.

You must therefore perform the following actions or satisfy their requirements by some other method:

1. Select the certificate you want the client to use. If you do not already have a private certificate, you can use a tool such as OpenSSL to generate a self-signed one for development and testing purposes.

2. Import your certificate into your browser. For Firefox, choose the ‘Ask me every time’ option in the browser (for choosing a Personal Certificate). Also make sure that SSL 3.0 is checked in the browser. Choose analogous options for other browser types.

3. The X.509 certificate you use for authentication must map to a valid user. To do this, create a user in the Access Manager LDAP provider whose CN attribute matches that of the subject.CN from the certificate.

   Consider the following example, which adds the user Alice. The subject.CN from the certificate must therefore be Alice.

   ```
   #ldapadd -x -h cieqalnx01.us.oracle.com -p 7001
   -D "cn=Admin" -w welcome1 -f uadd.ldif
   
   In this example, the contents of uadd.ldif are as follows:
   
   #contents of uaddalice.ldif
   # GlassFish, people, myrealm, oam_domain
   dn: uid=Alice,ou=people,ou=myrealm,dc=oam_domain
   objectclass: top
   objectclass: person
   objectclass: organizationalPerson
   objectclass: inetOrgPerson
   objectclass: wlsUser
   cn: Alice
   sn: Alice
   uid: Alice
   description: GlassFish End User
   wlsMemberOf: cn=GlassFish_Group,ou=groups,ou=myrealm,dc=oam_domain
   userpassword: Alice
   
   4. Create an X509 Authentication Module. Make sure that the Match LDAP Attribute is set to cn, and that the X509 Cert Attribute is Subject.CN. Set CertValidationEnabled to true. Consider the following example.
5. Create an Authentication Scheme that uses the X509 Challenge Method, and the X509 Authentication Module you just created.

Remember that the WebGate challenges the user for credentials based on the type of OAM Server authentication scheme configured for the resource.

Set the Challenge URL to the SSL port of OAM server. For example, https://<oam_server>:<ssl_port>/oam/CredCollectServlet/X509.

**Note** – To work around a known problem, the name of the Authentication Scheme must contain either or both of the (case sensitive) words “X509” and “Certificate”.

Consider the following example.
6. Configure WebLogic Server for Custom Identity and Custom Trust, as described in Configure Identity and Trust.

7. Import the root and intermediate certificates for your generated certificate into a custom keystore and truststore.

   ```
   keytool -import -keystore /scratch/mytruststore -storepass (...) -file root.cer -alias alice_root
   keytool -import -keystore /scratch/mykeystore -storepass (...) -file root.cer -alias alice_root
   keytool -import -keystore /scratch/mytruststore -storepass (...) -file ca.cer -alias alice_root_1
   keytool -import -keystore /scratch/aime1/mykeystore -storepass (...) -file ca.cer -alias alice_root_1
   ```

8. Copy the custom keystore and truststore to the WebLogic Server system.

9. Configure WebLogic Server to use this custom keystore and truststore, as described in Configure Identity and Trust.

10. Restart the Oracle Access Manager and the Oracle HTTP Server.

### Session Synchronization

When the same application is used by different user identities, GlassFish Server ensures that the same session is not used by the new user.
Specifically, when a user logs in using the BASIC, FORM or CLIENT-CERT authentication schemes after authentication from Oracle Access Manager, both an Oracle Access Manager session and a GlassFish Server HTTP session are established. When the Oracle Access Manager session is then cleared, the GlassFish Server HTTP session is not automatically cleared.

Therefore, after clearing the Oracle Access Manager session:
- If a user logs in through the OAM Security Provider, and
- The user principal differs from the principal in the previous HTTP session, then
- The previous HTTP session is cleared and a new session is created.
Index

A
additional information, on Message Security,  84
admin password,  21
  resetting,  40–41
administration realm,  56
Administrative Security, Managing,  89–105
aliases
  creating for passwords,  42–43
  deleting for a password,  43–44
  for passwords,  22,  41–45
  listing for passwords,  43
application security, overview,  72
asadmin, configure-ldap-for-admin,  63–64
audit modules,  28
  creating,  45
  deleting,  46
  listing,  45–46
authentication
  methods,  18–23
  overview,  18–23
  overview of types,  18
  realms,  55–64
  single sign-on,  22–23
authorization
  JACC providers,  24
  overview,  23–27

C
certificate files, overview,  30
certificate realm,  56
certificates
  administering with keytool,  47–52
  deleting with keytool,  52
  generating with keytool,  47–49
  overview,  29
  signing with keytool,  49–51
certificates, SSL,  61–62,  62–63,  63–64
change-admin-password command,  40–41
change-master-password subcommand,  36
changing
  admin password,  40–41
  master password,  36
clear text,  59–61
Cluster Mode, Administering Security,  85–87
configure-ldap-for-admin,  63–64
configuring, message protection policies,  76–79
create-audit-module subcommand,  45
create-auth-realm command,  57,  59–61
create-file-user command,  65
create-message-security-provider command,  80,  82–83
create-password-alias command,  42–43
creating
  a custom realm,  57
  audit modules,  45
  message security provider,  80,  82–83
  password alias,  42–43
  realms,  57
  users,  65
custom realm, creating,  57
Index

D
DAS, LDAP authentication, 63–64
delete-audit-module subcommand, 46
delete-auth-realm command, 59
delete-file-user command, 67–68
delete-message-security-provider command, 81–82
delete-password-alias command, 43–44
deleting
  audit modules, 46
  message security provider, 81–82
  password alias, 43–44
  realms, 59
  users, 67–68
digest realm, 56
  configuring, 59–61

E
enabling
  default client provider for messaging, 76
  default message security provider, 75
  encrypting a password, 42

F
file for passwords, 41
file groups, listing, 66
file realm, 56
file users
  creating, 65
  deleting, 67–68
  listing, 65–66
  listing groups, 66
  updating, 67
firewall guidelines, 28

G
generating certificates, with keytool, 47–49

J
JACC, overview, 24
JACC providers, administering, 52–54
JDBC, configuring realm, 59–61
JDBC realm, 56
JSSE security
  administering certificates, 47–52
  deleting a certificate, 52
  generating a certificate, 47–49
  signing a certificate, 49–51

K
keystore file, overview, 30
keytool, 61–62, 62–63, 63–64
keytool utility
  deleting a certificate, 52
  generating a certificate, 47–49
  signing a certificate, 49–51

L
LDAP
  DAS, 63–64
  OID, 61–62
  OID/OVD, 63–64
  OVD, 62–63
LDAP realm, OVD/OID, 56
list-audit-modules command, 45–46
list-auth-realm command, 58
list-file-groups command, 66
list-file-users command, 65–66
list-message-security-providers command, 81
list-password-aliases command, 43
listing
  audit modules, 45–46
  file groups, 66
  message security provider, 81
  password aliases, 43
  realms, 58
  users, 65–66
M
master password, 19
  changing, 36–37
  saving to file, 21
  understanding synchronization, 20
  use with start-cluster, 37–38
  use with start-instance, 37–38
  using default, 20
  using when creating domain, 21
message protection policies, 72
  configuring, 76–79
message security, 69–84
  overview, 69–75
  roles, 74
message security providers
  administering, 80–83
  creating, 80, 82–83
  deleting, 81–82
  listing, 81
  updating, 81

O
OAM security provider
  authentication methods, 132–133
  configuring, 130–132
  integrating with Oracle Access Manager
    10g, 133–147
    11g, 147–161, 165–175
  understanding use cases, 125–130
OID, LDAP, 61–62
OID/OVD, LDAP, 63–64
Oracle Access Manager
  about, 123–124
  about OAM security provider, 122–123
  integrating, 121–179
Oracle Solaris realm, 56
OVD, LDAP, 62–63
OVD/OID, LDAP realm, 56
overview
  certificates and SSL, 28–35
  message security, 69–75
  passwords, 19–22
overview (Continued)
  realms, 55–64
  roles, 23
  system security, 17–35
  web services security, 70

P
PAM realm, 56
passwordfile option, 41
passwords
  admin, 21
  administering, 36–45
  aliases, 22, 41–45
  changing admin password, 40–41
  changing master, 36
  encoded, 22
  encrypting, 42
  master, 19
  overview, 19–22
  setting from a file, 41

R
realms
  certificate, 29
  configuring digest realm, 59–61
  configuring JDBC, 59–61
  creating, 57
  deleting, 59
  listing, 58
  overview, 55–64
  updating, 58
  roles, overview, 23

S
sample application, web services, 75
Secure Environment, Running In, 107–120
security
  administering, 17–54
  JSSE, 47–52
Index

security (Continued)
  managing for users, 64–68
  message, 69–84
  overview, 17–35
  tools for managing, 35
server authentication modules, JSR 196, 18–19
server policy file
  changing default permissions, 27
  working with, 24–27
set command, for updating an authentication
  realm, 58
single sign-on, 22–23
SOAP, 70
SSL, 61–62, 62–63, 63–64
  overview, 30–32
SSL mutual authentication, custom authentication of
  client certificate, 32–35

T
  tools, for managing system security, 35
  truststore file, overview, 30

U
  update-file-user command, 67
  update-message-security-provider command, 81
  update-password-alias command, 44–45
  updating
    message security provider, 81
    password alias, 44–45
    realms, 58
    users, 67
user security
  administering, 55–68
  creating users, 65
  deleting users, 67–68
  listing file groups, 66
  listing users, 65–66
  managing, 64–68
  updating users, 67

V
  viewing
    audit modules, 45–46
    authentication realms, 58
    file users, 65–66

W
  Wallet Manager, 61–62, 62–63, 63–64
  web services
    message security, 69–84
    sample application, 75
  WSIT, 70