BEA AquaLogic Enterprise Security™®

Administration Reference

Version 2.6
Document Revised: April 2007

1. Introduction and Roadmap

2. Administrative Utilities

3. WLESblm.conf Reference

4. Provider Extensions

5. Audit Events

6. BLM Configuration API Security Providers Reference

CHAPTER 1

Introduction and Roadmap

The following sections describe the content and organization of this document:

Scope

This document describe how to write custom security provider extensions, describes how to extend the AuditContext interface, and describes how to use the Custom Extensions API.

Documentation Audience

This document distinguishes between two levels of security administrators.

- Application Security Administrators — these administrators are responsible for integrating ALES into application environments, managing interaction between an applications and ProductNameShort, and setting up application-level security administrators.

- Typical tasks include modifying deployment descriptors, managing security providers and other security configurations, managing single sign-on scripts, setting up application-level security administrators.

  Application-level Security Administrators — these administrators are responsible for securing applications using ALES policies.

  The primary task is to create and deploy the policies securing application resources.

Prerequisites for this Document

Prior to reading this guide, you should read the *Introduction to BEA AquaLogic Enterprise Security*. This document describes how the product works and provides conceptual information that is helpful to understanding the necessary installation components.

Additionally, BEA AquaLogic Enterprise Security includes many unique terms and concepts that you need to understand. These terms and concepts—which you will encounter throughout the documentation—are defined in the *Glossary*.

Guide to this Document

This document is organized as follows:

- Chapter 2, "Administrative Utilities," provides a reference to various command-line administrative utilities provided by AquaLogic Enterprise Security.
- Chapter 3, "WLESblm.conf Reference," describes the configuration parameters in the WLESblm.conf configuration file. You can edit this file to configure and tune AquaLogic Enterprise Security after installation.
- Chapter 4, "Provider Extensions," describes how to write custom security provider extensions. A provider extension is a plug-in function that you write to extend the capabilities of the existing providers. While the security providers supplied with AquaLogic Enterprise Security are configurable, the plug-ins enable you to customize them to add additional functionality.
- Chapter 5, "Audit Events," describes how to extend the AuditContext interface. The `AuditEvent` interface provides a mechanism for passing additional audit information to Auditing providers during a `writeEvent` operation. If you implement this interface and you expect to receive a ContextHandler argument from a caller, you can extend the `AuditContext` interface to provide more information.

- Chapter 5, "Policy Language Custom Extension Library API Reference," describes how to use the Custom Extensions API. The Custom Extensions API provides a policy language for writing custom extension libraries (plug-ins) to enhance features available through the ASIAuthorizer, such as routines for dynamic computation of an attribute value (credential function) or custom predicate (evaluation function).
- Chapter 6, "BLM Configuration API Security Providers Reference," describes the security provider attributes, their default values, and indicates whether the `getValue/setValue` and the `getValue/setValueList` methods can be used with the attributes. This information is needed if you want to use the BLM API to configure security providers.

Related Information

The BEA corporate web site provides all documentation for BEA AquaLogic Enterprise Security. Other BEA AquaLogic Enterprise Security documents that may be of interest to the reader include:

- *WSDL Documentation for the Web Service Interfaces*—This document provides reference documentation for the Web Services Interfaces that are provided with and supported by this release of BEA AquaLogic Enterprise Security.
- *Administration and Deployment Guide*—This document provides step-by-step instructions for performing various administrative tasks.
- *Integrating ALES with Application Environments*—This document describes important tasks associated with integrating AquaLogic Enterprise Security into application environments.
- *Policy Managers Guide*—This document how to write access control policies for BEA AquaLogic Enterprise Security, and describes how to import and export policy data.
- *Installing Security Services Modules*—This document describes how to install ALES Security Services Modules, including the Web Services Security Service Module.
- *Developing Security Providers* —This document provides security vendors and security and application developers with the information needed to develop custom security providers.
- *Javadocs for Security Service Provider Interfaces*—This document provides reference documentation for the Security Service Provider Interfaces that are provided with and supported by this release of BEA AquaLogic Enterprise Security.

- *Programming Security for Java Applications*—The document describes how to implement security in Java applications. It include descriptions of the Security Service Application Programming Interfaces and programming instructions for implementing security in Java applications.
- *Javadocs for Java API*—This document provides reference documentation for the Java Application Programming Interfaces that are provided with and supported by this release of BEA AquaLogic Enterprise Security.

CHAPTER 2

Administrative Utilities

AquaLogic Enterprise Security includes a number of helpful administrative utilities. This section provides a reference to the following utilities:

Note: "Configuration File Usage" on page 2-17 describes which configuration files are used by a particular utility.

In the syntax descriptions for these utilities:

- ALES_ADMIN_HOME is the directory in which you have installed the AquaLogic Enterprise Security Administration Server; by default, this is BEA_HOME/ales26-admin.
- angle brackets <> indicate required variable arguments
- square brackets [] indicate optional variable arguments

policyloader

This is the Policy Import tool, which you can use to import your policy files. Normally all the tool needs is a path to a valid policy loader configuration file. All the settings are listed in that file. You can use additional command line arguments to override the settings listed in the configuration file.

If you import a file that uses multi-byte characters, the file must be UTF-8 encoded.

As of AquaLogic Enterprise Security version 2.5, policy loading is now transactional: all policies are loaded, or none. In addition, the BLMContextManager API has been updated to include transactional methods.

For information about creating a policy loader configuration file, see Sample Configuration File in the *Policy Managers Guide*. For more information about running the Policy Import tool, see Running the Policy Import Tool and Understanding How the Policy Loader Works in the *Policy Managers Guide*.

Usage

```
ALES_ADMIN_HOME\bin\policyloader.bat <configuration_file>
[-initial|-recover] [-load|-remove] [-help|-?|-usage]
```

```
ALES_ADMIN_HOME/bin/policyloader.sh <configuration_file>
[-initial|-recover] [-load|-remove] [-help|-?|-usage]
```

Options

The following options are supported:

-help|-?|-usage
Print USAGE and exit.

-initial
Run in initial mode. There should be no versioned files in the policy directory in this mode.

-recover

Run in recover mode to revert to an earlier policy set. There should be checkpoint files (generated automatically during a previous load) in the policy directory in this mode.

-load

Run in policy load mode (default). Load policy from the files specified in the configuration file.

-remove

Run in policy remove mode. Remove the policies described in the files specified in the configuration file

Example

```
>policyloader.bat MyAppPolicy.conf
```

load_adminpolicy

Loads the admin policy. This tool does not take any arguments. It needs to be run only once per Administration Server installation. It needs to run after the database schema has been loaded. Once this is tool is run, it will set the correct policy that will allow the `system` user to access the Administration Console.

Usage

```
ALES_ADMIN_HOME\bin\load_adminpolicy.bat
```

Example

```
>load_adminpolicy.bat
```

policyIX

The Policy Propagation Import/Export tool. You can use this tool to propagate your policy from one environment to another, and to export SSM configuration data for use when an SCM is not associated with the SSM. An example would be moving policy from a development installation to a QA installation, or from a staging installation to a production deployment. You can also use policyIX to import and export policy data between ALES and AquaLogic Enterprise Repository.

If you import a file that uses multi-byte characters, the file must be UTF-8 encoded.

Exporting Policy

To use the policyIX tool to export policy, pass it an XML configuration file that basically specifies the top level resource node you want to export. The tool determines all the related policy elements that are related to that resource and its leaf nodes. When you import the exported file in another environment, the policyIX tool creates a replica of the original resource tree with accompanying policy.

Exporting Configuration Data

The PolicyIX tool allows you to export configuration data (configured either through the ALES Administration Console, or directly via the BLM API) for a given SSM to an XML file, and use it with the configured SSMs when the SCM is not available.

To use the tool to export SSM configuration data, pass it the SSM configuration ID to export, the `exportConfig` parameter, the `config.xml` file and, optionally, the name of the exported XML file.

PolicyIX uses the existing settings for the SSL infrastructure, specified during the Administration server installation, to sign the exported configuration files. Specifically, the `PolicyIX.bat` file invokes the tool with `-Dales.policyTool.signer=wles-admin`. The `ales.policyTool.signer` property is a required Java property that specifies the alias of the signing key in the identity keystore, which must be equal to the Administration server machine name.

The public key of the Administration server is then retrieved from the SSL peer keystore for the purpose of validating the configuration file's signature. This public key is available from the Administration server's certificate, which was added to the SSL peer keystore during the enrollment process.

The unencoded signature of the XML file is stored in a corresponding signature file, whose name is derived from the full name of the signed XML file (including extension) with the added `.sig` extension. For example, `myconfig.xml.sig`.

After you export the configuration data, you must manually copy the XML configuration file and signature file to the SSM configuration directory, *`BEA_HOME`*`/ales26-ssm/<`*`ssm-type`*`>/`*`instance-name`*`/config.`

If you do not use the default name (`wles.securityrealm.xml`) for this configuration file, set the `wles.realm.filename` property in the *`BEA_HOME`*`/ales26-ssm/<`*`ssm-type`*`>/`*`instance-name`*`/config/security.properties` file.

See Installing an SSM Without an Associated SCM in Installing Security Service Modules for additional information about the `security.properties` file.

Usage

```
ALES_ADMIN_HOME\bin\policyIX.bat <-import|-export> <config.xml>
<policy.xml> [-passwdPrompt]

ALES_ADMIN_HOME\bin\policyIX.bat <exportID> <-exportConfig> <config.xml>
[exportName] [-passwdPrompt]

ALES_ADMIN_HOME/bin/policyIX.sh <-import|-export> <config.xml>
<policy.xml> [-passwdPrompt]

ALES_ADMIN_HOME/bin/policyIX.sh <exportID> <-exportConfig> <config.xml>
[exportName][-passwdPrompt]
```

Options

-import
Run the tool in policy import mode.

-export
Run the tool in policy export mode.

exportID
Command line parameter that specifies the SSM configuration ID to export. This entry must match the SSM configuration ID that is specified when the SSM instance was created on the server machine. The configuration ID is the means by which the SSM receives it configuration. If -exportConfig is specified, the exportID is required and must be in the first position.

-exportConfig
Command line parameter that instructs PolicyIX to export the SSM configuration. If -exportConfig is specified it must be in the second position.

exportName
Command line parameter that specifies the name of the exported XML file. If it is not provided, `wles.securityrealm.xml` is used by default. If -exportConfig is specified exportName is optional, but must be in the forth position if present.

The default name for this configuration file is `wles.securityrealm.xml`. If you do not use the default name, set the `wles.realm.filename` property in the `security.properties` file.

config.xml

This configuration file contains BLM configuration and import or export configuration detail. If you run policyIX in import mode, then the configuration file may also contain policy data to be imported. A sample policyIX configuration file can be found at `ALES_ADMIN_HOME/config/policyIX_config.xml`. See Table 2-1 and the comments in the sample `policyIX_config.xml` file for information about the values to include in your configuration file.

Table 2-1 Configuration File Elements

Element	Description	Children or Attribute Examples
policy_propagation	The parent or container element.	`xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://policypropagation.ales.com/xmlbean" targetNamespace="http://policypropagation.ales.com/xmlbean">`
configuration	Container element.	Contains either one `export_configuration` or one `import_configuration` element, plus one `blm_configuration` element and optionally one `aler_configuration` element.
export_configuration	Used if `-export` switch is used for the policyIX tool	Contains one `clipping_resource` element.
clipping_resource	The clipping Resource node. All related policy elements will be exported.	The value attribute specifies the Resource node. For example: `<clipping_resource value="//app/policy"/>`
import_configuration	Used if `-import` switch is used for the policyIX tool	Contains one `policy_load_procedure` element.

Table 2-1 Configuration File Elements

Element	Description	Children or Attribute Examples
policy_load_procedure	Specifies how to handle existing policies.	Possible values: • override - Add policy to already existing policy • delete_existing - Delete the policy being imported from destination before importing new policy
blm_configuration	Container for elements that specify how to connect to ALES.	Contains multiple `blm_property` elements.
blm_property	Name/value pairs that specify how to connect to ALES.	Possible property names and values are: • server_ip - Machine name or IP address of server running BLM • server_port - Port of the BLM server. Default is normally the Admin Console SSL port +1. • userID - ALES Admin username. Default is `system` • userPassword - Can also be provided at the command prompt by using the `-passwdPrompt` option. Default is `weblogic`. • print_info - If set to `true`, then `BLMAlreadyExists` exceptions and exceptions related to removing Rules will be sent to standard console output.

Table 2-1 Configuration File Elements

Element	Description	Children or Attribute Examples
aler_configuration	Container for elements that specify how to connect to AquaLogic Enterprise Repository (ALER). Used only with `-exportToALER` or `-importFromALER` options.	Contains multiple `aler_property` elements.
aler_property	Name/value pairs that specify how to connect to ALER.	Possible property names and values are: • server_url - ALER connection URL • username - user name to use to connect to ALER • userPassword - user password to connect to ALER • assetDescription - A description of the asset, only used when the asset is submitted • assetName - name of the asset to export or import • importAssetVersion - Asset version to import, only valid if the `-importFromALER` policyIX option is used.

policy.xml

If you run policyIX in export mode, then policy data will be exported into this file. If you run policyIX in import mode and the XML configuration file does not contain policy data, then this file will contain policy configuration and data to be imported.

-passwdPrompt

If you use this option, the admin password will be read from command line.

exportToALER

Export data directly from ALES to ALER based on configuration parameters in the `config.xml` file. To export data to ALER from a policy file, specify the pathname of the file. If a policy file is specified, no connection is made to ALES.

importFromALER

Import policy data directly from ALER to ALES based on configuration parameters in the `config.xml` file. To import data from ALER to a policy file, specify the pathname of the file. If a policy file is specified, no connection is made to ALES.

Examples

To export a policy to a file:

```
>policyIX.bat –export MyServer1ExportConfig.xml MyPolicy.xml
```

To export an SSM configuration:

```
>policyIX.bat exportID -exportConfig MyServer1ExportConfig.xml MySSM.xml
```

To import a policy from a file:

```
>policyIX.bat –import MyServer2ImportConfig.xml MyPolicy.xml
```

To export a policy node to AquaLogic Enterprise Repository:

```
>policyIX.bat -exportToALER config.xml
```

To import policy data from AquaLogic Enterprise Repository:

```
>policyIX -importFromALER config.xml
```

policyexporter

Export ALES policy data from a database server to a directory in policyloader format. The tool requires an empty directory into which it will export the files and that directory must exist before running the tool. Any existing policy files in that directory will replaced or deleted. On UNIX, the program will prompt for each input, and then user can input the arguments. Make sure the current working directory is `ALES_ADMIN_HOME/bin` before running the tool.

Usage

```
ALES_ADMIN_HOME\bin\policyexporter.bat [directory]
```

```
ALES_ADMIN_HOME/bin/policyexporter.sh
```

Options

directory
Directory path to which the files will be exported. Use to export to the current directory.

Example

```
>policyexporter.bat c:\MyPolicy
```

install_ales_schema

Installs the ALES policy database schema into the database server. If the schema already exists, it will be replaced, including existing policy. On UNIX, the program prompts you to input the arguments. Make sure the current working directory is `ALES_ADMIN_HOME/bin` before running the tool.

Usage

```
ALES_ADMIN_HOME\bin\install_ales_schema.bat <db-username> <db-password>
```

```
ALES_ADMIN_HOME/bin/install_ales_schema.sh
```

Options

db-username
Login ID, usually same as owner

db-password
Password for the db-username

Example

```
>install_ales_schema.bat username password
```

asipassword

A secure password utility tool. Encrypts the password with the key and saves it using based64 encoding into the password file with corresponding alias. You can use this tool to store or update the password for the `system` user or the database user. The ASIAuthorizer and BLM both look into the `password.xml` for the correct password to connect to the ALES database.

Usage

```
ALES_ADMIN_HOME\bin\asipassword.bat <alias> [passwordFilename]
[keyFilename]
```

```
ALES_ADMIN_HOME/bin/asipassword.sh <alias> [passwordFilename]
[keyFilename]
```

Options

alias

The alias for the password, often the username.

passwordFileName

The filename for the xml password file. The default, `ssl/password.xml`, is used if you do not supply a different value for this option.

keyFileName

The filename for the password key file. The default, `ssl/password.key`, is used if you do not supply a different value for this option.

Example

```
cd ssl
../bin/asipassword.bat wles
```

asisignal

Sends an action command to the server via a Web Service interface.

Usage

```
ALES_ADMIN_HOME\bin\asisignal.bat -url server_url [-action
ping|comtest|wait|waitready|status] [-msg msg_to_log] [-reps 1] [-interval
1000] [-?] [-dbg]
```

```
ALES_ADMIN_HOME/bin/asisignal.sh  -url server_url [-action
ping|comtest|wait|waitready|status] [-msg msg_to_log] [-reps 1] [-interval
1000] [-?] [-dbg]
```

Options

-action ping, comtest
Send a simple SOAP call to the server, and see if server returns a valid SOAP result.

-action status
Get the server status. Could be INITING or READY.

-action wait
Continuously ping the server until the server replies. If you use this option together with the `-reps` option, sends ping until the server replies or the number of pings specified by the `-reps` option has been sent.

-action waitready
Like `wait`, but waits for the server to reach READY status, not just to respond to the SOAP communication.

-url
The Managed Server SOAP service URL (endpoint), usually ends with `/ManagedServer`. For example, `https://host:7011/ManagedServer`.

-msg
The message used by the log action to send to the server.

-reps
Repeat count. Used with the `-wait` and `-waitready` actions.

-interval
Sleep interval between each action, in milliseconds. Default is 1000 msecs (1s).

-?
Print a help message.

-dbg
Turn on debug for this utility.

Example

Ping the BLM Server running on the default port:

```
>asisignal.bat -action ping -url https://host:7011/ManagedServer
```

policy2XACML

A utility to translate policy rules from the ALES ASIAuthorizer format to XACML. It reads ALES policies from an input file in policyloader format, translates ALES rules to XACML, and stores the XACML rules to an output file.

Usage

```
ALES_ADMIN_HOME\bin\policy2XACML.bat [-in filename] [-out filename] [-?]
```

```
ALES_ADMIN_HOME/bin/policy2XACML.sh [-in filename] [-out filename] [-?]
```

Options

-in

The input policy file name. If no input file is provided, read standard input, until EOF is detected.

-out

The output policy file name. If no output file is provided, print to standard output.

Example

```
>policy2XACML.bat –in rule –out rule.xacml
```

enrolltool

Enrolls an SCM instance by acquiring security certificates from the associated ALES Administration Server. The enrollment is required to configure one-way or two-ways SSL communication (see Configuring SSL for Production Environments in the *Administration and Deployment Guide* for more information). Before enrolling an SCM instance, make sure that the ALES Administration Server is running.

Usage

```
ALES_SCM_HOME\bin\enrolltool.bat <demo|secure>
```

```
ALES_SCM_HOME/bin/enrolltool.sh <demo|secure>
```

Options

demo

Enrolls the SCM instance and verifies the Administration Server certificate using the demo CA certificate from the `DemoTrust.jks` key store in directory `ALES_SCM_HOME/ssl`. If this option is specified, the tool does not verify matching of the Administration Server host with the one from the certificate. This option should never be used in a production environment.

secure

Enrolls the SCM instance and verifies the Administration Server certificate using a CA certificate from the `trust.jks` key store in directory `ALES_SCM_HOME/ssl`. If this option is specified, the tool verifies matching of the Administration Server host with the one from the certificate.

Menu Options

When the tool is started, it displays the following menu options.

1. Show Enrolled Domains
2. Show Un-enrolled Domains
3. Register Domain
4. Unregister Domain
5. Enroll
6. Un-enroll
7. Exit

Below you will find the explanations for each option.

1. Show Enrolled Domains shows the list of all enrolled security domains including the following information for each of the domains:
   - URLs of primary and secondary policy distributors (BLM),
   - public and private ports of the SCM instance, and
   - the name of the SCM instance.
2. Show Un-enrolled Domains shows the list of all un-enrolled domains including the following information for each of the domains:

  - URLs of primary and secondary policy distributors (BLM),
  - public and private ports of the SCM instance, and
  - the name of the SCM instance.

3. Register Domain registers a new enterprise security domain. You must enter the following data about the domain:
   - the domain name,
   - the URLs of the primary and secondary Administration Severs,
   - listening port number and
   - name of the SCM instance.

   The new data is stored in the `ALES_SCM_HOME\config\SCM.properties` file. Initially, the new domain is un-enrolled. You must enroll it by selecting Option 1 of the menu.
4. Unregister Domain unregisters an enterprise security domain. The domain must be un-enrolled before it can be unregistered. You can un-enroll a domain by selecting Option 6 of the menu.
5. Enroll enrolls the SCM instance associated with the chosen security domain. You will be asked for the administrator's username and password to access the administration server. If the SCM is enrolled the first time, you will be asked to enter passwords for the SCM certificate private key and for key stores being generated by the tool.
6. Un-enroll un-enrolls the SCM instance associated with the chosen security domain. You will be asked for the administrator's username and password to access the administration server.

Example

```
>enrolltool demo
```

enroll

Enrolls an SSM instance by acquiring security certificates from the associated Administration Server. The enrollment is required to configure one-way or two-ways SSL communication (see Configuring SSL for Production Environments for more information). Before enrolling an SSM instance, make sure that the ALES Administration Server is running.

During the enrollment process, you will be asked for the administrator's username and password to connect to the ALES Administration Server. If the SSM is enrolled the first time, you will be

asked to enter passwords for the SSM certificate private key and for key stores being generated by the tool.

Usage

```
SSM_INSTANCE_HOME\adm\enroll.bat <demo|secure>
```

```
SSM_INSTANCE_HOME/adm/enroll.sh <demo|secure>
```

Options

demo

Enrolls the SSM instance and verifies Administration Server certificate using the demo CA certificate from the `DemoTrust.jks` key store in directory `SSM_INSTANCE_HOME/ssl`. If this option is specified, the tool does not verify matching of the Administration Server host with the one from the certificate. This option should never be used in a production environment.

secure

Enrolls the SSM instance and verifies the Administration Server certificate using trusted CA certificates from the file `cacerts` in directory `BEA_HOME/jdk142_08/jre/lib/security`. If this option is specified, the tool verifies matching of the Administration Server host with the one from the certificate.

Example

```
>enroll demo
```

unenroll

Un-enrolls an SSM instance. As the result of the un-enrollment, the SSM identity certificate will be removed from the trusted-peer key stores of servers the SSM communicates to. Before un-enrolling an SSM instance, make sure that the ALES Administration Server is running.

During the un-enrollment process, you will be asked for the administrator's username and password to connect to the ALES administration server.

Usage

```
SSM_INSTANCE_HOME\adm\unenroll.bat <demo|secure>
```

```
SSM_INSTANCE_HOME/adm/unenroll.sh <demo|secure>
```

Options

demo

Un-enrolls the SSM instance and verifies the Administration Server certificate using the demo CA certificate from the `DemoTrust.jks` key store in directory `SSM_INSTANCE_HOME/ssl`. If this option is specified, the tool does not verify matching of the Administration Server host with the one from the certificate. This option should never be used in a production environment.

secure

Un-enrolls the SSM instance and verifies the Administration Server certificate using trusted CA certificates from the file `cacerts` in directory `BEA_HOME/jdk142_08/jre/lib/security`. If this option is specified, the tool verifies matching of the Administration Server host with the one from the certificate.

Example

```
>unenroll demo
```

Configuration File Usage

All ALES configuration files are currently shipped in the `config` directory of the Admin, SCM, and SSM instance. This section describes which tools use the various configuration files, and for what purpose.

Administration Server Configuration Files

Table 2-2 describes which configuration files are required for which tools on an AquaLogic Enterprise Security Administration Server installation.

Table 2-2 Admin Configuration Files

Tool	Config File	Explanation
Admin Examples	`admin_install.properties`	This admin_install.properties file is set up like a Java properties file and can be read to determine the input parameters selected during the install of the Admin Server.
Admin installer when run in silent mode	`silent_install_admin.xml`	This silent_install_admin.xml file captures the input parameters selected during the install of the Admin Server. This file can later be used for doing silent installs for similar configurations.
annotation_transform.bat|sh	`annotation_config.properties` `annotation_transform.xml`	The annotation_transform tool invokes the annotation_transform.xml ant script and gets its configuration from annotation_config.proerties. This tool is only needed when you have created annotated policy files via Eclipse.
policyloader.bat|sh	`asi.properties`	The policyloader tool uses the asi.properties file to initialize the BLM API, which it uses to communicate with the BLM server.
policyIX.bat|sh	`policyIX_config.xml`	The policyIX_config.xml file needs to be updated before being used as input to the policyIX tool.

Table 2-2 Admin Configuration Files

Tool	Config File	Explanation
BLM WebApp	`WLESblm.properties`	The WLESblm.properties file is used to configure the BLM WebApp. The BLM looks for this file based on the setting for the Java option `ales.blm.home`, and then in `/config/WLESblm.propeties`.
WebService interface to BLM	`blm.wsdl and pd.wsdl`	The WSDL files are needed to compile a Web service client that will be able to talk to the BLM server via SOAP messages.
install_ales_schema.bat|sh uninstall_ales_schema.bat|sh upgrade_ales_schema.bat|sh databaseloader.bat|sh	`database.properties`	The database.properties file contains the JDBC URL specified during install and is used by persistence layer to connect to the database.
set-env.bat|sh set-wls-env.bat|sh (WLS 8.x) WLESWeblogic.conf (WLS 8.x) WLESTomcat.conf	`log4j.properties`	The log4j.properties file is referenced in the set-env files used when configuring an SSM. This file controls the log4j logging for the entire SSM.
set-wls-env.bat|sh (WLS 9.x) WLESWeblogic.conf (WLS 9.x)	`log4j.wls9.properties`	The log4j.wls9.properties file is similar to log4j.properties, but specific to when used for a WLS9.x SSM.
set-env.bat|sh set-wls-env.bat|sh WLESWeblogic.conf WLESTomcat.conf	`WLESarme.properties`	This file is used to configure the ASI Authorization provider.

Table 2-2 Admin Configuration Files

Tool	Config File	Explanation
WLESWebLogic.bat|sh	`WLESWebLogic.conf`	This file is used by the Wrapper tool that is used to start the WebLogic server.
WLESTomcat.bat|sh	`WLESTomcat.conf`	This file is used by the Wrapper tool that is used to start the Tomcat server.
propagateInitialCache.bat|sh	`asiadmin.xml`	This asiadmin.xml file is used by the propagateInitialCache tool. The tool runs only for bootstrap purposes upon install to properly initialize the SCM and Admin SSM. It is automatically run as part of database schema install or via "WLESadmin.bat|sh init".
load_adminpolicy.bat|sh	`load.standardbase.conf`	The load.standardbase.conf file is used by the load_adminpolucy tool to load the initial Admin policy after a fresh install.
	`security.properties`	This file contains ALES configuration properties for an SSM. By default, the ALES runtime looks for a property file called 'security.properties' in the working directory. Only applicable to SSM running on Tomcat and WLS8.x.
	`SSM.properties`	Can be used to determine the location of SCM and Admin install directories.

Table 2-2 Admin Configuration Files

Tool	Config File	Explanation
	`loaderauthority.xml`	This file was the Naming Authority file used by the policyloader tool. It use of this file has now been deprecated and is no longer used.
	`healthlog4j.properties`	This file controlled the log4j settings for the Java wrappers that were used for running BLM and ARME native processes. The use of this file has now been deprecated and is no longer used.

SCM Configuration Files

Table 2-3 describes which configuration files are used by the SCM install.

Table 2-3 SCM Configuration Files

Tool	Config File	Explanation
	`scm_install.properties`	This scm_install.properties file is set up like a Java properties file and can be read to determine the input parameters selected during the install of the Admin Server.
WLESscm.bat|sh	`WLESscm.conf`	This file is used by the Wrapper tool that is used to start the SCM server.
asisignal.bat|sh enrolltool.bat|sh WLESscm.conf	`log4j.properties`	The log4j.properties file is referenced from the startup scripts of the tools.
enrolltool.bat|sh	`SCM.properties`	Properties file for the SCM

Table 2-3 SCM Configuration Files

Tool	Config File	Explanation
WLESscm.conf	`kernel.xml`	Config file for the Phoenix Java container framework that is used for creating the SCM process.
WLESscm.conf	`java.policy`	Configures the security policy for the SCM Java process.

Note: The SCM process is also controlled by *`SCM_HOME`*`/apps/scm-asi/SAR-INF/config.xml`. This file controls the various modules that make up the SCM process.

SSM Common Configuration Files

Table 2-4 describes which configuration files are used by the SSM instance. Most files are common between various types of SSM instances; those that are specific to an SSM are described in the explanation column. Most files are located in the `config` directory but when this is not the case the directory is listed.

Table 2-4 Common SSM Config Files

Tool	Config File	Explanation
	`SSM_HOME/adm/ssm_install.properties`	This ssm_install.properties file is set up like a Java properties file and can be read to determine the input parameters selected during the install of the SSM. Unlike most other files, this file is located in the SSM_HOME/adm directory.
	`SSM_HOME/adm/silent_install.properties`	This silent_install.xml file captures the input parameters selected during the install of the SSM. This file can later be used for doing silent installs for similar configurations. Unlike most other files, this file is located in the SSM_HOME/adm directory.

Table 2-4 Common SSM Config Files

Tool	Config File	Explanation
SSM Examples	`adm/ssm_instance.properties`	This ssm_instance.properties file is set up like a java properties file and can be read to find out what were the input parameters selected during the install of the SSM and creation of the SSM instance. Unlike most other files, this file is located in the INSTANCE_HOME/adm directory.
SSM instance wizard when run in silent mode	`adm/silent_instance.xml`	This silent_instance_admin.xml file captures the input parameters selected during the install of the SSM and creation of the SSM instance. This file can later be used for doing silent installs for similar configurations. Unlike most other files, this file is located in the INSTANCE_HOME/adm directory.
annotation_transform.bat|sh	`annotation_config.properties` `annotation_transform.xml`	The annotation_transform tool invokes the annotation_transform.xml ant script and gets its configuration from annotation_config.proerties. This tool is needed only when you have created an annotated policy files via Eclipse.
policyloader.bat|sh	`asi.properties`	The policyloader uses the asi.properties file to initialize the BLM API that it uses to communicate with the BLM server.
policyIX.bat|sh	`policyIX_config.xml`	The policyIX_config.xml file needs to be updated before being used as input to policyIX tool.

Table 2-4 Common SSM Config Files

Tool	Config File	Explanation
set-env.bat|sh enroll.bat|sh unenroll.bat|sh WLESarme.properties	`log4j.properties`	The log4j.properties file is referenced from the startup scripts of the tools.
set-env.bat|sh	`WLESarme.properties`	This file is used to configure the ASI Authorization provider.
	`shortcut.xml`	Internal file used on Windows to control the shortcut menu items.
	`SSM.properties`	Can be used to determine the location of SCM and Admin install directories.
	`loaderauthority.xml`	This file was the Naming Authority file used by the policyloader tool. The use of this file has now been deprecated and it is no longer used.
	`healthlog4j.properties`	This file controlled the log4j settings for the Java wrappers that were used for running BLM and ARME native processes. The use of this file has now been deprecated and it is no longer used.

Web Serivce SSM Configuration Files

The files shown in Table 2-5 are specific to the Web Service SSM.

Table 2-5 Web Service SSM Configuration Files

Tool	Config File	Explanation

	`access_control-xacml-2.0-context-schema-os.xsd`
	`access_control-xacml-2.0-policy-schema-os.xsd`
	`xacml.wsdl`
		XML schema and WSDL files that will be required when creating a WS SSM XACML client to connect to the WS SSM XACML WebService endpoint.

	`ssm-soap-types.xsd` `SSM-SOAPWS.wsdl`	XML schema and WSDL files that will be required when creating a WS SSM client to connect to the WS SSM server.
WLESws.bat|sh	`WLESws.wrapper.conf`	This file is used by the Wrapper tool that is used to start the Web service server.
	`security.properties`	Properties file for the Web service server.
WLESws.wrapper.conf	`kernel.xml`	Config file for the Phoenix Java container framework that is used for creating the Web service Java process.
WLESws.wrapper.conf	`java.policy`	Configures the security policy for the Web service Java process.

WLS SSM Configuration Files

The files shown in Table 2-6 are specific to the WLS SSM.

Table 2-6 WLS SSM Configuration Files

Tool	Config File	Explanation
	`DefaultAuthorizerInit.ldift`	ALES version of the LDIF Template file used by the WLS DefaultAuthorizer Provider. This file needs to be copied to the WLS domain if you plan to configure the WLS DefaultAuthorizer and ASI Authorizer providers together for the same SSM configuration.
	`XACMLAuthorizerInit.ldift (only WLS9.x)`	ALES version of the LDIF Template file used by the WLS XACMLAuthorizer Provider. This file needs to be copied to the WLS domain if you plan to configure the WLS XACMLAuthorizer and ASI Authorizer providers together for the same SSM configuration.
WLESWebLogic.bat|sh	`WLESWebLogic.conf`	This file is used by the Wrapper tool that can be used to start the WebLogic server.
	`security.properties`	Properties file for the WLS SSM server. Only applicable to SSM running WLS8.x.

CHAPTER 3

WLESblm.conf Reference

Configuration parameters for the Business Logic Manager (BLM) are stored in the `WLESblm.conf` file, located in the `BEA_HOME/ales26-admin/config` directory. While in most cases configuring AquaLogic Enterprise Security can be accomplished using the installation program and the ALES Administration Console, there may be some cases in which you want to change default configurations by editing the `WLESblm.conf` file. This section provides a reference to the `WLESblm.conf` parameters.

Required Parameters

The following required parameters are set when the ALES Administration Server is installed. These configuration parameters are essential for the BLM to start; if you change any of these values, you must restart the server before the changes will take effect.

Table 3-1 Required Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin.domain	The enterprise domain on which BLM is running.	asi
BLM.wlesadmin.location	location (Must be DEFAULT)	DEFAULT
BLM.wlesadmin. ASIPolicyARMEAddresses	The address of the ARME as a URL. The BLM directs authorization requests to this URL.	https://hostname:7012
BLM.wlesadmin. trustedPeerKeyStore	The file that contains the trusted Peer certificates in PEM format.	D:/opt/bea/81/ales26-admin/ssl/peer.pem
BLM.wlesadmin. trustedCAKeyStore	The file that contains the trusted CA certificates in PEM format.	D:/opt/bea/81/ales26-admin/ssl/trust.pem
BLM.wlesadmin.identityKeyStore	The file that contains the server's own certificate in PEM format.	D:/opt/bea/81/ales26-admin/ssl/wles-admin.pem
BLM.wlesadmin.identityAlias	The alias that will be used to retrieve the identity private key	wles-admin
BLM.wlesadmin.passwordfile	Location of the password file that contains encrypted passwords indexed with an alias. The alias and the private key file are required to retrieve the password.	D:/opt/bea/81/ales26-admin/ssl/password.xml
BLM.wlesadmin.passwordkeyfile	The password key is the master key that is required to retrieve any passwords from the password file.	D:/opt/bea/81/ales26-admin/ssl/password.key
BLM.wlesadmin.configkeyfile	The config key is the master key that is required to decrypt any attributes set as sensitive in the database.	D:/opt/bea/81/ales26-admin/ssl/config.key

Miscellaneous Configuration Parameters

The following optional parameters are set to default values during installation.

Table 3-2 Miscellaneous Configuration Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin.port	The BLM's listening port. The BLM runs on HTTP/SOAP. The default value is the default SOAP port, 80.	80
BLM.wlesadmin. adminPolicyRoot	The admin policy root is created when you install the ALES Administration Server. If, after installation, you make any change to the tree structure, you need to update this parameter as well. You do not need to change this parameter unless you are making changes to the security policies that protect the ALES administration resources.	//app/policy/ASI/admin
BLM.wlesadmin. defaultdirectory	This setting is used by BLM to locate the Administrator user when the user's directory is not provided by the BLM client at the time of making connection. This directory stores the administration server user and user groups that are used to boot the server and BLM API login. By default, ALES admin user IDs are maintained in the asi admin directory and custom identities for application-related users would be stored in a directory other than the asi directory. You do not need to change this unless you are making changes to the default admin policy.	asi

Table 3-2 Miscellaneous Configuration Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin. AuditWebserviceURL	The URL of the Web Service host to which BLM directs authorization audit events. You do not need to change this parameter unless you have changed the IP address and port on which the Audit Web Service runs.	https://127.0.0.1:7014
BLM.wlesadmin.AuditRetries	Number of times the server will try to send audit events to the Audit Web Service before giving up. This must be an integer greater than 0. If the server cannot connect to the Audit Web Service, no exception is thrown, but a debug message will note the failure.	2
BLM.wlesadmin.contextsize	When the BLM reaches a number of connections equal to the `contextsize` value, including the connections that have already timed out, the BLM will try to drop the timed-out connections that have not been accessed for a number of seconds equal to or greater than the `sessionTimeout` value. Set a lower value for more frequent clean-up as compared to default value of 40.	40
BLM.wlesadmin. sessionTimeout	When the BLM has a number of connections equal to the `contextsize` value, it will try to drop connections that have not been accessed for a number of seconds equal to or greater than the `sessionTimeout` value not been accessed for a number of seconds equal to or greater than the `sessionTimeout` value.	7200 seconds (2 hours)

Table 3-2 Miscellaneous Configuration Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin. maxCollectionSize	The maximum number of entries in one collection. This limits the collection size used by the BLM process when dealing with collections such as collection of users, user groups, subjects, attributes, etc. For example, if you are listing the users in the identity directory user groups, the BLM would retrieve the first 500 users under the user group the first time, but the console would display a part of the 500 users and get the rest as the console user views them using the up and down arrows in the console. If you increase the value of `maxCollectionSize`, the result set would increase accordingly, thereby loading more users even though you may not list all the users. As a result the performance is more of a management time latency (administration time) and not a runtime evaluation latency, since the ARME caches the policy and user information locally rather than using the BLM for runtime authorization and role mapping decisions. If this value is set too large, it will reduce console and BLM performance and increase BLM memory usage.	500

Table 3-2 Miscellaneous Configuration Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin. maxTreeSizeWithResourceNodes	The maximum number of app nodes with resource nodes to display in the object tree. This is just a display and fetch restriction; the subsequent 500 resources are fetched as the console user views them with the up and down arrows. If this value is set too large, it will reduce administration console and BLM performance and increase BLM memory usage.	500
BLM.wlesadmin. requestThreads	The size of the ASI thread pool size that handles client requests. This value should be increased only if the server that hosts the BLM server is able to handle that many threads without maxing out the CPU usage.	10
BLM.wlesadmin. masterSocketReadTimeoutMs	Timeout for the master socket on which server was reading a request. Determines how long to wait on the sockets with no input before timing out. This is used both to periodically check for a shutdown request, and to allow connections which have given up their thread to be watched and rescheduled.	1
BLM.wlesadmin. childSocketReadTimeoutMs	Timeout for the child socket on which server was reading a request. Determines how long to wait on the sockets with no input before timing out. This is used both to periodically check for a shutdown request, and to allow connections which have given up their thread to be watched and rescheduled.	1

Logging Configuration Parameters

The following optional configuration parameters control ALES logging behavior. Note that you may direct all logging entries to a single file. You can also direct logging entries to the stdout or stderr streams using the keywords `stdout` or `stderr`.

Table 3-3 Logging Configuration Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin.logLevel	Determines which events get logged. Valid values are integers from 0 to 63. The value is interpreted as a bitfield. Add the levels together to determine the value. The following log levels are defined: 0 (error) If errors are generated, then they will always be logged. 1 (log) Enable log output. 2 (dbg) Enable debug output. 4 (eviction) Log any session eviction that takes place to free up idle connections. 8 (exceptions) Exceptions thrown by the BLM server. 16 (comTest calls) Server heartbeat calls to check that server is functioning correctly. 32 (soap calls) All IN/OUT SOAP messages at the transport level.	0
BLM.wlesadmin.logfile	Logging location.	D:/opt/bea/81/ales26-admin/log/WLESblm.log
BLM.wlesadmin.errfile	Logging location for error log entries.	D:/opt/bea/81/ales26-admin/log/WLESblm.log
BLM.wlesadmin.dbgfile	Logging location for debug log entries.	D:/opt/bea/81/ales26-admin/log/WLESblm.log
BLM.wlesadmin.DbgOut	Logging location for C++ client debug entries, which do not have log levels assigned.	D:/opt/bea/81/ales26-admin/log/WLESblm.log

Table 3-3 Logging Configuration Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin. logShowDateTime	Include the date and time in the logging header. If enabled, the date and time the message was logged are prepended to the log message. 0 - disabled 1 - enabled	1
BLM.wlesadmin. logShowFileName	Include the file name and line number in the logging header. If enabled, the file name and line number causing the event being logged are prepended to the log message. 0 - disabled 1 - enabled	1
BLM.wlesadmin. logShowThread	Include the executing thread number in the logging header. If enabled, the executing thread number causing the event being logged are prepended to the log message. 0 - disabled 1 - enabled	1

Database Configuration Parameters

The following configuration parameters are set during installation. You do not need to change these values unless you change the database to which the BLM connects.

Table 3-4 Database Configuration Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin.dbsystem	The database system used by the client. Valid values are: ORACLE92, ORACLE90, ORACLE81 SYBASE125, SYBASE120, SYBASE119 In addition, for backwards compatibility, the value ORACLE is treated as ORACLE81 and the value SYBASE is treated as SYBASE125.	ORACLE92
BLM.wlesadmin.dbserver	The database server name (the database service name for Oracle).	ASI.DB.EXAMPLE.COM
BLM.wlesadmin.dbname	Database name. This parameter is only applicable in Sybase and is ignored in Oracle.	sspolicy
BLM.wlesadmin.dbpolicyowner	The user name of the policy owner. Generally, this will be the same as the `dblogin` user.	
BLM.wlesadmin.dblogin	Database login ID. This user ID must be granted database permissions. Usually it is the schema (policy) owner which has all the permissions.	

Table 3-4 Database Configuration Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin. dbpoolsize	Number of database connections in shared pool to be allocated for the BLM. Consult with your database administrator before setting this to a number greater than 20, since there is typically a limited number of connections configured in the database server.	20
BLM.wlesadmin. dbconnidletimeout	If a database connection has been idle for this number of seconds, it is disconnected. This is to make sure the BLM does not hold on to unused database connections for a long period.	600 seconds

Table 3-4 Database Configuration Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin.sqldebug	Enables or disables database SQL debugging (bit wise). In a production environment, set it to 0 or 1. In order for SQL debug logging to function, the BLM.wlesadmin.logLevel dbg bit must be set. 0 - hard database error 1 - soft database error (recoverable) 2 - SQL debugging 4 - Stored procedure debugging Add the levels together to come up with the value.	0
BLM.wlesadmin. fetchnumrows	Indicates how many delta elements should be returned from the database as part of the query resultset as opposed to loading all of the results at once. The subsequent get on the 1001th item would fetch the next 1000 results and so forth. To process the results, the collection configuration would take two passes with 500 items on the first and 500 on the next pass and so on. This value is a trade-off between latency during administration actions and latency at evaluation.	1000

CPP API Configuration Parameters

The following parameters relate to the CPP API used by the BLM to call the ARME for Authorization decisions.

Table 3-5 CPP API Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin.cacheEnabled	Is authorization caching enabled? 0 indicates disabled and 1 indicated enabled.	0
BLM.wlesadmin.cacheFileName	If authorization caching enabled, the cache is written to this file.	asiCpp.cache
BLM.wlesadmin.relocateRetries	Number of times the server will try to get a new reference to an ARME in case the current one becomes unavailable.	10
BLM.wlesadmin.relocateInterval	The interval between retries in milliseconds.	1000
BLM.wlesadmin.autoRelocateInterval	Setting this to anything other then MS_INFINITE causes the server to automatically drop current ARME connection and try to establish a new connection after every interval in milliseconds.	MS_INFINITE
BLM.wlesadmin.reconnectRetries	Number of times the server will try the same connection before relocating and using another one.	1

Distribution Parameters

The following set of parameters are dependent on the ARME policy distribution and provisioning states. The BLM distribution component uses these timeout settings to communicate with ARME. Override timeouts as desired during the various distributed transaction phases.

Table 3-6 Distribution Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin. ARMECountRequiredToCommit	Defines the number of ARME instances within a ARME group that are required to successfully receive the new policy in order for the group to commit the policy. If this number isn't met then the entire group will be rolled back and stay on the existing policy. If the number is met, then any ARME instance that did not successfully commit the new policy will be put into the unbound group. If fewer ARME instances are alive than this value, then all ARME instances in the group must successfully receive the policy for the group to commit.	1
BLM.wlesadmin. ARMEPrepareToCommitTimeoutMS	Determines how long the BLM will wait for a ARME to finish the prepareToCommit stage of a policy distribution.	10800000 (3 hours)
BLM.wlesadmin. pendingARMEWaitMS	Determines how long the BLM should wait for new ARMEs to request a policy, before distributing in parallel to the unbound group.Each time a new ARME shows up, the BLM will wait this long for one more to show up.	10000
BLM.wlesadmin. pendingARMEWaitMaxMS	Determines the maximum amount of time the BLM will wait before distributing to the unbound group	120000
BLM.wlesadmin. ARMECommitTimeoutMS	Determines how long the BLM will wait for a ARME to finish the commit stage of a policy distribution.	300000 (5 minutes)

Table 3-6 Distribution Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin. ARMERollbackTimeoutMS	Determines how long the BLM will wait for a ARME to finish the potential rollback stage of a policy distribution.	300000 (5 minutes)
BLM.wlesadmin. ARMEDeltaTimeoutMS	Determines how long the BLM will wait for the ARME to finish the delta stage of a policy distribution.	300000 (5 minutes)
BLM.wlesadmin. ARMEBeginPolicyUpdateTimeoutMS	Determines how long the BLM will wait for a BLE to finish the begin policy update stage of a policy distribution. The default is 5 minutes	300000 (5 minutes)
BLM.wlesadmin.deltaSendNumRows	Indicates how many delta elements should be sent to the servers by the BLM at one time. Increasing the number may improve performance but will increase overhead.	1000
BLM.wlesadmin.syncType	Defines the desired level of synchronization when committing a new policy. There are three levels of synchronization; each level includes the previous levels: 0 - group, all instances in a group must be able to commit the new policy for any to commit. 1 - location, all groups in the location must be able to commit the new policy for any to commit. 2 - domain, all locations in the domain must be able to commit the new policy for any to commit.	1

Default Timeout Parameters

The following parameters are default values for the underlying transport for client/server connections made by the BLM to the ALES Administration Servers.

Table 3-7 Default Timeout Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin. defaultSendTimeoutMs	When the BLM is sending a request, this transport timeout controls when, in milliseconds, to time out if it cannot send data.	10000
BLM.wlesadmin. defaultRecvTimeoutMs	When the BLM has made a request to another server, this transport timeout controls how long to wait, in milliseconds, before disconnecting.	10000
BLM.wlesadmin. defaultConnectTimeoutMs	When the transport cannot connect to another server, this transport timeout controls how long to wait in milliseconds before giving up.	10000

Override Timeout Parameters

The following timeout parameters are used by the BLM pool manager to override timeouts on its pool of BLM connections based on the activity performed.

Table 3-8 Override Timeout Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin.connectTimeout	When the transport cannot connect to the ARME, this transport timeout controls how long to wait in milliseconds before giving up	10000
BLM.wlesadmin.sendTimeout	When the BLM is sending a request to the ARME, this transport timeout controls when to timeout, in milliseconds, if it cannot send data	10000

Table 3-8 Override Timeout Parameters

BLM.wlesadmin.requestTimeout	When a request has been made by the BLM to the ARME, this transport timeout controls how long to wait, in milliseconds, before disconnecting	10000
BLM.wlesadmin.relocateOnError	Controls whether to keep using the same connection (0) or relocate (1) if errors occur while communicating with the ARME	1
BLM.wlesadmin.maxRetries	Maximum number of retries for the same ARME before relocation takes place	10

CHAPTER 4

Provider Extensions

The following topics are covered in this section:

What is a Provider Extension?

A provider extension is a plug-in function that you write to extend the capabilities of the existing providers. You can use these plug-ins to manipulate existing policy data in a way that is not already provided, or to retrieve data from external sources to add to an authorization or role mapping decision or a deployment audit. You can use these plug-ins with the ASI Authorization, ASI Role Mapping, Log4j Audit Channel, and Database Authentication providers.

While the security providers supplied with AquaLogic Enterprise Security are configurable, the plug-ins enable you to customize them to add additional functionality. For example, you may want some form of special business logic to retrieve additional data that you want to use before the authorization decision is made or for the custom processing of data, such as the audit context. Plug-ins are provided for a variety of functions:

- You can use Java-based plug-ins to perform attribute retrieval, attribute conversion, and resource conversion. You can use attribute retrievers to retrieve embedded data from complex data objects or external data sources. You can use resource converters to convert WebLogic Server and AquaLogic Enterprise Security data to an internal resource format. You can use attribute converters to convert context data to an internal attribute format.
- You can use Java extensions plug-ins to add your own custom authorization and role mapping evaluation functions to the standard ones provided. After you develop a function, administrators can manipulate its input using the Administration Console. The plug-in appears to the administrator as simply new evaluation functions or newly available dynamic attributes.
- You can use the audit plug-ins to help format audit events that are generated by the Security Framework, the runtime API, or custom implementations.
- You can use the database authentication plug-in with the Database Authentication provider to customize authentication features.

Note: If you using the WebLogic Server 9.x SSM:

- The security provider JAR files in the WebLogic 9.x SSM (in directory `SSM/lib/providers/wls/v9`) are not compatible with those in the WebLogic 8.1 SSM (in directory `SSM/lib/providers`). For building classes for use with the WebLogic 9.x SSM, include the JARs in `SSM/lib/providers/wls/v9` in your CLASSPATH and all JAR files you build for use by the providers should be placed in `SSM/lib/providers/wls/v9`. To build classes for use with other SSMs, include the JARs in `SSM/lib/providers` in your CLASSPATH and all JAR files you build for use by the providers should be placed in `SSM/lib/providers`.
- Configuration of the ASI Authorization security provider and the ASI RoleMapper security provider needs to be done using both the ALES Administration Console and the WebLogic Server Administration Console.

The following sections provide more information on the plug-ins and how to use them.

- "Authorization and Role Mapping Extensions" on page 4-3
- "Custom Audit Plug-ins" on page 4-19
- "Database Authentication Plug-in" on page 4-20

asi_classes JAR Required for Provider Extensions

The `asi_classes.jar` (located in `SSM/lib`) contains classes required for provider extensions. That is, in order to implement provider extensions you need classes in `asi_classes.jar`.

Authorization and Role Mapping Extensions

AquaLogic Enterprise Security supports using Java-based plug-ins and language extensions with security providers. You can use these plug-ins to performed custom functions for authorization and role mapping.

The following types of plug-ins are supported:

- "Using Java-Based Plug-ins" on page 4-3
- "Configuring a Custom Attribute Converter" on page 4-13
- "Using Java Entensions Plug-Ins" on page 4-13

Using Java-Based Plug-ins

AquaLogic Enterprise Security providers support three types of Java-based plug-ins: resource converters, attribute retrievers, and attribute converters. Table 4-1 shows the types of Java-based plug-ins that each security provider supports.

Table 4-1 Java-based Plug-in Support for Security Providers

Security Provider	Supports Resource Converter Plug-in	Supports Attribute Retrievers Plug-in	Supports Attribute Converter Plug-in
ASI Authorization provider	Yes	Yes	Yes
ASI Role Mapping provider	Yes	Yes	Yes

To use these plug-ins, you must perform the following tasks:

1. Write a Java class that implements the plug-in interface.
2. Place the Java class in the appropriate directory of the Security Service Module with which you intend to use the plug-in. A single Java class may be used with more than one Security Service Module.
3. Use the Administration Console to register the Java class on the desired Security Service Module(s). If you are using the WebLogic 9.x SSM, configuration of the ASI Authorization security provider and the ASI RoleMapper security provider needs to be done using both the ALES Administration Console and the WebLogic Server Administration Console.

For instructions for performing these tasks, refer to the following sections:

- "Using the Java-based Plug-in Interfaces" on page 4-4
- "Resource Converter" on page 4-5
- "Attribute Retriever" on page 4-7
- "Attribute Converter" on page 4-12

Using the Java-based Plug-in Interfaces

To implement a Java-based plug-in interface, you must perform the following steps:

1. Refer to the description of the plug-in interface you want to use and write a Java class to implement the interface. The following sections provide descriptions of each type of plug-in interface:
   - "Resource Converter" on page 4-5
   - "Attribute Converter" on page 4-12
   - "Attribute Retriever" on page 4-7
2. Use the Java class to create a JAR file and place the JAR file in the `/lib/providers` directory (or, if you are using the WebLogic 9.x SSM, in the `/lib/providers/wls/v9` directory) in the installation directory for the Security Service Module on the machine on which the Security Service Module is installed. For example, the default location of this directory for the WebLogic Server Security Service Module is `C:\bea\ales26-ssm\wls-ssm\lib\providers`.
3. You can use Log4j libraries to insert debug statements in your plug-ins. The example found in `java-ssm\examples\AttributeRetriever` illustrates use of Log4j debugging messages.
4. Refer to the following topics in the Console Help and use the Administration Console to register the Java plug-ins in the desired security providers for the desired Security Service Modules:
   - Configuring an ASI Authorization Provider
   - Configuring an ASI Role Mapping Provider

Resource Converter

Resource converters are used by ASI Authorization and ASI Role Mapping providers to convert WebLogic Server resources into an internal resource format that is recognized by AquaLogic Enterprise Security. For a description of the policy data formats, see the *BEA AquaLogic Enterprise Security Policy Managers Guide*.

`ResourceConverter` is an interface in the `com.bea.security.providers.authorization.asi` package. This interface is used to implement plug-ins for converting from the Security Framework defined resource interface into an access query. There is no standard for resource definitions so plug-ins are needed to handle each of the resource types. The set of resource types is not fixed and you can define your own resource, in which case, you need to define a resource converter to allow the ASI Authorization provider to protect the resource. Numerous resource converters are supplied for your use, one for each defined WebLogic Server and AquaLogic Enterprise Resource type. Table 4-2 lists and describes the methods provided by the `ResourceConverter` interface.

Table 4-2 ResourceConverter Interface Methods

Method	Description
`String[] getHandledTypes()`	This method is called when the plug-in is instantiated and is used to determine what resource types the converter knows how to handle. The Security Framework represents resource types internally as strings.
`AccessElement convertResource(Resource resource,ContextHandler contextHandler) throws ResourceConversionException`	This method extracts enough information from a `Resource` and `ContextHandler` to form an access query. The minimum amount of required information to be extracted is the resource object and privilege. Additional information that can be included is the application name and input attributes extracted from the Resource or `ContextHander`: If the application is not specified, then the provider uses the following rules for selecting one: • If no application is specified, then the object is queried under the shared resource node as specified in the provider configuration. • If an unqualified application is specified, the object is queried under the default deployment node, plus the application, plus the object. • If a fully qualified application is specified, then the object is queried under that node. If the resource converter is unable to generate an access query from the information provided in the `Resource`, it throws a `ResourceConversionException` indicating to the provider and framework that this query cannot be answered by this provider.
`Object getAttributeValue (Resource resource, String name,ContextHandler contextHandler)`	This method finds the value of a missing attribute. It is left up to you as the developer of the ResourceConverter plug-in to determine how the `ResourceConverter` gets the required value. The plug-in may return null if the value is not found.

Configuring a Custom Resource Converter

To configure a custom resource converter, you must implement the resource converter and register it with the configured ASI Authorization and ASI Role Mapping providers.

To configure a resource converter, perform the following steps:

1. Implement a custom resource converter and use the Java class to create a JAR file.

The `com.bea.security.providers.authorization.asi.ResourceConverter` class is in the *`BEA_HOME`*`\ales26-ssm\<`*`ssm-type`*`>\lib\providers\ASIAuthorizer.jar` (or, if you are using the WebLogic Server 9.x SSM, use `BEA_HOME\ales26-ssm\wls9-ssm\lib\providers\wls\v9\ASIAuthorizer.jar`). Include this file, and the *`BEA_HOME`*`\ales26-ssm\<`*`ssm-type`*`>\lib\asi_classes.jar`, in the classpath when compiling the custom resource converter.

2. Place the JAR file in the `/lib/providers` directory (or, if you are using the WebLogic Server 9.x SSM, use `/lib/providers/wls/v9`) in the installation directory for the Security Service Module (SSM) on the machine on which the SSM is installed (either the WebLogic Server SSM or the Java SSM). For example, the default directory for the WebLogic Server SSM is `C:\bea\ales26-ssm\wls-ssm`.

3. In the left pane of the Administration Console, click the ASI Authorization provider configured for the SSM instance, select the Advanced tab in the right pane, type in the fully qualified name of your custom converter in the Resource Converters field, and click Apply. If you are using the WebLogic Server 9.x SSM, configuration changes to the ASI Authorization provider must also be made using the WebLogic Server Administration Console.

4. Repeat step 3. to register the Resource Converter with the ASI Role Mapping provider. If you are using the WebLogic Server 9.x SSM, configuration changes to the ASI Role Mapping provider must also be made using the WebLogic Server Administration Console.

5. In the left pane, click Deployment, select the Configuration tab, and deploy the configuration change to the SSM.

6. Restart the SSM process.

Attribute Retriever

Attribute retrievers are used by ASI Authorization and ASI Role Mapping providers to retrieve attributes for use by AquaLogic Enterprise Security authorization and role mapping. `AttributeRetriever` is an interface in the `com.bea.security.providers.authorization` package that you can use to implement plug-ins for retrieving attributes. You use an implementation of the `AttributeRetriever` interface to get embedded data from complex data objects. For example, if the `ContextHandler` includes an address element, you can use an attribute retriever to make the zip code portion of the address available separately. You can also use an attribute retriever to fetch data from external data sources, for example, JDBC data stores.

Version 2.5 of AquaLogic Enterprise Security includes a new version of the `AttributeRetriever` interface, `AttributeRetrieverV2`. The `AttributeRetrieverV2` interface includes an additional RequestHandle parameter.

Note: It is generally not necessary to write attribute retrievers for objects that appear directly in the `ContextHandler`; attribute retrievers are used to extract embedded or otherwise inaccessible data.

You can register multiple attribute retrievers with the same attribute name. If you do so, the attribute retrievers are called in order until one of them returns a non-null result.

Table 4-3 lists and describes the methods provided by the `AttributeRetriever` interface.

Table 4-3 AttributeRetriever Interface Methods

Method	Description
`String[] getHandledAttributeNames()`	This method returns the names of attributes handled by this object. An attribute retriever must return at least one attribute name in the getHandledAttributeNames() method. If the method returns null or an empty value, this attribute retriever will not be invoked again.
`Object getAttributeValue( String name, Subject subject, Map roles, Resource resource, ContextHandler contextHandler)`	This method retrieves the value of the named attribute. Additional authorization request data is made available to allow for more complex attribute retrieval. The parameters are as follows: • name—name of the needed attribute • subject—subject associated with the request • roles—role membership of the subject, or `null` if this is a role mapping call • resource—resource associated with the request • contextHandler—context associated with the request; may be `null` if non-existent This method returns the attribute value, or `null` if the attribute is not found.

Table 4-4lists and describes the methods provided by the `AttributeRetrieverV2` interface.

Table 4-4 AttributeRetrieverV2 Interface Methods

Method	Description
`String[] getHandledAttributeNames()`	This method returns the names of attributes handled by this object. An attribute retriever must return at least one attribute name in the getHandledAttributeNames() method. If the method returns null or an empty value, this attribute retriever will not be invoked any more.
`Object getAttributeValue( String name, RequestHandle requestHandle, Subject subject, Map roles, Resource resource, ContextHandler contextHandler)`	This method retrieves the value of the named attribute. Additional authorization request data is made available to allow for more complex attribute retrieval. The parameters are as follows: • name—name of the needed attribute • subject—subject associated with the request • RequestHandle—the provider configuration parameters associated with the request, through which the function can get the required attribute value. The `com.bea.security.providers.authorization.asi.ARME.evaluator.RequestHandle` interface is described in "RequestHandle getAttribute Method" on page 4-9. • roles—role membership of the subject, or `null` if this is a role mapping call • resource—resource associated with the request • contextHandler—context associated with the request; may be `null` if non-existent This method returns the attribute value, or `null` if the attribute is not found.

RequestHandle getAttribute Method

The `com.bea.security.providers.authorization.asi.ARME.evaluator.RequestHandle` interface, which is implemented for you and passed to your `AttributeRetrieverV2` implementation, has two methods, `getAttribute()` and `appendReturnData()`, which are defined as follows:

```
public AttributeElement getAttribute(String name, boolean typeCheck)
throws ArmeRuntimeException, BadParameterException,
CredvarException,BoolEvalInternalException, NotReadyException;

public void appendReturnData(String name, Object data);
}
```

Of the two methods, the `RequestHandle.getAttribute()` is of most interest to `AttributeRetrieverV2` implementations. The `RequestHandle.getAttribute()` method gets the named attribute and its value, and optionally type-checks the value. The `getAttribute()` method returns the attribute name and value as a `com.wles.util.AttributeElement` object.

The `AttributeElement` object represents an attribute name/value pair for a single element or a list. In the case of a list, your AttributeRetrieverV2 code might then transfer the `AttributeElement` list value to a list of String-type objects, such as in the following code fragment from the AttributeRetrieverV2 example:

```
try {

            AttributeElement attrElement =
requestHandle.getAttribute("asi_subjectgroups", true);


            Object value = null;


            //It must be a list attribute

            if(!attrElement.isList()) {

                return null;

            }


          //transfer AttributeElement value to a list of String type object.

            List subjectGroupList =
(List)attrElement.getValueAs(String.class);

            value = String.valueOf(subjectGroupList.size());

            return value;

        } catch(Exception e) {
```

```
            //failed to retrieve attributes.
            return null;
        }
```

Your `AttributeRetrieverV2` implementation can use the `RequestHandle.getAttribute()` method to get the value for user and resource attributes. However, it should not rely on the `RequestHandle.getAttribute()` method to get values of dynamic or extension attributes, as they may or may not be computed for a particular request.

The ASI providers always evaluate the minimum number of policies that are logically required, and attributes are retrieved only when a policy that references them is evaluated. Therefore, some attribute values will not be available. `AttributeRetrieverV2` will not get the attribute value unless the policy has been explicitly evaluated.

Configuring a Custom Attribute Retriever

Note: This release of AquaLogic Enterprise Security includes attribute retriever examples, in `BEA_HOME\ales26-ssm\java-ssm\examples\AttributeRetriever`.

To configure a custom attribute retriever, you must implement the attribute retriever, and then register it with the configured ASI Authorization and ASI Role Mapping providers.

To configure an attribute retriever, perform the following steps:

1. Implement a custom attribute retriever and use the Java class to create a JAR file.

   The `com.bea.security.providers.authorization.asi.AttributeRetriever` and `com.bea.security.providers.authorization.asi.AttributeRetrieverV2` classes are in the *`BEA_HOME`*`\ales26-ssm\<`*`ssm-type`*`>\lib\providers\ASIAuthorizer.jar` (or, if you are using the WebLogic Server 9.x SSM, use `BEA_HOME\ales26-ssm\wls9-ssm\lib\providers\wls\v9\ASIAuthorizer.jar`). Include this file, and the *`BEA_HOME`*`\ales26-ssm\<`*`ssm-type`*`>\lib\asi_classes.jar`, in the classpath when compiling the custom attribute retriever.

2. Place the JAR file in the `/lib/providers` directory (or, if you are using the WebLogic Server 9.x SSM, use `/lib/providers/wls/v9`) in the installation directory for the Security Service Module (SSM) on the machine on which the SSM is installed (either the WebLogic Server SSM or the Java SSM). For example, the default directory for the WebLogic Server SSM is `C:\bea\ales26-ssm\wls-ssm`.

3. In the left pane of the Administration Console, click the ASI Authorization provider configured for the SSM instance, select the Advanced tab in the right pane, type in the fully qualified name of your custom retriever in the Attribute Retrievers field, and click Apply. If

you are using the WebLogic Server 9.x SSM, configuration changes to the ASI Authorization provider must also be made using the WebLogic Server Administration Console.

4. Repeat step 3 to register the Attribute Retriever with the ASI Role Mapping provider. If you are using the WebLogic Server 9.x SSM, configuration changes to the ASI Role Mapping provider must also be made using the WebLogic Server Administration Console.
5. In the left pane, click Deployment, select the Configuration tab, and deploy the configuration change to the SSM.
6. Restart the SSM process.

Attribute Converter

Attribute converters are used by ASI Authorization and ASI Role Mapping providers to convert context data to an internal attribute format. For a description of the policy data formats, see the *BEA AquaLogic Enterprise Security Policy Managers Guide*.

To create attribute converters, you implement the `TypeConverter` interface. This interface converts between native Java types and ASI formatted Strings. If you create a new ASI type, you may want to create a Java class to handle it and implement a `TypeConverter` interface to handle that class. ASI types are the credential types that are visible through the console such as integer, date, and string types, and so on, versus Java data types.

Table 4-5 lists and describes methods provided by the `TypeConverter` interface.

Table 4-5 TypeConverter Interface Methods

Method	Description
`Class getType()`	This method returns the type which this converter converts.
`String getASITypeName()`	This method returns the ASI type name.
`String convertToASI(Object javaFormat) throws UnsupportedTypeException`	This method converts a java object into a ASI string.
`Object convertFromASI(String asiFormat) throws TypeConversionException`	This method converts a ASI string to a Java Object.

Configuring a Custom Attribute Converter

To configure a custom attribute converter, you must implement the attribute converter and register it with the configured ASI Authorization and ASI Role Mapping providers.

To configure an attribute converter, perform the following steps:

1. Implement a custom attribute converter and use the Java class to create a JAR file
2. Place the JAR file in the `/lib/providers` directory (or, if you are using the WebLogic Server 9.x SSM, use `/lib/providers/wls/v9/ASIAuthorizer.jar`) in the installation directory for the Security Service Module (SSM) on the machine on which the SSM is installed (either the WebLogic Server SSM or the Java SSM). For example, the default directory for the WebLogic Server SSM is `C:\bea\ales26-ssm\wls-ssm`.
3. In the left pane of the Administration Console, click the ASI Authorization provider configured for the SSM instance, select the Advanced tab in the right pane, type in the fully qualified name of your custom converter in the Attribute Converters field, and click Apply. If you are using the WebLogic Server 9.x SSM, configuration changes to the ASI Authorization provider must also be made using the WebLogic Server Administration Console.
4. Repeat step 3 to register the Attribute Converter with the ASI Role Mapping provider. If you are using the WebLogic Server 9.x SSM, configuration changes to the ASI Role Mapping provider must also be made using the WebLogic Server Administration Console.
5. In the left pane, click Deployment, select the Configuration tab, and deploy the configuration change to the SSM.
6. Restart the SSM process.

Using Java Entensions Plug-Ins

Note: This section replaces the version 2.2 Policy Language Custom Extension Library API Reference.

You can use Java extensions plug-ins to add your own custom initialization, shutdown, and authorization and role mapping evaluation functions to the standard ones provided. After you develop a function, administrators can manipulate its input using the Administration Console. The plug-in appears to the administrator simply as new evaluation functions or newly available dynamic attributes.

This section describes the Application Programming Interface (API) for writing custom extension functions to enhance features available through the policy language.

Custom Initialization/Shutdown Interface

Note: This release of AquaLogic Enterprise Security includes an initialization/shutdown function example, in *BEA_HOME*\ales26-ssm\java-ssm\examples\MyEvaluationFunction.

You can implement the `com.bea.security.providers.authorization.asi.InitializationShutdownFunction` interface to add initialization and shutdown functions for your plug-in.

The `init()` method is invoked during the Authorization or Role Mapping provider initialization and initializes attributes for the provider. For example, it might open a database connection pool. The `shutdown()` method is invoked during the Authorization or Role Mapping provider shutdown and might then close the connections to the pool.

The `InitializationShutdownFunction` interface includes a key map of constants and attributes primarily supported by the AsiAuthorizer security provider. A portion of this key map is shown in Listing 4-1.

Listing 4-1 `InitializationShutdownFunction` **Key Map**

```
/**
     * map key for get ATZ provider or RM provider's version info.
     */
    public static final String VERSION = "Version";


    /**
     * map key for get ATZ provider or RM provider's description info.
     */
    public static final String DESCRIPTION = "Description";
    /**
     * map key for get ATZ provider or RM provider's class name.
     */
    public static final String PROVIDER_CLASSNAME = "ProviderClassName";
    /**
```

```
     * map key for get advanced configuration properties for ATZ provider
or RM provider. The corresoponding value in map is <code>Properties<code>.
     */
    public static final String ADVANCED_CONFIGURATION_PROPERTIES =
"AdvancedConfigurationProperties";
    /**
     * map key for get ATZ provider or RM provider's directory.
     */
    public static final String DIRECTORY = "Directory";
    /**
     * map key for get configuration for preload attributes.
     */
    public static final String PRELOAD_ATTRIBUTES = "PreLoadAttributes";
    /**
     * map key for get configuration for session eviction capacity.
     */
    public static final String SESSION_EVICTION_CAPACITY =
"SessionEvictionCapacity";
:
:
```

The attributes are described in the BLM Configuration API Security Providers Reference. You use this key map to get initialization parameters for the plug-in.

Listing 4-2 shows the `init()` function from the *BEA_HOME*`\ales26-ssm\java-ssm\examples\MyEvaluationFunction` example.

Listing 4-2 Sample MyEvaluationFunction `init()` Method

```
/**
```

```
public void init(Map config) {
        // Initialization function executed.

        // Samples to get value from Map.

        //get advanced configuration properties
        Properties advanced =
(Properties)config.get(ADVANCED_CONFIGURATION_PROPERTIES);

        //get all configured evaluation function class names.
       String[] evalFunctions = (String[])config.get(EVALUATION_FUNCTIONS);

        String ifActivateOnStartUp =
config.get(ACTIVATE_ON_STARTUP).toString();
        return;
    }
```

Custom Evaluation Method Plug-in

You can use a named evaluation method to make additional authorization request data available, and therefore allow a more complex attribute evaluation to be performed. This method is invoked while the policy contains a custom evaluation function with a matching name. For example:

```
grant(any, //app/policy/ASI, //user/asi/test/) if myFunc(name);
```

where *myFunc()* is the custom evaluation function name. You must register one evaluation class that includes the *myFunc()* method.

You can use whatever name you want, but the name must exactly match the name you supply in the policy, and the method must have exactly these arguments:

```
public boolean some_method_name(RequestHandle requestHandle,
            Object[] args,
            Subject subject,
```

```
Map roles,
Resource resource,
ContextHandler contextHandler)
```

Table 4-6 contains a description of the evaluation method arguments.

Table 4-6 Evaluation Method Arguments

requestHandle	The attributes container associated with the request, through which the method can get required attribute value. See "RequestHandle getAttribute Method" on page 4-9 for a description of this method. See "RequestHandle appendReturnData Method" on page 4-17 for a description of this method.
args	An array of method arguments. Each element is either <code>null</code>, or a String
subject	The subject associated with the request
roles key value	The role membership of the subject. Null if the method is called during role mapping. The role name The role object.
resource	The resource associated with the request
contextHandler	The context associated with the request, may be <code>null</code> if non-existent
return	True or false as the result of the method

RequestHandle appendReturnData Method

The `RequestHandle.appendReturnData()` method appends the evaluation result to an object so that it can be retrieved after `atzSvc.isAccessAllowed()` is invoked.

You pass in an Object containing the evaluation result. The `appendReturnData()` method converts the Object to a suitable ASI type, and then appends it to a `com.bea.security.providers.authorization.asi.ARME.engine.EvalResultI` object. You can access this result after the `atzSvc.isAccessAllowed()` method is invoked. The

QueryResources example (*BEA_HOME*\ales22-ssm\java-ssm\examples\QueryResources) shows how to access the returned data.

If the same attribute value is redefined by another plug-in within the same policy, the result value is overwritten.

The evaluation result (the appended data) is returned only if the complete rule condition result is true. That is, the evaluation result is returned only if the policy successfully executes.

For example, if the rule condition includes two evaluation functions, both `func1(arg1)` and `func2(arg2)` can invoke `appendReturnData`, but the data is not available unless both functions evaluate to true. As a programming practice, do not call `appendReturnData` unless your evaluation result is true.

The policy might not execute if another policy makes evaluating it superfluous. For example, once a user is explicitly denied access with a deny policy, a thousand grant policies cannot undo the one deny. Knowing this, as soon as a single grant policy for an access attempt is found, BEA AquaLogic Enterprise Security looks only for deny policies. When a single deny policy is found, it stops looking.

Configuring a Custom Entensions Plug-In

Note: This release of AquaLogic Enterprise Security includes an evaluation function example, in `BEA_HOME\ales26-ssm\java-ssm\examples\MyEvaluationFunction`.

To configure a custom extensions plug-in, you must implement the extensions plug-in, and then register it with the configured ASI Authorization and ASI Role Mapping providers.

To configure a custom extensions plug-in, perform the following steps:

1. Implement a custom extensions plug-in and use the Java class to create a JAR file.

   The `com.bea.security.providers.authorization.asi.InitializationShutdownFunction` class is in the *BEA_HOME*\ales26-ssm\<*ssm-type*>\lib\providers\ASIAuthorizer.jar. Include this file, and the *BEA_HOME*\ales26-ssm\<*ssm-type*>\lib\asi_classes.jar, in the classpath when compiling the custom attribute retriever.

2. Place the JAR file in the `/lib/providers` (`/lib/providers/wls/v9` for the WLS 9.x SSM) directory in the installation directory for the Security Service Module (SSM) on the machine on which the SSM is installed (either the WebLogic Server SSM or the Java SSM). For example, the default directory for the WebLogic Server SSM is *BEA_HOME*\ales26-ssm\wls-ssm.

3. In the left pane of the Administration Console, click the ASI Authorization provider configured for the SSM instance, select the Advanced tab in the right pane, type in the fully qualified name of your custom extensions plug-in in the Evaluation Functions field, and click Apply.

   For the WLS 9.x SSM, configure the Authorization and Role Mapping providers, and the custom extension, in the WebLogic Administration Console.

4. Repeat step 3 to register the extensions plug-in with the ASI Role Mapping provider.

5. In the left pane, click Deployment, select the Configuration tab, and deploy the configuration change to the SSM.

6. Restart the SSM process.

Custom Audit Plug-ins

The Log4j Audit Channel provider uses Log4j renderer classes that convert the associated audit event object into a simple string representation. However, you can write custom renderers that convert the audit event object to something other than the default string representation and register them as plug-ins using the Administration Console.

Refer to the following topics for information how to write and register custom audit plug-ins:

- "Using the Custom Audit Plug-in" on page 4-19
- "Audit Plug-in Renderer Class" on page 4-20

Using the Custom Audit Plug-in

To implement an audit plug-in interface, you must perform the following steps:

1. Refer to "Audit Plug-in Renderer Class" on page 4-20 for a description of the audit plug-in renderer class and write a Java class to implement a new renderer class.

2. Use the Java class to create a JAR file and place the JAR file in the `/lib/providers` directory (or, if you are using the WebLogic Server 9.x SSM, use `/lib/providers/wls/v9`) in the installation directory for the Security Service Module on the machine on which the Security Service Module is installed. For example, the default location of this directory for the WebLogic Server Security Service Module is:
   `C:\bea\ales26-wls-ssm\lib\providers`.

3. For instructions on using the Administration Console to register the audit plug-in for the desired Log4j Audit Channel provider, refer to Configuring a Log4j Audit Channel Provider in the Console Help.

Audit Plug-in Renderer Class

To write a plug-in renderer class, you must implement the `org.apache.log4j.or.ObjectRenderer` interface and then register the renderer class to the type of Audit Event class for which you want to use that renderer. For example,

```
weblogic.security.spi.MyAuditEvent=com.bea.security.providers.
       audit.MyAuditEventRenderer
```

For instructions on how to write a renderer for a custom object, see the Log4j documentation located at http://logging.apache.org/log4j/docs/documentation.html.

Table 4-7 lists and describes a sample `AuditEventRenderer` class.

Table 4-7 AuditEventRenderer Class Method

Method	Description

`public class MyAuditAtnEventRenderer implements org.apache.log4j.or.ObjectRenderer {`
`public String doRender(Object o) {`
`String eventStr = null;`
`if(o instanceof MyAuditEvent) {`
`MyAuditEvent event = (MyAuditEvent) o;`
`eventStr = event.getEventType()+" --`
`"+event.toString();`
`}`
`return eventStr;`
`}`
`}`
	In this sample, this method renders the AuditEvent object as a simple string. To render the Audit Event as something other than a simple string, modify this method to form your own string representation.

Database Authentication Plug-in

The Database Authentication extension is used by the Database Authentication provider to customize authentication features. The default database authentication extension (located in the `com.bea.security.providers.authentication.dbms.DefaultDBMSPluginImpl` package) is designed to authenticate the user against the policy database. This implementation uses a specific password hashing algorithm, namely, SHA1 and SHA-1. It also uses a special

format for the user name and the group name that is pertinent to the policy database. The hashing algorithm used is:

```
{Algorithm} + 4 byte Salt+passwordhash
```

The policy database uses name scope (for example, directory name) and a qualified name format to store the user and group. See the *BEA AquaLogic Enterprise Security Policy Managers Guide* for details.

If you are authenticating users against another database that uses a different password hashing algorithm and a different user/group name format, you may decide to implement your own plug-in by following the guidelines provided with the plug-in.

A custom database authentication plug-in must also extend the `DBMSPlugin` abstract class (located in the `com.bea.security.providers.authentication.dbms.DBMSPlugin` package). The `DBMSPlugin` abstract class implementation must include the methods described in Table 4-8.

To use your plug-in implementation, you need to deploy the plug-in class (or its JAR file) in the classpath of the Database Authentication provider (`/lib/providers` or, if you are using the WebLogic Server 9.x SSM, `/lib/providers/wls/v9`) and use the WebLogic Server Administration Console (if you are using the WebLogic Server 9.x SSM) or the ALES Administration Console (for other SSMs) to configure the Database Authentication provider to use the plug-in.

Table 4-8 lists and describes the methods provided by `the DBMSPlugin` abstract class.

Table 4-8 DBMSPlugin Methods

Method	Description
`public void initialize()`	This method is executed when the authorization provider is initialized on startup.
`public void shutdown()`	This method is executed when the authorization provider is shut down.

Table 4-8 DBMSPlugin Methods

Method	Description
`public boolean authenticate( String user, char[] password, char[] databasePassword, Map options)`	When the Database Authentication provider attempts to authenticate a user, the authenticate method is called on the plug-in. This method may be called in one following two scenarios. If the provider is configured with the SQL Query to retrieve password, the password (`databasePassword`) is retrieved from the database using this query and is provided to this method. This authenticate method must determine if the user provides the correct password (password) and return true, if authenticated, or false. The options map contains a TRUE value for key = "`QueryPassword`". If no SQL Query string is configured for retrieving the password, the Database Authentication provider assumes that the authentication plug-in retrieves the password and then authenticates the user. The options map contains values for these keys, "scope" and "connection", and a FALSE value for key = "QueryPassword". Also, databasePassword = null.
`public String formatUser(String user, Map options)`	This method is executed before any call to the database. The user string is the one passed into the login module. This method returns a formatted user name, which is later used as the input parameter in all the SQL queries to verify user, to retrieve password, and to retrieve groups. The options Map contains values for these keys, "scope" and "connection", and the configured string of the SQL query to verify user with key = "SQL_QUERY".
`public Vector formatGroups(String user, Vector groups, Map options)`	This method is executed after the call to retrieve groups from the database. A vector of strings containing the groups the user belongs to are passed in. Any formatting of group names that is required before inserting these into the Subject should be done and the resulting vector passed back. The options Map contains values for these keys, `scope` and `connection`, and the configured string of the SQL query to retrieve groups with key = "`SQL_QUERY`".
`public String unformatUser(String user, Map options)`	This method is executed after any call to the database that returns users. The user string is the one received from the database. This method returns a unformatted user name. The options Map contains values for these keys, "scope" and "connection".

Table 4-8 DBMSPlugin Methods

Method	Description
`public String formatGroup(String group, Map options)`	This method is executed after the call to retrieve groups from the database. Any formatting of group name that is required before inserting these into the Subject should be done and the resulting string passed back. The options Map contains values for these keys, scope and connection, and the configured string of the SQL query to retrieve group with key = "SQL_QUERY".
`public String unformatGroup(String group, Map options)`	This method is executed after any call to the database that returns a group membership for a user. The group string is the one received from the database. This method returns an unformatted group name. The options Map contains values for these keys, "scope" and "connection".
`public Vector unformatGroups(String user, Vector groups, Map options)`	This method is executed after any call to the database that returns a list of groups. The user is the unformatted user name and the vector of groups is the one received from the database. This method returns a vector of unformatted group names. The options Map contains values for these keys, "scope" and "connection".

The `options` object is a map containing optional information that the plug-in may want to use. The most common options of use and their keys for retrieval are:

- `key = scope`—the configured scope for the Database Authentication provider.
- `key = QueryPassword`—the `java.lang.Boolean` value that indicates whether the password SQL Query String was configured and executed. If it is false, then the password was not retrieved from the database. This key is only present for the authentication method.
- `key = connection`—an open JDBC `java.sql.Connection` object. Do not close this object; it is returned to the pool after authentication.

CHAPTER 5

Audit Events

The following topics are covered in this section:

What is an AuditEvent?

The `AuditEvent` interface provides a mechanism for passing additional audit information to Auditing providers during a `writeEvent` operation. This is the base interface that is extended by components in the Security Framework to compose specific audit event types. Extending this interface helps auditing providers determine the calling security component.

If you implement this interface and you expect to receive a ContextHandler argument from a caller, you can extend the `AuditContext` interface to provide more information.

Some of the sub-interfaces defined by the security SPI are listed in Table 5-1. Table 5-1 also indicates which sub-interfaces implement the `AuditContext` interface. These interfaces are documented in the *BEA AquaLogic Enterprise Security Provider SSPI API Reference*.

Table 5-1 Audit Events

Audit Event Name	Interface Class	Interfaces Implemented	
		AuditEvent	AuditContext
Authentication Audit Event	weblogic.security.spi.AuditAtnEvent	Yes	No
Authentication Audit Event V2	weblogic.security.spi.AuditAtnEventV2	Yes	Yes
Authorization Audit Event	weblogic.security.spi.AuditAtzEvent	Yes	Yes
Role Mapping Audit Event	weblogic.security.spi.AuditRoleEvent	Yes	Yes
Credential Mapping Audit Event	weblogic.security.spi.AuditCredentialMappingEvent	Yes	Yes
Management Audit Event	weblogic.security.spi.AuditMgmtEvent	Yes	No
Policy Audit Event	weblogic.security.spi.AuditPolicyEvent	Yes	No
Role Deployment Audit Event	weblogic.security.spi.AuditRoleDeploymentEvent	Yes	No
Provider Audit Record	com.bea.security.spi.ProviderAuditRecord	Yes	Yes

Table 5-2 lists WebLogic 9.x audit events.

Table 5-2 WebLogic 9.x Audit Events

Audit Event Name	Interface Class	Interfaces Implemented	
		Audit Event	Audit Context
Application Version Event	`weblogic.security.spi.AuditApplicationVersionEvent`	Yes	No
Authentication Audit Event	`weblogic.security.spi.AuditAtnEvent`	Yes	No
Authentication Audit Event V2	`weblogic.security.spi.AuditAtnEventV2`	Yes	Yes
Authorization Audit Event	`weblogic.security.spi.AuditAtzEvent`	Yes	Yes
CertPathBuilder Audit Event	`weblogic.security.spi.AuditCertPathBuilderEvent`	Yes	Yes
CertPathValidator Audit Event	`weblogic.security.spi.AuditCertPathValidatorEvent`	Yes	Yes
Configuration Audit Event	`weblogic.security.spi.AuditConfigurationEvent`	Yes	Yes
Credential Mapping Audit Event	`weblogic.security.spi.AuditCredentialMappingEvent`	Yes	Yes
Life Cycle Event	`weblogic.security.spi.AuditLifecycleEvent`	Yes	No
Audit Management Event	`weblogic.security.spi.AuditMgmtEvent`	Yes	No
Policy Audit Event	`weblogic.security.spi.AuditPolicyEvent`	Yes	No

Table 5-2 WebLogic 9.x Audit Events

Audit Event Name	Interface Class	Interfaces Implemented	
		Audit Event	Audit Context
Policy Consumer Audit Event	`weblogic.security.service.internal.PolicyConsumerAuditEvent`	AuditPolicyEvent	No
Provider Audit Record	`com.bea.security.spi.ProviderAuditRecord`	Yes	Yes
Role Consumer Audit Event	`weblogic.security.service.internal.RoleConsumerAuditEvent`	AuditRoleEvent	Yes
Role Deployment Audit Event	`weblogic.security.spi.AuditRoleDeploymentEvent`	Yes	No
Role Mapping Audit Event	`weblogic.security.spi.AuditRoleEvent`	Yes	Yes

Typically, the audit providers implement the `weblogic.security.spi.AuditChannel` interface and the `weblogic.security.spi.AuditProvider` interface, and post events

The `AuditEvents` that also implement the `AuditContext` interface can provide more information via a `ContextHandler`. The `ContextHandler` interface provides a way for an internal WebLogic container to pass additional information to a WebLogic Security Framework call, so that a security provider can obtain additional context information beyond what is provided by the arguments to a particular method. A `ContextHandler` is essentially a name/value list. The name/value list is also called a context element, and is represented by a `ContextElement` object.

What Events are Audited?

Depending on the interface that the `AuditEvent` has implemented, different information is audited. For all audit events, the `toString`() method is called on the event and that string is audited. Some audit events have a `ContextHandler`, such as the `AuditAtzEvent` and `AuditRoleEvent`, in which case the context is audited in addition to calling the `toString`() method on the `AuditEvent`. You can have many `ContextElements`, but each `NAME`/`VALUE` pair must be iterated over and audited.

The Log4j Audit Channel provider ships with Log4j renderers that are aware of these interfaces and know how to extract the appropriate audit information. You can change this behavior by writing custom renderers and updating the Custom Log4j Renderer Properties text box on the Advanced tab for the Log4j Auditor page in the Administration Console. A custom renderer is useful if only a particular subset of context elements are required or if the default style of audit events needs to be changed.

Each audit record has the following format:

```
2004-04-22 12:21:55,833 [Thread-27] SUCCESS ASI_AUDIT - My Custom Event -
Custom Event msg -- <attr1 = value1><attr2 = value2>
```

A custom renderer may require square brackets [] instead of angle brackets <>.

To be audited, you can select which severity the audit event must equal or be greater than; and you can select the types of `AuditEvents` by setting the Custom Audit Events attribute. If an `AuditEvent` implements or is an instance of any of the classes listed, then you can audit it. Only new custom events need to be listed here. The default events already exist and are controlled by selecting either: `DISABLED`, `WITH_CONTEXT`, or `WITHOUT_CONTEXT` on the Details tab for the Log4j Auditor page in the Administration Console. For a list of audit events, see "Audit Events" on page 5-1.

Note: Printing the entire context by enabling `WITH_CONTEXT` can be an expensive task and is proportional to the number of context elements contained in the `ContextHandler`.

All audit events generated through the Java API are called through the Provider Audit Records interface using the `AuditRecord` method. This includes `PolicyAdministrationEvent` and `ARMEAuthorizationEvent`. A `PolicyAdministrationEvent` is generated when a policy change is made through the Administration Console. An `ARMEAuthorizationEvent` is generated when the ASIAuthorizer makes a authorization request for a policy change.

All audit events can be `DISABLED` or `WITHOUT_CONTEXT`. For those that have context, you can select `WITH_CONTEXT`. The `AuditAtzEvents` have more options then all the other types, you can select the events to audit based on the following options:

- `DISABLE`—No auditing occurs.
- `WITHOUT_CONTEXT`—Audits what is in the event message.
- `WITH_REQUEST_CONTEXT`—Audits the event message plus the request context.
- `WITH_RESPONSE_CONTEXTS`—Audits the event message plus all the response contexts. Only contains the context that was populated with responses from the ASI Authorization provider. There can be many contexts returned for a single query and hence the `CONTEXTS`.

- WITH_ALL_CONTEXTS—Audits the event message plus all the contexts (request as well as response contexts).

Custom Audit Context Extensions

The Log4J Audit Channel provider is used to audit events that are generated by the Security Framework, the runtime API, or custom implementations based on the weblogic.security.spi.AuditEvent interface AuditEvent class.

Audit plug-ins can be used to audit with minimal awareness of the audit data formats being passed in by the calling Security Framework component. Additionally, Log4j plug-ins written or supplied by third parties can implement actions (such as paging security personal) based on audit severity/criteria you set in the Log4j Audit Channel provider Details tab in the Administration Console. Some general descriptions or suggestions for the information suitable for auditing by AuditEvent are as follows:

- Audit events are structured to have a two-tier model. There is a weblogic.security.spi.AuditEvent interface that defines the minimum requirements for an audit event. This interface includes type, severity, toString(), and, if there was an exception associated with the event, a reference to the exception.
- In addition to the core AuditEvent interface, several additional interfaces are defined to further elaborate on the audit types, and, for providers that need to retrieve audit properties that are specific to the audit type, interfaces exist that allow the providers to extract these values.
- A provider that is not reporting specific event properties can be coded to only recognize the core AuditEvent class and to use toString to output its representation of the event as a String.
- Audit providers that need to do other things (such as selectively log events based on event properties) must be specifically coded to the interfaces described so that they know how to extract these event values from the audit event.

Adding Application Context from the BLM API

The BLM API has been enhanced to allow you to send an Application Context to the auditing service.

An Audit Context is a name=value pair that contains additional audit data that is made available to the Audit provider. Like the Audit Context, the Application Context is also a name=value pair

data structure, and it contains additional application-specific audit data that is appended to the Audit Context when audit messages are written.

This additional information can be used by a custom Audit provider. Note, however, that the default Log4j Audit provider does not use this additional context.

When you create the Application Context, it is reused for each audit message associated with this BLM Context until it is overwritten by a call to set it, or you clear it.

The following BLM API methods have been added to provide for the Application Context:

- `BLMManager.create(java.util.Hashtable credentials, java.util.Hashtable appCtx)`. This method creates an instance of the BLMContextManager and initializes the BLMContextManager with an Application Context. The BLM then adds the Application Context data to all auditing messages associated with this BLM Context sent to the Audit provider.
- `BLMContextManager.setApplicationContext(Hashtable appCtx)`. This method replaces an existing application context with the new one provided. (You must have called `BLMManager.create(java.util.Hashtable credentials, java.util.Hashtable appCtx)` method prior to calling `setApplicationContext(Hashtable appCtx)`. All subsequent audit messages associated with this BLM Context have the Application Context added to them when they are sent to the Audit provider.
- `BLMContextManager.clearApplicationContext()`. This method clears the Application Context associated with this BLM Context so that it is no longer included with audit messages sent to the Audit provider.

Audit Event Interfaces and Audit Events

In the security provider interface package, WebLogic Security defines one top-level base interface (`AuditEvent`) with different derived interfaces that represent the different types of audit events.

The following sections describe when the security framework and security providers post some prominent types of audit events:

- AuditAtnEvent
- AuditAtzEvent
- AuditMgmtEvent
- AuditCredentialMappingEvent

- AuditPolicyEvent
- AuditRoleDeploymentEvent
- AuditRoleEvent
- ProviderAuditRecord

For a list of the events that are audited for the default Admin policy, see "Admin Policy Audit Events" on page 5-12.

AuditAtnEvent

Authentication audit events are posted by the security framework. Table 5-3 describes the conditions under which the event is posted and severity level of the event.

Table 5-3 Authentication Audit Events

Component	Description	Severity
Security Framework	Posted after successful authentication of a user.	Success
Security Framework	Posted after unsuccessful authentication (a LoginException thrown from JAAS login method). This LoginException can be thrown by either JAAS framework or by JAAS LoginModule of WebLogic Server authentication provider.	Failure
Security Framework	Posted after an identity assertion to an anonymous user.	Success
Security Framework	Posted after an unsuccessful identity assertion (IdentityAssertionException thrown from identity assertion method).	Failure
Security Framework	Posted after an unsuccessful identity assertion (IOException is thrown by identity assertion callback handler when retrieving username from callback).	Failure
Security Framework	Posted after an unsuccessful identity assertion (UnsupportedCallbackException is thrown by identity assertion callback handler when retrieving username from callback).	Failure

Table 5-3 Authentication Audit Events (Continued)

Component	Description	Severity
Security Framework	Posted after an unsuccessful identity assertion (when username returned from identity assertion callback handler is null or zero length).	Failure
Security Framework	Posted after a successful identity assertion.	Success
Security Framework	Posted after an unsuccessful identity assertion.	Failure
Security Framework	Posted after a successful impersonate identity (anonymous identity).	Success
Security Framework	Posted after a successful impersonate identity.	Success
Security Framework	Posted after an unsuccessful impersonate identity.	Failure
Security Framework	Posted after a failure of principal validation.	Failure
Security Framework	A user has been locked by the user lockout manager.	Failure
Security Framework	A user has been unlocked	Success
Security Framework	A user lockout has expired	Success

AuditAtzEvent

Authorization audit events are posted by the security framework. Table 5-4 describes the conditions under which the events are posted and severity level of the event.

Table 5-4 Authorization Audit Events

Component	Description	Severity
Security Framework	Posted if access is not allowed to resource (exception thrown by authorization provider).	Failure
Security Framework	Posted if access is allowed to resource.	Success

AuditCredentialMappingEvent

Credential Mapping audit events are posted by the security framework. Table 5-5. describes the condition under which the events are posted and severity level of the event.

Table 5-5 Credential Mapping Audit Events

Component	Description	Severity
Security Framework	Posted after each successful get of credentials.	Success

AuditMgmtEvent

Management audit events are not currently posted by either the security framework or by the supplied providers.

AuditPolicyEvent

AuditPolicyEvent are posted by the security framework and the WebLogic Authorization provider. The security framework posts audit policy events when policies are deployed to or undeployed from an authorization provider. The WebLogic Server authorization provider posts audit policy events when creating, deleting, or updating policies. Table 5-6 describes the conditions under which the events are posted and lists the event severity level.

Table 5-6 Audit Policy Events

Component	Description	Severity
Security Framework	Posted after successful deploy of policy.	Success
Security Framework	Posted after unsuccessful deploy of policy.	Failure
Security Framework	Posted after successful undeploy of policy.	Success
Security Framework	Posted after an unsuccessful undeploy of policy.	Failure

Table 5-6 Audit Policy Events (Continued)

Component	Description	Severity
WebLogic Authorization Provider	Posted after the following events occur: • A successful create of policy from console • An unsuccessful create of policy from console (various exceptions) • A successful remove of policy from console • An unsuccessful remove of policy from console (various exceptions) • A successful update of policy from console • An unsuccessful update of policy from console (various exceptions)	Success
WebLogic Authorization Provider	Application deletion of security policies has succeeded.	Success
WebLogic Authorization Provider	Application deletion of security policies has failed.	Failure

AuditRoleDeploymentEvent

The security framework posts audit role deployment events when roles are deployed to or undeployed from a role mapping provider. Table 5-7 describes the conditions under which the events are posted and lists the event severity level.

Table 5-7 Audit Role Deployment Events

Component	Description	Severity
Security Framework	Posted after each successful role deployment to a role mapping provider.	Success
Security Framework	Posted after each unsuccessful role deployment to a role mapping provider.	Failure
Security Framework	Posted after each successful role undeployment from a role mapping provider.	Success

Table 5-7 Audit Role Deployment Events (Continued)

Component	Description	Severity
Security Framework	Posted after each unsuccessful role undeployment from a role mapping provider.	Failure
Security Framework	Application deletion of security roles to a Role Mapping provider has succeeded.	Success
Security Framework	Application deletion of security roles to a Role Mapping provider has failed.	Failure

AuditRoleEvent

The WebLogic Role Mapping provider posts audit role events when roles are created, deleted, or updated. Table 5-8 describes the conditions under which the events are posted and lists the event severity level.

Table 5-8 Audit Role Events

Component	Description	Severity
WebLogic Role Mapping Provider	Posted after the following events occur: • A successful create of role from console • An unsuccessful create of role from console (various exceptions) • A successful remove of role from console • An unsuccessful remove of role from console (various exceptions) • A successful update of role from console • An unsuccessful update of role from console (various exceptions)	Success

Admin Policy Audit Events

Table 5-9 lists and describes the administration policy events that are audited.

Table 5-9 Admin Policy Audit Events

Policy Element	Action	Type	Event Description
Declaration/Attribute	create	declaration, value	Create a new attribute declaration.
	delete	declaration, value	Delete an attribute declaration.
	rename	action group, new_name	Rename an attribute declaration.
	modify	declaration, value, new value	Modify an attribute declaration.
Declaration/Constant	create	declaration, value	Create a new constant.
	delete	declaration, value	Delete a constant.
	rename	action group, new_name	Rename a constant.
	modify	declaration, value, new_value	Modify a constant.
Declaration/Enumeration	create	declaration, value	Create a new enumeration.
	delete	declaration, value	Delete an enumeration.
	rename	action group, new_name	Rename an enumeration.
	modify	declaration, value, new_value	Modify an enumeration.
Declaration/Evaluation Function	create	declaration	Create an evaluation function.
	delete	declaration	Delete an evaluation function.
	rename	action group, new_name	Rename an evaluation function.
	modify	declaration, value, new value	Modify an evaluation function.

Table 5-9 Admin Policy Audit Events (Continued)

Policy Element	Action	Type	Event Description
Identity/Directory/Instance	create	directory	Create a directory.
	delete	directory	Delete a directory.
	cascade Delete	directory	Delete a directory and all its users.
	rename	directory, new_name	Rename a directory.
Identity/Directory/AttributeMapping/Single	create	attribute, default_value, directory	Add a scalar attribute to a directory attribute schema.
	delete	attribute, default_value, directory	Delete a scalar attribute from a directory attribute schema.
	modify	attribute, default_value, directory, new_default_value	Modify a scalar attribute in a directory attribute schema.
Identity/Directory/AttributeMapping/List	create	attribute, default_value, directory	Add a vector attribute to a directory attribute schema.
	delete	attribute, default_value directory	Delete a vector attribute from a directory attribute schema.
	modify	attribute, default_value, directory, new_default_value	Modify a vector attribute in a directory attribute schema.
Identity/Subject/User	create	subject_name	Create a new user.
	copy	subject_name, new_subject_name	Copy a user.
	delete	subject_name	Delete a user.
	rename	subject_name, new_subject_name	Rename a user.

Table 5-9 Admin Policy Audit Events (Continued)

Policy Element	Action	Type	Event Description
Identity/Subject/ Group	create	subject_name	Create a new group.
	delete	subject_name	Delete a group.
	rename	subject_name, new_subject_name	Rename a group.
	addMember	subject_name, member_subject	Add a member to a group.
	removeMember	subject_name, member_subject	Remove a member from a group.
Identity/Subject/ Attribute Assignment	create	attribute, value, subject_name	Set a value to a currently unset scalar subject attribute.
	delete	attribute, value, subject_name	Unset a currently set scalar subject attribute.
	modify	attribute, value, subject_name, new_value	Modify the value of a currently set scalar subject attribute.
Identity/Subject/ Password	modify	subject_name	Modify the user password. The “subject_name” attribute contains the name of the user with which the password is associated.
Resource/Instance	create	resource, resource_type	Create a new resource.
	delete	resource	Delete a resource.
	rename	resource, new_name	Rename a resource.

Table 5-9 Admin Policy Audit Events (Continued)

Policy Element	Action	Type	Event Description
Resource/Attribute Assignment	create	attribute, resource, value	Set a value to a currently unset scalar resource attribute.
	delete	attribute, resource, value	Unset a currently set scalar resource attribute.
	modify	attribute, resource, value, new_value	Modify the value of a currently set scalar resource attribute.
Policy/Rule/Grant	create	action, resource, subject_name, constraint	Create a new grant policy. The “action”, “resource”, and “subject_name” attributes are lists.
	delete	action, resource, subject_name, constraint	Delete a grant policy. The “action”, “resource”, and “subject_name” attributes are lists.
	modify	action, resource, subject_name, constraint	Modify a grant policy. The “action”, “resource”, and “subject_name” attributes are lists.
Policy/Rule/Deny	create	action, resource, subject_name, constraint	Create a new deny policy. The “action”, “resource”, and “subject_name” attributes are lists.
	delete	action, resource, subject_name, constraint	Delete a deny policy. The “action”, “resource”, and “subject_name” attributes are lists.
	modify	action, resource, subject_name, constraint	Modify a deny policy. The “action”, “resource”, and “subject_name” attributes are lists.

Table 5-9 Admin Policy Audit Events (Continued)

Policy Element	Action	Type	Event Description
Policy/Rule/Delegate	create	action, resource, subject_name, delegator, constraint	Create a new delegate policy. The “action”, “resource”, and “subject_name” attributes are lists.
	delete	action, resource, subject_name, delegator, constraint	Delete a delegate policy. The “action”, “resource”, and “subject_name” attributes are lists.
	modify	action, resource, subject_name, constraint	Modify a delegate policy. The “action”, “resource”, and “subject_name” attributes are lists.
Policy/Action/Role/ Instance	create	action	Create a new role.
	delete	action	Delete a role.
	rename	action, new_name	Rename a role.
Policy/Action/ Privilege/Instance	create	action	Create a privilege.
	delete	action	Delete a privilege.
	rename	action, new_name	Rename a privilege.
Policy/Action/ Privilege/Group	create	action_group	Create a privilege group.
	delete	action_group	Delete a privilege group.
	rename	action_group, new_name	Rename a privilege group.
	addMemb er	action_group, action	Add a privilege to a privilege group.
	remove Member	action_group, action	Remove a privilege from a privilege group.

Table 5-9 Admin Policy Audit Events (Continued)

Policy Element	Action	Type	Event Description
Policy/Analysis/ Inquiry Query	create	title, owner, effect_type, subjects, actions, resources, delegator	Create a new policy query.
	modify	title, owner, effect_type, subjects, actions, resources, delegator	Modify a policy query.
	Read		Read policy inquiry.
Policy/Repository	deploy Update	resource, directory	Deploy a policy update. The "resource" is the distribution node; all nodes below it may be affected. This check is made for each chosen distribution point
	deploy Structural Change	deleted_directories, deployed_engines, deleted_engines, deleted_bindings, deleted_applications	Deploy a structural change.
Infrastructure/Engines	create	engine	Create a new SSM.
	delete	engine	Delete an SSM.
	rename	engine, new_name	Rename an SSM.
	bind	engine, resource	Bind a resource to an SSM.
	unbind	engine, resource	Unbind a resource from an SSM.

Table 5-9 Admin Policy Audit Events (Continued)

Policy Element	Action	Type	Event Description
Infrastructure/Engines /SCM	create	engine	Create an SCM.
	delete	engine	Delete an SCM.
	rename	engine, new_name	Rename an SCM.
	bind	engine, resource	Bind an SSM to an SCM. A "resource" contains the name of the SSM.
	unbind	engine, resource	Unbind an SSM from an SCM. A "resource" contains the name of the SSM.

ProviderAuditRecord

This interface is defined in the AquaLogic Enterprise Security Provider SSPI package and provides an extended version of the AuditEvent. Refer to BEA AquaLogic Enterprise Security Provider SSPI API Reference for full documentation of this interface. The SSPI package includes ALES-specific interfaces, classes, and exceptions for developing ALES security providers.

Providers written to work in both the WLS Security Framework and the ALES Security Providers environments must handle both WebLogic audit records and extended AuditEvents. Examples of extended AuditEvents are subinterfaces and implementations of `ProviderAuditRecord interface`.

In the providers, the `instanceof` operator can be used to distinguish between the WLS Security Framework interfaces and ALES Enterprise Security Provider interfaces. For example:

```
if ( myauditrecord instanceof com.bea.security.spi.ProviderAuditRecord){
// This is a ALES audit record that uses the enhanced SSPI.
} else {
// This is a WLS audit record. You must test further for more object
// types and handle them explicitly.
}
```

A simple audit provider can use the `toString()` method to render the audit record as a string; the provider does not require specific knowledge of the audit record type. A more complex auditing provider that tracks events by many keys and needs to distinguish messages by various types and attributes requires a data-driven method of event introspection. The complex auditing provider can get an enumeration of `com.bea.security.spi.NameValueTypes` that contain this audit record's name fields using the `ProviderAuditRecord.getEnumeration()` and `ProviderAuditRecord.getDeepEnumeration()` methods.

Additionally, the `ProviderAuditRecord` interface can associate an application context with an audit event. This allows the auditing provider to select some context elements to audit when events occur. For example, when an audit event occurs, you may choose to audit the number of concurrent sessions, the time the user logged on, or some other application-specific value propagated by the application context.

The following code fragment shows how a custom provider can access the context added by a client application. The modification is primarily in the `writeEvent` method, as shown.

```
public void writeEvent(AuditEvent event){

    // write the event out to the sample auditor's

    // log file using the event's "toString" method.

    // followed by the string version of the application

    // context name value pairs.

    ProviderAuditRecord par = (ProviderAuditRecord) event;

    ContextHandler ch = par.getContext();

    String[] names = ch.getNames();

    StringBuffer ctxReader = new StringBuffer();

    for ( int i=0; i < names.length; i++ ) {

         String value = (String) ch.getValue(names[i]);

         ctxReader.append(names[i]).append("=").append(value);

    }

    log.println(event + "Context = " + ctxReader.toString());

}
```

Other Audit Events

This section describes some of the other AuditEvents used by the security framework and security providers.

AuditApplicationVersionEvent

Table 5-10 describes the conditions under which the event is posted and severity level of the event.

Table 5-10 Application Version Audit Event

Component	Description	Severity
Security Framework	Authorization Manager application version creation has succeeded or failed.	Success
Security Framework	Authorization Manager application version deletion has succeeded or failed.	Failure
Security Framework	Authorization Manager non-versioned application deletion has succeeded or failed.	Success
Security Framework	Role Manager application version creation has succeeded or failed.	Failure
Security Framework	Role Manager application version deletion has succeeded or failed.	Failure
Security Framework	Role Manager non-versioned application deletion has succeeded or failed.	Failure
Security Framework	Credential Manager application version creation has succeeded or failed.	Failure
Security Framework	Credential Manager application version deletion has succeeded or failed.	Success
Security Framework	Credential Manager non-versioned application deletion has succeeded or failed.	Failure

AuditCertPathBuilderEvent and AuditCertPathValidatorEvent

These events are posted by the CertPathBuilder providers.

AuditLifecycleEvent

Life cycle audit events are posted by the WLS framework, as follows:

Table 5-11 Audit Life Cycle Events

Component	Description	Severity
Security Framework	After the auditing service in the framework is started.	Success
Security Framework	Before the auditing service in the framework is stopped.	Success

Additional Audit Event Information

Some implementations of the `AuditEvent` interface contain additional information that can be accessed by the providers and security framework. All interfaces that extend the `weblogic.security.spi.AuditEvent` interface or all the implementations of that interface have the following information available:

- The type of the event represented as string
- If available associated failure exception
- The severity level associated with the event: INFORMATION, WARNING, ERROR, SUCCESS, FAILURE

The following sections provide additional information about specific audit events:

- "Authentication - AuditAtnEvent" on page 5-23
- "Authorization - AuditAtzEvent" on page 5-23
- "AuditCredentialMappingEvent" on page 5-24
- "Policy Events - AuditPolicyEvent" on page 5-24
- "Role Mapping - AuditRoleEvent" on page 5-25

- “AuditRoleDeploymentEvent” on page 5-26
- “Predicate Events - AuditPredicateEvent” on page 5-27

Authentication - AuditAtnEvent

The `AuditAtnEvent` interface provides an interface for audit providers to determine the instance types of the extended authentication event type objects. Table 5-12 describes the event properties.

Table 5-12 Authentication - AuditAtnEvent Event Type Property Values

Event Type Property	Description
AUTHENTICATE	Represents the "simple authentication" authentication type.
USERLOCKED	Indicates that a user was locked because of a series of failed login attempts.
USERLOCKOUTEXPIRED	Indicates that a lock on a user has expired.
USERUNLOCKED	Indicates that a lock on a user was cleared.
ASSERTIDENTITY	Represents the identity assertion authentication token type.
IMPERSONATEIDENITY	Represents the impersonate identity authentication type.
VALIDATEIDENTITY	Represents the validate identity authentication type.

When this event is generated, the following information associated with this `AuditAtnEvent` is available:

- The username associated with this `AuditAtnEvent`; that is, the username of the person who is attempting authentication.
- The event type associated with this `AuditAtnEvent`

There are both pre- and post-authorization access control checks; each of which generates pre- and post-operation audit write events.

Authorization - AuditAtzEvent

The `AuditAtzEvent` event interface is used to report events that result when access is allowed to a resource. The Audit Channel provider is called both prior to and after the authorization operation.

This event has the following information available:

- The name of the resource
- The name of the subject
- The `ContextHandler` object. The resource container that handles the type of resource requested (for example, in WebLogic Server, the EJB container receives the request for an EJB resource) constructs a `ContextHandler` object that may be used by an authorization provider for making the access decisions.

AuditCredentialMappingEvent

The `AuditCredentialMappingEvent` interface is used to post credential mapping audit events when credentials for a WebLogic Server user are requested, or when credentials for any subject are requested. The following information is available with this event:

- The resource for which the credentials are requested causing this event
- The requestor subject for the operation
- The subject of the initiator of the operation
- The string representation of the initiator of the operation
- The credential types requested in the operation
- Object array of credentials returned by the operation
- The ContextHandler object.

Policy Events - AuditPolicyEvent

The AuditPolicyEvent interface determines the instance types of extended Authorization event type objects. Table 5-13 describes the event subtypes.

Table 5-13 Policy Event- AuditPolicyEvent

Event Subtype	Description
AuditPolicyDeployEvent	Indicates that a policy deployment event occurred.
AuditPolicyUndeployEvent	Indicates that a policy undeployment event occurred.
AuditPolicyDeleteAppEvent	Indicates that an application deleted policy.
AuditStartPolicyDeployEvent	Indicates start of the policy deployment.
AuditEndPolicyDeployEvent	Indicates end of the policy deployment.

All AuditPolicyEvents have:

- The subject associated with the policy related operation
- The resource associated with the operation

Policy Deployment - AuditPolicyDeployEvent

The `AuditPolicyDeployEvent` is type of a `AuditPolicyEvent` used when the Authorization Manager `deployPolicy` method is called. When this event is generated, the following information is available:

- Role names associated with this policy deploy event

Policy Undeployment - AuditPolicyUndeployEvent

The `AuditPolicyUndeployEvent` is type of a `AuditPolicyEvent` used when the Authorization Manager `undeployPolicy` method is called. When this event is generated, the same information as listed for the AuditPolicyEvent is available.

Role Mapping - AuditRoleEvent

The `AuditRoleEvent` event provides an interface for auditing providers to determine the instance types of extended Role Mapping event type objects.

When an AuditRoleEvent is generated, the following information is available:

- The Subject that is attempting to access the resource associated with this AuditRoleEvent.
- The resource attempting to be accessed by the subject associated with this AuditRoleEvent.
- Role objects that are being manipulated by the action being audited.
- The ContextHandler object, as described in "ContextHandler Object" on page 5-28.

AuditRoleDeploymentEvent

The AuditRoleDeploymentEvent provides an interface for auditing providers to determine the instance types of extended Role deployment event type objects. Table 5-14 describes the event subtypes.

Table 5-14 Role Deployment - AuditRoleDeploymentEvent

Event Subtype	Description
AuditRoleDeployEvent	Indicates that a role mapping deployment event occurred.
AuditRoleUndeployEvent	Indicates that a role mapping undeployment event occurred.
AuditRoleDeleteAppEvent	Indicates that an application deleted role.
AuditStartRoleDeployEvent	Indicates that the role deployment has begun.
AuditEndRoleDeployEvent	Indicates that the role deployment has ended.

Role Deployment - AuditRoleDeployEvent

The AuditRoleDeployEvent event is used by the role mapping service to determine the instance types of extended Role Mapping deployment event type objects. In addition to the information listed by the AuditEvent interface, this event also has following additional information:

- Resource for which the role is being added.
- The name of the role being added.
- The subject that is attempting to deploy a role.
- The user and group names for the role being added.

Role Undeployment - AuditRoleUndeployEvent

The `AuditRoleUndeployEvent` event is used by the role mapping service to determine the instance types of extended Role Mapping undeployment event type objects. In addition to the information listed by the `AuditEvent` interface, this event also has the following additional information:

- Resource for which the role is being undeployed.
- The name of the role being undeployed.
- The sublect that is attempting to undeploy the role.

Role Delete App Event

In addition to the information listed by the `AuditEvent` interface, this event also provides the following additional information:

- Resource for which the role is being deleted.
- The subject that is attempting to delete the role.

Role Deployment Start and End Events

In addition to the information listed by the `AuditEvent` interface, this event also provides the following additional information:

- Resource for which the role deployment is being started or finished.
- The subject attempting to start or finish the deployment.

Predicate Events - AuditPredicateEvent

The `AuditPredicateEvent` event is used by Auditing providers to determine the instance type of extended predicate event type objects. A predicate event occurs when a policy expression

is either registered or unregistered in the Administration Console. Table 5-15 describes the event subtypes.

Table 5-15 Predicate Events - AuditPredicateEvent

Event Subtype	Description
REGISTER	Occurs when a policy expression is registered.
UNREGISTER	Occurs when a policy expression is registered.

ContextHandler Object

A `ContextHandler` is a class that obtains additional context and container-specific information from the resource container, and provides that information to security providers making access or role mapping decisions. The `ContextHandler` interface provides a way for an application or container to pass additional information to a Security Framework call, so that a security provider can obtain contextual information beyond what is provided by the arguments to a particular method. A `ContextHandler` is essentially a name/value list and as such, it requires a security provider to know what names to look for. In other words, use of a `ContextHandler` requires close cooperation between the resource container and the security provider. Each name/value pair in a `ContextHandler` is known as a context element, and is represented by a `ContextHandler` object.

A context handler is an object that is included with some event types that allows an audit provider to extract other information about the state of the application server at the time of the audit event. The audit provider may log this other contextual information as a way to elaborate on the event and provide other useful information about the causes of the event.

ALES Policy Administration Messages

When AquaLogic Enterprise Security policy is modified or deployed using the AquaLogic Enterprise Security Administration console or BLM Java API, informative messages are audited. The following information is available in these messages:

- Detailed information regarding the change being made
- The application context associated with the BLM Session being used.

The exception that occurred (if any) while attempting to carry out this action. Typically, there will only be an exception if the severity is error or failure.

Using Custom Audit Providers

You can use a custom auditing provider instead of the Log4j Audit Channel provider. For a custom auditing provider to be configurable through the Administration Console, the MBean JAR file for the provider must be installed into the `BEA_HOME..//lib/providers` directory on both the machine on which the Administration Application is installed and on the machine on which the Security Service Module is installed. For complete instructions for configuring a custom security provider, see Configuring a Custom Security Provider in the Console Help.

CHAPTER 6

BLM Configuration API Security Providers Reference

This section provides a reference for the security provider attributes, and their default values.

Because default security provider attributes are not stored in the database, the BLM configuration API cannot discover the security provider attribute names or default values. Further, since there is an inheritance model with the provider attributes, if a given provider extends another, all the attributes from the parent are available as well.

You use these attribute names and default values with the BLM configuration API classes. For example, the `SSMConfigurationManager.createProviderConfiguration()` method has a parameter for `mgmtinterface`, which is the full name of the management interface associated with this provider. The `mgmtinterface` values are documented in this section.

As another example, the `SSMProviderManager.getPropertyReport()` method returns a report on a provider's properties collection. However, attributes that have not been explicitly set use their default values, which are not returned in the array of `SSMProviderConfigElement` objects. The default attribute values are documented in this section.

Note: All information entered through the BLM Configuration API is string based.

Each of the following sections includes a table that lists the attributes supported by each security provider. Each table includes a List column that designates whether the `getValue/setValue` or `getValueList/setValueList` methods should be used with each attribute.

ActiveDirectoryAuthenticator

The ActiveDirectoryAuthenticator extends com.bea.security.providers.authentication.LDAPAuthenticator. Table 6-1 describes the attributes supported by this provider.

Table 6-1 ActiveDirectoryAuthenticator

Attribute Name	Default Value	Description	List
mgmtinterface	com.bea.security.providers.authentication.ActiveDirectoryAuthenticator		N
UserNameAttribute	"cn"	The attribute of the LDAP user object the specifies the name of the user.	N
UserBaseDN	"ou=WLSMEMBERS,dc=example,dc=com"	The base distinguished name (DN) of the tree in the LDAP directory that contains users.	N
UserFromNameFilter	"(&;(cn=%u)(objectclass=user))"	LDAP search filter for finding a user given the name of the user. If the attribute (user name attribute and user object class) is not specified (that is, if the attribute is null or empty), a default search filter is created based on the user schema.	N
UserObjectClass	"user"	The LDAP object class that stores users.	N
GroupBaseDN	"ou=WLSGROUPS,dc=example,dc=com"	The base distinguished name (DN) of the tree in the LDAP directory that contains groups.	N
GroupFromNameFilter	"(&;(cn=%g)(objectclass=group))"	LDAP search filter for finding a group given the name of the group. If the attribute is not specified (that is, if the attribute is null or empty), a default search filter is created based on the group schema.	N

Table 6-1 ActiveDirectoryAuthenticator (Continued)

Attribute Name	Default Value	Description	List
StaticGroupDNsfromMemberDNFilter	"(&(member=%M)(objectclass=group))"	LDAP search filter that, given the distinguished name (DN) of a member of a group, returns the DNs of the static LDAP groups that contain that member.	N
StaticGroupObjectClass	"group"	The name of the LDAP object class that stores static groups.	N
StaticMemberDNAttribute	"member"	The attribute of the LDAP static group object that specifies the distinguished names (DNs) of the members of the group.	N
UseTokenGroupsForGroupMembershipLookup	"false"	Boolean value that indicates whether to use TokenGroups attribute lookup algorithm instead of the standard recursive group membership lookup algorithm.	N
EnableSIDtoGroupLookupCaching	False	Indicates whether SID to group name lookup results are cached. This attribute is only used if the token group membership lookup algorithm is enabled (see UseTokenGroupsForGroupMembershipLookup).	N
MaxSIDToGroupLookupsInCache	"500"	The maximum size of the LRU cache for holding SID to group lookups if caching of SID to group name mappings is enabled and if the tokenGroups group membership lookup is enabled. The default is 500.	N

ALESIdentityAsserter

ALESIdentityAsserter extends com.bea.security.providers.authentication.alesidentity. Table 6-2 describes the attributes supported by this security provider.

Table 6-2 ALESIdentityAsserter

Attribute Name	Default Value	Description	List
mgmtinterface	"com.bea.security.providers.authentication.alesidentity.ALESIdentityAsserter		N
ActiveTypes	"{"ALESIdentityAssertion"}"		Y
Base64Decoding Required	"false"	Specifies whether the request header value or cookie value must be decoded using Base64 before it is sent to the Identity Assertion provider. This box is checked by default for purposes of backward compatibility; however, most Identity Assertion providers do not require this decoding.	N
TrustedCAKeystore	"{HOME}/ssl/demoProviderTrust.jks"	The location of the Trusted Keystore stored in the TrustedCAKeystoreType keystore format. {HOME} will be replaced with the Security Service Module (SSM) instance directory at runtime. This attribute is determined by the value of instance.home in SSM.properties located in the /config directory of the SSM instance. If SSM.properties cannot be located, then the system property wles.ssmws.instance.home is checked. For the Web Services SSM, this attribute is automatically set to the Web Services SSM instance home. If DEFAULT is specified, then the java.home env variable is used to locate the cacerts keystore normally located at JAVA_HOME/lib/security/cacerts.	

Table 6-2 ALESIdentityAsserter

Attribute Name	Default Value	Description	List
TrustedKeystore	"{HOME}/ssl/demoProviderTrust.jks"	The Location of the Trusted Keystore stored in the TrustedKeystoreType keystore format. {HOME} will be replaced with the SSM instance directory at runtime. This attribute is determined by the value of instance.home in SSM.properties located in the /config directory of the SSM instance. If SSM.properties cannot be located, then the system property wles.ssmws.instance.home is checked. For the Web Services SSM, this attribute is automatically set to the Web Services SSM instance home.	N
TrustedCAKeystoreType	"JKS"	The type of keystore to which the trustedCAKeystore is configured.	N
TrustedKeystoreType	"JKS"	The type of keystore to which the trustedKeystore is configured.	N
TrustedCertAlias	"demo_provider_trust"	The Cert Alias to be used to verify the ALES Identity Assertion.	N
TrustedCertAliasPasswd	"password"	The password to use for the Cert Alias specified to retrieve the private key from the keystore.	N

ALESIdentityCredentialMapper

ALESIdentityCredentialMapper extends weblogic.management.security.credentials.CredentialMapper. Table 6-3 describes the attributes supported by this security provider.

Table 6-3 ALEsIdentityCredential Mapper

Attribute Name	Default Value	Description	List
mgmtinterface	"com.bea.security.providers.credentials.alesidentity.ALESIdentityCredentialMapper"		N
TrustedKeystore	"{HOME}/ssl/demoProviderTrust.jks"	The Keystore to be used to get the Certificate chain to sign the ALES Identity Assertion with. {HOME} will be replaced with the SSM instance directory at runtime. This attribute is determined by the value of instance.home in SSM.properties located in the /config directory of the SSM instance. If SSM.properties cannot be located, then the system property wles.ssmws.instance.home is checked. For the Web Services SSM, this attribute is automatically set to the Web Services SSM instance home.	N
TrustedKeystoreType	"JKS"	The TYPE of keystore that is specified in the TrustedKeystore.	N
TrustedCertAlias	demo_provider_trust	The Cert Alias to be used to sign the ALES Identity Assertion.	N
TrustedCertAliasPasswd	"password"	The Password to use for the Cert Alias specified to retrieve the private key from the keystore.	N

AsiAdjudicator

AsiAdjudicator extends weblogic.management.security.authorization.Adjudicator. Table 6-4 describes the attributes supported by this security provider.

Table 6-4 AsiAdjudicator

Attribute Name	Default Value	Description	List
mgmtinterface	"com.bea.security.providers.authorization.ASIAdjudicator"		N
RequireUnanimousPermit	"true"	Requires all authorization providers to vote PERMIT in order for the adjudication provider to vote PERMIT. If the attribute is set to disabled, ABSTAIN votes are ignored.	N

AsiAuthorizationProvider

ASIAuthorizationProvider extends com.bea.security.providers.authorization.asi Table 6-5 describes the attributes supported by this security provider.

Table 6-5 AsiAuthorizationProvider

Attribute Name	Default Value	Description	List
mgmtinterface	"com.bea.security.providers.authorization.asi.ASIAuthorizationProvider"		N
IgnoreNonASIRoles	"false"	Specifies if the provider should ignore roles generated by role mapping providers other than the ASI Role Mapping provider.	N
AccessAllowedCaching	"true"	When enabled results from authorization queries are cached providing significantly improved performance for applications which make repetitive queries.	N

ASIAuthorizer

ASIAuthorizer extends weblogic.management.security.Provider. Table 6-6 describes the attributes supported by this security provider.

Table 6-6 AsiAuthorizer

Attribute Name	Default Value	Description	List
mgmtinterface	"com.bea.security.providers.authorization.asi.ASIAuthorizer"		N
AdvancedConfigurationProperties		Specifies additional advanced configuration parameters.	Y
Directory	"asi"	Specifies the identity directory to use when performing the authorization or role mapping.	N
PreLoadAttributes	"adaptive-private"	Determines whether or not the provider loads ContextHandler data before starting to evaluate policy or waits for a callback to ask for specific items. Pre-loading attributes can dramatically improve performance in policies that use contextual attributes.	N
SessionEvictionCapacity	500	The number of authorization and role mapping sessions to actively maintain. Once the limit is reached, old sessions are dropped and automatically re-established when needed.	N
SessionEvictionPercentage	10	The percentage of authorization and role mapping sessions to drop when the eviction capacity is reached.	N
ApplicationDeploymentParent	"//app/policy"	Specifies the root of the resource tree for this SSM.	N

Table 6-6 AsiAuthorizer (Continued)

Attribute Name	Default Value	Description	List
SharedResourcesParent	“shared”	Specifies the root on the shared resource tree for this SSM. This item may be relative to the value specified by Application Deployment Parent on the Details tab.	N
ResourceConverters		Specifies the types of resources supported by these providers. The value is a list of fully-qualified Java class names. These classes should implement the ResourceConverter interface. This product includes resource converters for the standard WebLogic resource types.	Y
InstantiateWeblogicResourceConverters	“true”	Instantiate Resource Converters for all default WebLogic resource types.	N
AttributeRetrievers		Specifies plugins used to retrieve attribute values from complex data objects. These classes should implement the AttributeRetriever interface.	Y
EvaluationFunctions		Specifies plugins used to perform complex evaluations. These classes should implement the EvaluationFunction interface.	Y
AttributeConverters		Specifies the plugins to use when converting native Java types into the required string representation used when evaluating policy. If a converter is not registered for a given type, then the toString() method is used by default.	Y
AnonymousSubjectName	“anonymous”	The name to use when performing queries for an unauthenticated user.	N

Table 6-6 AsiAuthorizer (Continued)

Attribute Name	Default Value	Description	List
"UseUserAttributes"	"true"	Specifies whether or not user attributes are used in evaluation of policy.	N
ActivateOnStartUp	"true"	Determines whether or not the authorization and role mapping providers process policy requests from cached policy before contacting the Policy Distributor for a policy update.	N
SessionExpirationSec	"60"	The duration for which to cache session data, in seconds.	N
SubjectDataCacheExpirationSec	"60"	The duration for which to cache subject data, in seconds.	N

ASIRoleMapperProvider

ASIRoleMapperProvider extends weblogic.management.security.authorization.RoleMapper. Table 6-7 describes the attributes supported by this security provider.

Table 6-7 ASIRoleMapperProvider

Attribute Name	Default Value	Description	List
mgmtinterface	"com.bea.security.providers.authorization.asi.ASIRoleMapper"		N

Table 6-7 ASIRoleMapperProvider (Continued)

Attribute Name	Default Value	Description	List
LazyRoleProvider	"true"	When enabled the role provider will delay calculation of role membership until the result is inspected. Leaving this attribute set to `true` provides significant performance improvements when used in conjunction with the ASI Authorization provider.	N
GetRolesCaching	"true"	When enabled results from role mapping queries are cached providing significantly improved performance for applications which make repetitive queries.	N

DatabaseAuthenticator

DatabaseAuthenticatorextends com.bea.security.providers.authentication.dbms.DBMSAuthenticator.

DatabaseCredentialMapper

DatabaseCredentialMapper extends weblogic.management.security.credentials.CredentialMapper. Table 6-8 describes the attributes supported by this security provider.

Table 6-8 DatabaseCredentialMapper

Attribute Name	Default Value	Description	List
mgmtinterface	"com.bea.security.providers.credentials.dbms.DatabaseCredentialMapper"		
AllowedTypes	"DBPASSWORD"	The types of credentials this provider is allowed to retrieve. If this attribute is set to a single value of asterisk (*), then all credential types are accepted and the queries determine if the type is appropriate.	Y
SelectByIdent	"true"	Enables selection of credentials from the database based on the username of the requesting identity.	N
SelectByIdentGroup	"false"	Enables selection of credentials from the database based upon the groups of the requesting identity.	N
DatabaseUserName		The username to use to log into the primary database connection pool.	N
DatabasePassword		The password to use to log into the primary database connection pool.	N
AdministratorUserName		The database username used for the administration of mappings.	N
AdministratorPassword		The database password used for the administration of mappings.	N
DatabaseProperties		Properties to use when creating a database connection in the primary connection pool. These properties are entered as NAME=VALUE	Y

Table 6-8 DatabaseCredentialMapper (Continued)

Attribute Name	Default Value	Description	List
DatabaseURL		The JDBC URL the primary connection pool uses to connect to the database. This attribute is also used for credential mapping administration.	N
DatabaseDriverName		The class name of the JDBC driver to use for the provider database connections. This attribute is also used for credential mapping administration.	N
ConnectionPoolMin	“5”	The minimum number of connections to allow in the primary connection pool.	N
ConnectionPoolMax	“20”	The maximum number of connections to allow in the primary connection pool.	N
ConnectionRetireTime	“120”	The number of seconds of idle time before a connection is removed from a connection pool.	N
EnableAutomaticFailover	“false”	Enables the use of the backup connection pool if the primary connection pool fails.	N
BackupDatabaseUserName		The username to use to log into the backup database.	N
BackupDatabasePassword		The password to use to log into the backup database.	N
BackupDatabaseProperties		Properties to use when obtaining the JDBC connection to the backup database. These properties are entered as NAME=VALUE	Y
BackupDatabaseURL		The JDBC URL to use to connect to the backup database.	N

Table 6-8 DatabaseCredentialMapper (Continued)

Attribute Name	Default Value	Description	List
BackupConnectionPoolMin	"0"	The minimum number of connections to allow in the backup connection pool.	N
BackupConnectionPoolMax	"20"	The maximum number of connections to allow in the backup connection pool.	N
FailureThreshold	"3"	The number of database errors that must occur sequentially on a connection pool before that pool is considered failed.	N
PrimaryRetryInterval	"30"	When operating with the backup pool, this setting determines how often the primary pool is evaluated for fail back. This value is in seconds.	N
QueryByIdent	"select username, password from asi_credential_map where byident = {0} and forident = {1} and config = {5}"	The query to use to retrieve credentials from the database based upon the requester identity. This query must return two columns, username and password. The password should be encrypted. The following placeholders are replaced in the query at runtime: {0} the username of the requesting identity {1} the username of the target identity {2} the normalized form of the resource {3} the normalized form of the action or default if none is defined {4} the credential type being requested {5} the name of this provider configuration	N

Table 6-8 DatabaseCredentialMapper (Continued)

Attribute Name	Default Value	Description	List
QueryByIdentGroup		The query to use to retrieve credentials from the database during group membership evaluation. If enabled, this query is called once for every group the forIdent user is in. This query must return two columns, username and password. The password should be encrypted. The following placeholders are replaced in the query at runtime: {0} the group name of the requesting identity {1} the username of the target identity {2} the normalized form of the resource {3} the normalized form of the action or default if none is defined {4} the credential type being requested {5} the name of this provider configuration	N
CountRecordQuery	"select count(*) from asi_credential_map where config = {0}"	The query to use to retrieve a count of the credential records associated with a specific configuration for administration of credential mappings. This query must return one numeric value. The following placeholders are replaced in the query at runtime: {0} the name of this provider configuration.	N

Table 6-8 DatabaseCredentialMapper (Continued)

Attribute Name	Default Value	Description	List
RetrieveRecordQuery	"select map_id, byident, forident, username, password, normalres, normalact, config from asi_credential_map where config = {0} and map_id = {1}"	The query to use to retrieve a credential record from the database for administration of credential mappings. This query must return a column for record id (numeric), byIdent, forIdent, username, password, resource, action, and config in that order. The password is encrypted. Resource, action and config are optional values (you may return null). All other columns must have values. The following placeholders are replaced in the query at runtime: {0} the name of the provider configuration {1} the record id being retrieved (numeric).	N
ListRecordsQuery	"select map_id, byident, forident, username, password, normalres, normalact, config from asi_credential_map where config = {0} order by byident,forident,username,normalres,normalact,map_id"	The query to use to retrieve a list of records from the database for use in the administration of credential mappings. This query must return a column for record id (numeric), byIdent, forIdent, username, password, resource, action and config in the correct order. The password is encrypted. Resource, action and config are optional values (you may return null). All other columns must have values. The following placeholders are replaced in the query at runtime: {0} the name of the provider configuration.	N

Table 6-8 DatabaseCredentialMapper (Continued)

Attribute Name	Default Value	Description	List
DeleteRecordQuery	"delete asi_credential_map where map_id = {1}"	The query to use delete a credential mapping record from the database. The following placeholders are replaced in the query at runtime: {0} the name of the provider configuration {1} the record id being deleted (numeric).	N
SaveRecordQuery	"update asi_credential_map set byident={0}, forident={1}, username={2}, normalres={3}, normalact={4} where map_id = {6}"	The query to use to update a credential mapping record from the database. This query is called whenever updates need to be recorded without a password change. The following placeholders are replaced in the query at runtime: {0} the username of the requesting user. {1} the username or alias of the target user. {2} the remote username {3} the normalized form of the resource {4} the normalized form of the action or default if none is defined {5} the name of the provider configuration {6} the record id being update (numeric).	N

Table 6-8 DatabaseCredentialMapper (Continued)

Attribute Name	Default Value	Description	List
SaveRecordWithPasswordQuery	"update asi_credential_map set byident={0}, forident={1}, username={2}, normalres={3}, normalact={4}, password={7} where map_id = {6}"	The query to use to update a credential mapping record from the database. This query is called whenever updates need to be recorded with a password change. The following placeholders are replaced in the query at runtime: 0} the username of the requesting user {1} the username username of the target user {2} the remote username {3} the normalized form of the resource {4} the normalized form of the action or default if none is defined {5} the name of the provider configuration {6} the record id being updated (numeric) {7} the encrypted password.	N

Table 6-8 DatabaseCredentialMapper (Continued)

Attribute Name	Default Value	Description	List
AddRecordQuery	"insert into asi_credential_map (byident, forident, username, password, normalres, normalact, config) values ({0}, {1}, {2}, {6}, {3}, {4}, {5})"	The query to use to add a credential mapping record to the database. The following placeholders are replaced in the query at runtime: {0} the username of the requesting user {1} the username or alias of the target user {2} the remote username {3} the normalized form of the resource {4} the normalized form of the action or default if none is defined {5} the name of the provider configuration {6} the encrypted password.	N
SharedSecret		A secret pass-phrase used to decrypt passwords stored in the database. Only passwords encrypted with this same secret pass-phrase are available to this provider. **Note:** Changing this secret phrase invalidates all currently stored passwords. If you change this shared secret you will have to reset the passwords in the database so that they match.	N

DefaultAuthenticator

DefaultAuthenticator extends weblogic.management.security.authentication.Authenticator. Table 6-9 describes the attributes supported by this security provider.

Table 6-9 DefaultAuthenticator

Attribute Name	Default Value	Description	List
mgmtinterface	weblogic.security.providers.authentication.DefaultAuthenticator		N
MinimumPasswordLength	“8”	The minimum number of characters required in a password.	N
SupportedImportFormats	{“DefaultAtn”}	The format of the file to import. The list of supported import formats is determined by the AUthentication provider from which the users and groups were originally exported.	Y
SupportedImportConstraints		The users and groups that you want to be imported into this Authentication provider’s database. If none are specified, all are imported.	Y
SupportedExportFormats	{“Default”}	The format of the file to export. The list of supported export formats is determined by this Authentication provider.	Y
SupportedExportConstraints	{“users”,”groups”}	The users and groups that you want to be exported from this Authentication provider’s database. If none are specified, all are exported.	Y
GroupMembershipSearching	“unlimited”	Specifies whether recursive group membership searching is unlimited or limited. Valid values are unlimited and limited	N

Table 6-9 DefaultAuthenticator (Continued)

Attribute Name	Default Value	Description	List
MaxGroupMembershipSearchLevel	“0”	This setting specifies how many levels of group membership can be searched. This setting is valid only if GroupMemberShipSearching is set to limited	N
UseRetrievedUserNameAsPrincipal	“false”	This flag specifies whether we should use the username retrieved from LDAP as the principal in the subject.	N

DefaultAuthorizer

DefaultAuthorizer extends weblogic.management.security.authorization.DeployableAuthorizer. Table 6-10 describes the attributes supported by this security provider.

Table 6-10 DefaultAuthorizer

Attribute Name	Default Value	Description	List
mgmtinterface	weblogic.security.providers.authorization.DefaultAuthorizer		N
SupportedImportFormats	{“DefaultAtz”}	The format of the file to import. The list of supported import formats is determined by the Authorization provider from which the authorization policies were originally exported.	Y
SupportedImportConstraints		The authorization policies that you want to be imported into this Authorization provider's database. If none are specified, all are imported.	Y

Table 6-10 DefaultAuthorizer (Continued)

Attribute Name	Default Value	Description	List
SupportedExport Formats	{“DefaultAtz”}	The format of the file to export. The list of supported export formats is determined by this Authorization provider.	Y
SupportedExport Constraints		The authorization policies that you want exported from this Authorization provider's database. If none are specified, all are exported.	Y

DefaultCredentialMapper

DefaultCredentialMapper extends weblogic.management.security.credentials.DeployableCredentialMapper. Table 6-11 describes the attributes supported by this security provider.

Table 6-11 DefaultCredentialMapper

Attribute Name	Default Value	Description	List
mgmtinterface	weblogic.security.providers.credentials.DefaultCredentialMapper		N
SupportedImport Formats	{“DefaultCreds”}	The format of the fie to import. The list of supported import formats is determined by the Credential Mapping provider from which the credential maps were originally exported.	Y
SupportedImport Constraints		The credential maps that you want to be imported into this Credential Mapping provider’s database. If none are specified, all are imported.	Y

Table 6-11 DefaultCredentialMapper (Continued)

Attribute Name	Default Value	Description	List
SupportedExport Formats	{“DefaultCreds”}	The format of the file to export. The list of supported export formats is determined by this Credential Mapping provider.	Y
SupportedExport Constraints	{“passwords”}	The credential maps that you want to be exported from this Credential Mapping provider’s database. If none are specified, all are exported.	Y

DefaultRoleMapper

DefaultRoleMapper extends weblogic.management.security.authorization.DeployableRoleMapper. Table 6-12 describes the attributes supported by this security provider.

Table 6-12 DefaultRoleMapper

Attribute Name	Default Value	Description	List
mgmtinterface	weblogic.security.providers.authorization.DefaultRoleMapper		
SupportedImport Formats	{“DefaultRoles”}	The format of the file to import. The list of supported import formats is determined by the Role Mapping provider from which the security roles were originally exported.	
SupportedImport Constraints		The security roles that you want to be imported into this Role Mapping provider’s database. If none are specified, all are imported.	

Table 6-12 DefaultRoleMapper (Continued)

Attribute Name	Default Value	Description	List
SupportedExport Formats	{“DefaultRoles”}	The format of the file to export. The list of supported export formats is determined by this Role Mapping provider.	
SupportedExport Constraints		The security roles you want to be exported from this Role Mapping provider’s database. If none are specified, all are exported.	

IPlanetAuthenticator

IPlanetAuthenticator extends com.bea.security.providers.authentication.LDAPAuthenticator. Table 6-13 describes the attributes supported by this security provider.

Table 6-13 IPlanetAuthenticator

Attribute Name	Default Value	Description	List
mgmtinterface	“com.bea.security.providers.authentication.IplanetAuthenticator”		N
GroupFromName Filter	“(|(&(cn=%g)(objectclass=groupofUniqueNames))(&(cn=%g)(objectclass=groupOfURLs)))”	An LDAP search filter for finding a group given the name of the group. If the attribute is not specified (that is, if the attribute is null or empty), a default search filter is created based on the group schema.	N
StaticMemberDN Attribute	“member”	The attribute of an LDAP static group object that specifies the distinguished names (DNs) of the members of the group.	N
DynamicGroupObjectClass	“groupofURLs”	The LDAP object class that stores dynamic groups.	N

Table 6-13 IPlanetAuthenticator (Continued)

Attribute Name	Default Value	Description	List
DynamicGroupNameAttribute	“cn”	The attribute of the dynamic LDAP group object that specifies the name of the group.	N
DynamicMemberURLAttribute	“memberURL”	The attribute of the dynamic LDAP group object that specifies the URLs of the members of the dynamic group.	N

LDAPAuthenticator

LDAPAuthenticator extends weblogic.management.security.authentication.Authenticator. Table 6-14 describes the attributes supported by this security provider.

Table 6-14 LDAPAuthenticator

Attribute Name	Default Value	Description	List
mgmtinterface	com.bea.security.providers.authentication.LDAPAuthenticator		N
UserObjectClass	“person”	The LDAP object class that stores users.	N
UserNameAttribute	“uid”	The attribute of an LDAP user object that specifies the name of the user.	N
UserDynamicGroupDNAttribute		The attribute of an LDAP user object that specifies the distinguished names (DNs) of dynamic groups to which this user belongs. If such an attribute does not exist, WebLogic Server determines if a user is a member of a group by evaluating the URLs on the dynamic group. If a group contains other groups, WebLogic Server evaluates the URLs on any of the descendents (indicates parent relationship) of the group.	N

Table 6-14 LDAPAuthenticator (Continued)

Attribute Name	Default Value	Description	List
UserBaseDN	"ou=people, o=example.com"	The base distinguished name (DN) of the tree in the LDAP directory that contains users.	N
UserSearchScope	"subtree"	Specifies how deep in the LDAP directory tree to search for Users. Valid values are subtree and onelevel.	N
UserFromNameFilter	"(&(uid=%u)(objectclass=person))"	An LDAP search filter for finding a user given the name of the user. If the attribute (user name attribute and user object class) is not specified (that is, if the attribute is null or empty), a default search filter is created based on the user schema.	N
AllUsersFilter		An LDAP search filter for finding all users beneath the base user distinguished name (DN). If the attribute (user object class) is not specified (that is, if the attribute is null or empty), a default search filter is created based on the user schema.	N
GroupBaseDN	"ou=groups, o=example.com"	The base distinguished name (DN) of the tree in the LDAP directory that contains groups.	N
GroupSearchScope	"subtree"	Specifies how deep in the LDAP directory tree to search for groups. Valid values are subtree and onelevel.	N
GroupFromNameFilter	(&(cn=%g)(objectclass=groupofuniquenames))	An LDAP search filter for finding a group given the name of the group. If the attribute is not specified (that is, if the attribute is null or empty), a default search filter is created based on the group schema.	N

Table 6-14 LDAPAuthenticator (Continued)

Attribute Name	Default Value	Description	List
AllGroupsFilter		An LDAP search filter for finding all groups beneath the base group distinguished name (DN). If the attribute is not specified (that is, if the attribute is null or empty), a default search filter is created based on the Group schema.	N
StaticGroupObjectClass	“groupofuniquenames”	The name of the LDAP object class that stores static groups.	N
StaticGroupNameAttribute	“cn”	The attribute of a static LDAP group object that specifies the name of the group.	N
StaticMemberDNAttribute	“uniquemember”	The attribute of a static LDAP group object that specifies the distinguished names (DNs) of the members of the group.	N
StaticGroupDNsfromMemberDNFilter	(&(uniquemember=%M)(objectclass=groupofuniquenames))	An LDAP search filter that, given the distinguished name (DN) of a member of a group, returns the DNs of the static LDAP groups that contain that member. If the attribute is not specified (that is, if the attribute is null or empty), a default search filter is created based on the group schema.	N
DynamicGroupObjectClass		The LDAP object class that stores dynamic groups.	N
DynamicGroupNameAttribute		The attribute of a dynamic LDAP group object that specifies the name of the group.	N
DynamicMemberURLAttribute		The attribute of the dynamic LDAP group object that specifies the URLs of the members of the dynamic group.	N

Table 6-14 LDAPAuthenticator (Continued)

Attribute Name	Default Value	Description	List
AutomaticFailoverEnabled	“false”	The option to enable automatic failover when using the LDAP server.	N
BackupHost	“localhost”	The host name or IP address of the backup LDAP server.	N
BackupPort	“389”	The port number on which the backup LDAP server is listening.	N
BackupSSLEnabled	“false”	The option to enable SSL when connecting to the backup LDAP server.	N
BackupPrincipal		The Distinguished Name (DN) of the LDAP user that is authorized to connect to the backup LDAP server.	N
BackupCredential		The credential (generally a password) used to authenticate the backup LDAP user that is defined in the Principal attribute.	N
PrimaryRetryInterval	“3600”	Length of time, in seconds, before the backup LDAP server tries to fail back to the primary LDAP server.	N
GroupMembershipSearching	“unlimited”	Specifies whether recursive group membership searching is unlimited or limited. Valid values are unlimited and limited.	N
MaxGroupMembershipSearchLevel	“0”	This setting specifies how many levels of group membership can be searched. This setting is valid only if GroupMemberShipSearching is set to limited. Valid values are 0, and positive integers. For example, 0 indicates only direct group memberships will be found, positive number indicates the number of levels to go down.	N

Table 6-14 LDAPAuthenticator (Continued)

Attribute Name	Default Value	Description	List
VerifyUserForIdentityAssertion	“false”	Whether to verify that the user is present in the LDAP repository when an identity assertion is provided.	N
AddGroupsFromIdentityAssertion	“false”	Whether to add groups for the user from the identity assertion when Identity Assertion is turned on.	N
AddGroupsFromLocalLDAP	“true”	Whether to add groups for the user from the local LDAP identity store after user authentication.	N

Log4jAuditor

Log4jAuditor extends weblogic.management.security.audit.Auditor. Table 6-15 describes the attributes supported by this security provider.

Table 6-15 Log4jAuditor

Attribute Name	Default Value	Description	List
mgmtinterface	com.bea.security.providers.audit.Log4JAuditor		N
Severity	"ERROR"	Severity is the lowest level at which auditing is initiated. Audit event severity is treated as follows by the Log4j Audit Channel provider. INFORMATION SUCCESS WARNING ERROR FAILURE For example, if the log4j severity threshold is set to ERROR (default setting), then all audit events with severity ERROR and FAILURE are audited. Different audit events can be selectively audited depending on the setting for each of them. All audit events can be DISABLED or WITHOUT_CONTEXT. Those that have context, you can select WITH_CONTEXT.	N

Table 6-15 Log4jAuditor (Continued)

Attribute Name	Default Value	Description	List
Log4jConfigProperties	{“log4j.appender.ASIauditFile=org.apache.log4j.RollingFileAppender”,“log4j.appender.ASIauditFile.File={HOME}/log/secure_audit.log”,“log4j.appender.ASIauditFile.layout=org.apache.log4j.PatternLayout”,“log4j.appender.ASIauditFile.layout.ConversionPattern=%d[%t] %-5p %c - %m%n”,“log4j.logger.ASI_AUDIT=NULL, ASIauditFile”,“log4j.additivity.ASI_AUDIT=false”	These properties are passed to log4j upon initialization of the log4j provider. By default the log4j provider uses the RollingFileAppender. {HOME} will be replaced with the current location of the SSM at runtime. This setting is determined by the value of instance.home in SSM.properties. Custom log4j appenders can be configured here to send the Auditing information to other destinations such as JMS, NT Events log, JDBC etc. For more information, see the log4j documentation.	Y
Log4jRendererProperties		Custom renderers can be added here for rendering classes that implement the weblogic.security.spi.AuditEvent interface. For example, weblogic.security.spi.AuditEvent=com.bea.security.providers.audit.AuditEventRenderer See Log4J documentation on how to write a renderer for a custom object. Be sure to include the jar file containing the custom renderer classes in the `ALES_HOME/lib/providers` directory	Y

Table 6-15 Log4jAuditor (Continued)

Attribute Name	Default Value	Description	List
EnabledAuditEvents		List of AuditEvent types that will be Audited other than the default ones that can be configured using drop down boxes. Custom AuditEvents not listed here will not be audited. The exception to this is if you set Audit Event to `WITHOUT_CONTEXT`. In that case all AuditEvents will be audited. Custom AuditEvents can be added using following interface: weblogic.security.spi.AuditEvent.	
AuditEvent	"WITHOUT_CONTEXT"	Setting for events of type weblogic.security.spi.AuditEvent. **Note:** If you set Audit Event to `WITHOUT_CONTEXT`, then all AuditEvents will be enabled for auditing.	N
AuditAuthenticationEvent	"WITHOUT_CONTEXT"	Setting for events of type weblogic.security.spi.AuditAtnEvent.	N
AuditAuthorizationEvent	"WITHOUT_CONTEXT"	Setting for events of type weblogic.security.spi.AuditAtzEvent.	N
AuditRoleEvent	"WITHOUT_CONTEXT"	Setting for events of type weblogic.security.spi.AuditRoleEvent.	N
AuditProviderRecordEvent	"WITHOUT_CONTEXT"	Setting for events of type com.bea.security.spi.ProviderAuditRecord	N
AuditManagementEvent	"WITHOUT_CONTEXT"	Setting for events of type weblogic.security.spi.AuditMgmtEvent	N

Table 6-15 Log4jAuditor (Continued)

Attribute Name	Default Value	Description	List
AuditPolicyEvent	"WITHOUT_CONTEXT"	Setting for events of type weblogic.security.spi.AuditPolicyEvent	N
AuditRoleDeploymentEvent	"WITHOUT_CONTEXT"	Setting for events of type weblogic.security.spi.AuditRoleDeploymentEvent	N

NovellAuthenticator

NovellAuthenticator extends com.bea.security.providers.authentication.LDAPAuthenticator. Table 6-16 describes the attributes supported by this security provider.

Table 6-16 NovellAuthenticator

Attribute Name	Default Value	Description	List
mgmtinterface	"com.bea.security.providers.authentication.NovellAuthenticator"		N
UserNameAttribute	"cn"	The attribute of an LDAP user object that specifies the name of the user.	N
UserFromNameFilter	"(&(cn=%u)(objectclass=person))	An LDAP search filter for finding a user given the name of the user. If the attribute is not specified (that is, if the attribute is null or empty), a default search filter is created based on the user schema.	N
GroupFromNameFilter	(&(cn=%g)(objectclass=groupofnames))	An LDAP search filter for finding a group given the name of the group. If the attribute is not specified (that is, if the attribute is null or empty), a default search filter is created based on the group schema.	N

Table 6-16 NovellAuthenticator

Attribute Name	Default Value	Description	List
StaticGroupObjectClass	“groupofnames”	The name of the LDAP object class that stores static groups.	N
StaticGroupDNsfromMemberDNFilter	“(&(uniquemember=%M)(objectclass=groupofnames))	An LDAP search filter that, given the distinguished name (DN) of a member of a group, returns the DNs of the static LDAP groups that contain that member. If the attribute is not specified (that is, if the attribute is null or empty), a default search filter is created based on the group schema.	N

NTAuthenticator

NTAuthenticator extends weblogic.management.security.authentication.Authenticator. Table 6-17 describes the attributes supported by this security provider.

Table 6-17 NTAuthenticator

Attribute Name	Default Value	Description	List
mgmtinterface	“weblogic.security.providers.authentication.NTAuthenticator”		N
DomainControllers	“localanddomain”	The domain controllers to use for locating unscoped usernames during authentication, listing users or groups, and handling unscoped names. local - Uses only the local machine. localanddomain - the default, uses the local machine and the domain of which the machine is a member, if it is not standalone. domain - Uses only the domain of which the machine is a member. list - Uses the list of domain controllers specified ny the DomainControllerList setting.	N
DomainControllerList	{“ [localanddomain]”}	The list of Domain controllers to use for locating unscoped usernames during authentication, listing users or groups, and handling unscoped names. This setting is only used if the DomainControllers setting is set to list. The list should contain the domain controller names for trusted domains that you want to use. Placeholders are supported and expanded if specified. Supported placeholders are [local], [localanddomain], [domain]	Y

Table 6-17 NTAuthenticator (Continued)

Attribute Name	Default Value	Description	List
BadDomainControllerRetry	“delay”	Controls how the provider reacts when a bad domain controller name is found. BADDCRetryDelayString indicates the domain controller can be used again only after a certain amount of time has elapsed since it was last tried unsuccessfully. BADDCRetryNeverString indicates a bad domain controller is never retried. BADDCRetryAlwaysStringindicates a bad domain controller is always retried. The default is BADDCRetryDelayString.	N
BadDomainControllerRetryInterval	“60000”	Amount of time to wait when a bad domain controller name is found before trying to use the domain controller again. This option is only used when the BadDomainControllerRetry setting is configured to use delay (BADDCRetryDelayString). The default setting is 60000 ms (one minute). This setting helps reduce performance hits when a domain controller in the list of controllers is temporarily unavailable.	N

Table 6-17 NTAuthenticator (Continued)

Attribute Name	Default Value	Description	List
MapUPNNames	“first”	Indicates how the Authenticator attempts to map UPN style names for authentication. For example, username@domain. domain\\username is not ambiguous and is always allowed. MAP UPNNames First String - a name that matches the UPN format is treated as a UPN name first. If it is not a UPN name, the name is treated as an unscoped name. MAP UPNNames Last String - a name that matches the UPN format is treated as a UPN name, only if the name fails to match as an unscoped name. MAP UPN Names Always String - a name that matches the UPN format is always treated as an unscoped name and not treated as a UPN name. MAP UPNNames Never String - a name that matches the UPN format is always treated as a UPN name. Only use this option when you are certain there are no usernames that contain an @ symbol.	N

Table 6-17 NTAuthenticator (Continued)

Attribute Name	Default Value	Description	List
LogonType	"interactive"	This option indicates whether to perform a network or an interactive logon.	N
MapNTDomainName	"never"	Indicates whether to insert the Windows NT domain information into the principal name during authentication and the proper format to use. MAP NTDomain Name Never String - the Windows NT domain name is never inserted into the principal name. MAP NTDomain Name UPNString - the Windows NT domain name is inserted into the principal name using the style domain\\name. MAP NTDomain Name Never String - the Windows NT domain name is inserted into the principal name using the style name@domain.	N

OpenLDAPAuthenticator

OpenLDAPAuthenticator extends com.bea.security.providers.authentication.LDAPAuthenticator. Table 6-18 describes the attributes supported by this security provider.

Table 6-18 OpenLDAPAuthenticator

Attribute Name	Default Value	Description	List
mgmtinterface	com.bea.security.providers.authentication.OpenLDAPAuthenticator		N
UserNameAttribute	"cn"	The attribute of an LDAP user object that specifies the name of the user.	N

Table 6-18 OpenLDAPAuthenticator (Continued)

Attribute Name	Default Value	Description	List
UserBaseDN	“ou=people, dc=example, dc=com”	The base distinguished name (DN) of the tree in the LDAP directory that contains users.	N
UserFromNameFilter	“((cn=%u)(objectclass=person))”	An LDAP search filter for finding a user given the name of the user. If the attribute (user name attribute and user object class) is not specified (that is, if the attribute is null or empty), a default search filter is created based on the user schema.	N
GroupBaseDN	“ou=groups, dc=example, dc=com”	The base distinguished name (DN) of the tree in the LDAP directory that contains groups.	N
GroupFromNameFilter	“((cn=%g)(objectclass=groupofnames))”	An LDAP search filter for finding a group given the name of the group. If the attribute is not specified (that is, if the attribute is null or empty), a default search filter is created based on the group schema.	N
StaticGroupObjectClass	“groupofnames”	The name of the LDAP object class that stores static groups.	N
StaticMemberDNAttribute	“member”	The attribute of an LDAP static group object that specifies the distinguished names (DNs) of the members of the group.	N
StaticGroupDNsfromMemberDNFilter	“((member=%M)(objectclass=groupofnames))”	An LDAP search filter that, given the distinguished name (DN) of a member of a group, returns the DNs of the static LDAP groups that contain that member.	N

PerfDBAuditor

PerfDBAuditor extends weblogic.management.security.audit.Auditor. Table 6-19 describes the attributes supported by this security provider.

Table 6-19 PerfDBAuditor

Attribute Name	Default Value	Description	List
mgmtinterface	com.bea.security.providers .audit.PerfDBAuditor		N
PerformanceStatistics Interval	5	Performance statistics gathering interval, in minutes.	
PerformanceStatistics Duration	0	Length of circular buffer, in minutes . Must be greater than or equal to the interval. A value of 0 means unlimited duration.	
EnablePerformanceSt atistics	true	Enables/disables performance-gathering counters.	
JDBCDriverClassNa me	oracle.jdbc.driver.OracleD river	The Java class name of the JDBC Driver.	
JDBCConnectionUR L		The connection string for the authentication database.	
DatabaseUserLogin		The user id to access the authentication database.	
DatabaseUserPasswor d		The user password to access the authentication database.	
JDBCConnectionProp erties		Optional parameters for configuring the JDBC Connection. Legal values are determined by the JDBC Driver. These properties are entered as NAME=VALUE.	
AuthenticationStatisti csTable	PERF_ATH_STAT	Database table for collecting authentication statistics.	

Table 6-19 PerfDBAuditor

Attribute Name	Default Value	Description	List
AuthorizationStatisticsTable	PERF_ATZ_STAT	Database table for collecting authorization statistics.	
AuthorizationAttrStatisticsTable	PERF_ATZ_ATTR_STAT	Database table for collecting authorization attribute statistics.	
AuthorizationFuncStatisticsTable	PERF_ATZ_FUNC_STAT	Database table for collecting authorization function statistics.	

ResourceDeploymentAuditor

ResourceDeploymentAuditor extends weblogic.management.security.audit.Auditor. Table 6-20 describes the attributes supported by this security provider.

Table 6-20 ResourceDeploymentAuditor

Attribute Name	Default Value	Description	List
mgmtinterface	com.bea.security.providers.audit.ResourceDeploymentAuditor		N
ResourceDeploymentEnabled	“true”	If “true” the audit provider will publish resources to the AquaLogic Enterprise Security Administration Application.	N
ResourceDeploymentNamingAuthority	“RESOURCEDEPLOYMENT”	The naming authority of audit events to process as resource deployment audit events.	N
SessionEvictionCapacity	“40”	The number of sessions to actively maintain. Once the limit is reached, old sessions are dropped and automatically reestablished when needed.	N
SessionEvictionPercentage	“25”	The percentage of the sessions to drop when the eviction capacity is reached.	N

Table 6-20 ResourceDeploymentAuditor

Attribute Name	Default Value	Description	List
SessionLifetime	"120000"	Specifies the maximum length of time (in milliseconds) a session can use before it is discarded. A value of 0 indicates that sessions are indefinite.	N
SessionMaxUses	"100"	Specifies the maximum number of times a session can be used before it is discarded. A value of 0 indicates that sessions are indefinite.	N
ApplicationDeploymentParent	"//app/policy"	Specifies the root of the resource tree where new resources are published.	N
SharedResourcesParent	"shared"	Specifies the root of the resource tree where new shared resources are published. This item may be relative to the value specified by ApplicationDeploymentParent	N
ResourceConverters		Specifies the types of resources which are supported by this provider. The value is a list of fully qualified Java class names. These classes should implement the ResourceConverter interface. AquaLogic Enterprise Security includes resource converters for the standard WebLogic Server resource types.	Y
InstantiateWeblogicResourceConverters	"true"	Instantiate Resource Converters for all default WebLogic resource types. When set to true, these converters are not listed in the ResourceConverter configuration attribute. The default set of converters supports all native WebLogic Server resource types.	N

Table 6-20 ResourceDeploymentAuditor

Attribute Name	Default Value	Description	List
AttributeConverters		Specifies plugins to convert native Java types into the corresponding AquaLogic Enterprise Security string representation. If a converter is not registered for a given type, then the `toString()` method is used by default.	Y
AnonymousSubjectName	"anonymous"	The subject name to use when publishing resources for an anonymous user.	N
IdentityDirectory	"asi"	Specifies the identity directory to use while publishing resources.	N
Domain		Specifies the enterprise domain to use while publishing resources.	N

SAMLCredentialMapper

SAMLCredentialMapper extends weblogic.management.security.credentials.CredentialMapper. Table 6-21 describes the attributes supported by this security provider.

Table 6-21 SAMLCredentialMapper

Attribute Name	Default Value	Description	List
mgmtinterface	"com.bea.security.providers.credentials.saml.SAMLCredentialMapper"		N
TrustedKeystore	"{HOME}/ssl/demoProviderTrust.jks"	The Keystore to be used to get the Certificate chain to sign the SAML Assertion with. {HOME} will be replaced with the SSM instance directory at runtime. This setting is determined by the value of instance.home in SSM.properties located in the /config directory of the SSM instance. If SSM.properties cannot be located, then the system property wles.ssmws.instance.home is checked. For the Web Services SSM, this attribute is automatically set to the Web Services SSM instance home.	N
TrustedKeystoreType	"JKS"	The TYPE of keystore that is specified in TrustedKeystore.	N
TrustedCertAlias	"demo_provider_trust"	The Cert alias to be used to sign the SAML Assertion.	N
TrustedCertAliasPasswd	"password"	The password to use for the CertAlias specified to retrieve the private key from the keystore.	N
NotBeforeOffset	"120"	The number of seconds in the past to make an assertion valid to allow for clock skew.	N
NotAfterOffset	"300"	The number of seconds in the future to make an assertion valid.	N

Table 6-21 SAMLCredentialMapper (Continued)

Attribute Name	Default Value	Description	List
IssuerURI	“https://www.bea.com”	The value of the Issuer attribute for SAML assertions.	N
Base64Encoding Required	“false”	Encode generated SAML Assertion using Base64.	N

SAMLIdentityAsserter

SAMLIdentityAsserter extends weblogic.management.security.authentication.IdentityAsserter. Table 6-22 describes the attributes supported by this security provider.

Table 6-22 SAMLIdentityAsserter

Attribute Name	Default Value	Description	List
SupportedTypes	{“SAML.Challenge”,“SAML.Assertion”,“SAML.Profile.POST”}	The active types supported by the SAML Identity Assertion provider.	Y
ActiveTypes	“SAML.Challenge”,“SAML.Assertion”,“SAML.Profile.POST”}	Specifies the type currently used by the SAML Identity Assertion provider.	Y

Table 6-22 SAMLIdentityAsserter (Continued)

Attribute Name	Default Value	Description	List
TrustedCAKeystore	"{HOME}/ssl/demoProviderTrust.jks"	The location of the Trusted Keystore stored in the TrustedCAKeystoreType keystore format. {HOME} will be replaced with the SSM instance directory at runtime. This setting is determined by the value of instance.home in SSM.properties located in the /config directory of the SSM instance. If SSM.properties cannot be located, then the system property wles.ssmws.instance.home is checked. For the Web Services SSM, this attribute is automatically set to the Web Services SSM instance home. If DEFAULT is specified, then the java.home env variable is used to locate the cacerts keystore normally located at JAVA_HOME/lib/security/cacerts.	N
TrustedKeystore	{HOME}/ssl/demoProviderTrust.jks"	The Location of the Trusted Keystore stored in the TrustedKeystoreType keystore format. {HOME} will be replaced with the SSM instance directory at runtime. This setting is determined by the value of instance.home in SSM.properties located in the /config directory of the SSM instance. If SSM.properties cannot be located, then the system property wles.ssmws.instance.home is checked. For the Web Services SSM, this attribute is automatically set to the Web Services SSM instance home.	N

Table 6-22 SAMLIdentityAsserter (Continued)

Attribute Name	Default Value	Description	List
TrustedCAKeystoreType	"JKS"	The type of keystore to which the trustedCAKeystore is configured.	N
TrustedKeystoreType	"JKS"	The type of keystore to which the trustedKeystore is configured.	N
Base64DecodingRequired	"false"	Decode inbound SAML Assertion using Base64.	N

SinglePassNegotiateIdentityAsserter

SinglePassNegotiateIdentityAsserter extends weblogic.management.security.authentication.IdentityAsserter. Table 6-23 describes the attributes supported by this security provider.

Table 6-23 SinglePassNegotiateIdentityAsserter

Attribute Name	Default Value	Description	List
mgmtinterface	com.bea.security.providers.authentication.spnego.SinglePassNegotiateIdentityAsserter		N
SupportedTypes	{"SPNEGO.AtnAssertion","Authorization"}	The token types supported by the Single Pass Negotiate Identity Assertion provider.	Y
ActiveTypes	{"SPNEGO.AtnAssertion","Authorization"	Specifies the token types currently used by the Single Pass Negotiate Identity Assertion provider.	N

X509IdentityAsserter

X509IdentityAsserter extends weblogic.management.security.authentication.IdentityAsserter. Table 6-24 describes the attributes supported by this security provider.

Table 6-24 X509IdentityAsserter

Attribute Name	Default Value	Description	List
mgmtInterface	“com.bea.security.providers.authentication.X509IdentityAsserter”		N
SupportedTypes	AuthenticatedUser X.509 CSI.PrincipalName&quot CSI.ITTAnonymous CSI.X509CertChain&quot CSI.DistinguishedName	The token types supported by the AquaLogic Enterprise Security Identity Assertion provider.	Y
UserNameMapperClassName		The name of the Java class that maps X.509 digital certificates and X.501 distinguished names to AquaLogic Enterprise Security user names.	N
TrustedClientPrincipals		The list of trusted client principals to use in CSI v2 identity assertion. The wildcard character (*) can be used to specify all principals are trusted. If a client is not listed as a trusted client principal, the CSIv2 identity assertion fails and the invoke is rejected.	Y
UseDefaultUserNameMapper	“false”	Specifies whether this X.509 Identity Assertion provider uses the default user name mapper implementation.	N
DefaultUserNameMapperAttributeType	“E”	The name of the attribute from the subject Distinguished Name (DN), which this Identity Assertion provider uses when mapping from the X.509 digital certificate or X.500 name token to the user name.	N
DefaultUserNameMapperAttributeDelimiter	“@”	The delimiter that ends the attribute value when mapping from the X.509 digital certificate or X.500 name token to the user name.	N

XACMLAuthorizer

XACMLAuthorizer extends weblogic.management.security.authorization.Authorizer.

Table 6-25 describes the attributes supported by this security provider.

Table 6-25 XACMLAuthorizer

Attribute Name	Default Value	Description	List
mgmtInterface	com.bea.security.providers.authorization.xacml.XACMLAuthorizer		N
PolicyDirectory	"xacmlpolicy"	The directory that contains XACML policy files.	N
SCMPolicyDeploymentEnabled	"false"	Enables XACML policy deployment via the SCM.	N
SCMPollingPeriod	"1000"	When XACML SCM policy deployment is enabled, this parameter configures how often (in milliseconds) the provider polls the SCM for XACML policy changes.	N
XACMLPolicy		The XACML policy that is provisioned to the XACML authorization provider via the SCM.	Y