BEA AquaLogic Enterprise Security™®

Administration Reference

Version 2.2
Document Revised: June 2006

Copyright

Restricted Rights Legend

Third-Party Software License Agreement

Sun Microsystems, Inc.’s XACML implementation v2.0

This product includes Sun Microsystems, Inc.’s XACML implementation v2.0, which is governed by the following terms:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistribution of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistribution in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Sun Microsystems, Inc. or the names of contributors maybe used to endorse or promote products derived from this software without specific prior written permission.

This software is provided "AS IS," without a warranty of any kind. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN MICROSYSTEMS, INC. ("SUN") AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You acknowledge that this software is not designed or intended for use in the design, construction, operation or maintenance of any nuclear facility.

For all third-party software license agreements, see the 3rd_party_licenses.txt file, which is placed in the \ales22-admin directory when you install the AquaLogic Enterprise Security Administration Server.

Trademarks and Service Marks

1. Introduction and Roadmap

2. Administrative Utilities

3. Provider Extensions

4. Audit Events

5. Policy Language Custom Extension Library API Reference

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, "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 4, "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 Authorization and Role Mapping Engine (ARME), 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:

- "unenroll" on page 2-17

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/ales22-admin`.
- angle brackets <> indicate required variable arguments
- square brackets [] indicate optional variable arguments

bootloader

Loads a boot policy for the default set of providers for the ALES Administration SSM. This bootloader is a privileged loader that only loads the initial boot policy so that the providers required for the Administration Server are configured to their initial settings. Once this boot policy has been set, then the regular policyloader can be used to load the admin policy. The policyloader requires authentication and authorization to run.

The only input to the boot loader is a Java properties file, `[asi.properties]`. The `ALES_ADMIN_HOME/config/asi.properties` file will be used if no filename is provided on the command line.

Usage

```
ALES_ADMIN_HOME\bin\bootloader.bat [asi.properties] [-help] [-recover]
[-recoverWithRecoveryAppParent]
```

```
ALES_ADMIN_HOME/bin/bootloader.sh [asi.properties] [-help] [-recover]
[-recoverWithRecoveryAppParent]
```

Options

The following options are supported:

-help

Print USAGE: bootLoader [asi.properties] [-help] [-recover] [-recoverWithRecoveryAppParent] and exit.

-recover

Run in Recover mode. In this mode, the bootloader tool will delete the existing admin provider configuration and install the initial boot configuration. This option should be used only if you want to get back to the original default settings for the providers used by

the admin configuration. This will not affect any policy or any other SSM configurations other then the `asiadmin` SSM configuration.

-recoverWithRecoveryAppParent

Run in Recover mode with Recovery App Parent. In this mode, the bootloader tool will delete the existing admin provider configuration and install the initial boot configuration. This option needs to be used in conjunction with the BLM also running with the configuration property `BLM.wlesadmin.adminPolicyRoot` pointing to `//app/policy/ASI/recovery`. To use this mode, you must edit the `ALES_ADMIN_HOME/config/WLESblm.conf` file and restart the BLM server before running the bootloader utility in this mode.This is needed in the rare case that the `asiadmin` SSM configuration has been updated in a way that prevents access to the ALES Administration Console and you have locked the system user from making changes to the regular admin policy.

Example

```
>bootloader.bat -recover
```

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.

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 and the bootloader has been run. 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. An example would be moving policy from a development installation to a QA installation, or from a staging installation to a production deployment. 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.

Usage

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

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

Options

-import

Run the tool in policy import mode.

-export

Run the tool in policy export mode.

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 the comments in the sample `policyIX_config.xml` file for information about the values to include in your configuration file.

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.

Example

To export a policy:

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

To import a policy:

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

export_policy_oracle

Export ALES policy data from an Oracle 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.

The `ORACLE_HOME` environment variable must be set to the Oracle Client directory. Also make sure your `PATH` environment variable includes the current directory and the `/bin` directory of the Oracle client. On UNIX, make sure that `LD_LIBRARY_PATH` is also set correctly.

Usage

```
ALES_ADMIN_HOME\bin\export_policy_oracle.bat <server> <owner> <dblogin>
<password> [directory]
```

```
ALES_ADMIN_HOME/bin/export_policy_oracle.sh
```

Options

server
Oracle database server name

owner
Owner of the policy

dblogin
Login id, usually same as owner

password
Password for the owner

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

Example

```
>export_policy_oracle.bat DBSERVER wles wles password c:\MyPolicy
```

export_policy_sybase

Exports ALES policy data from Sybase 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.

The `SYBASE` environment variable must be set. The `SYBASE_OCS` environment variable must be set for the Sybase 12 open client. Also make sure your PATH environment variable includes the current directory and the `\bin` and `\dll` subdirectories of the Sybase open client. On UNIX, make sure that `LD_LIBRARY_PATH` is also set correctly.

Usage

```
ALES_ADMIN_HOME\bin\export_policy_sybase.bat <server> <database> <owner>
<login> <password> [directory]
```

```
ALES_ADMIN_HOME/bin/export_policy_sybase.sh
```

Options

server
Sybase database server name

database
Database name

owner
Owner of the policy

dblogin
Login id, usually same as owner

password
Password for the owner

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

Example

>export_policy_sybase.bat DBSERVER sspolicy wles wles password c:\MyPolicy

install_schema_oracle

Installs the ALES policy database schema into an Oracle 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.

The `ORACLE_HOME` environment variable must be set to the Oracle Client directory. Also make sure your `PATH` environment variable includes the current directory and the `/bin` directory of the Oracle client. On UNIX, make sure that `LD_LIBRARY_PATH` is also set correctly.

Usage

```
ALES_ADMIN_HOME\bin\install_schema_oracle.bat [-s] dbserver dblogin
dbpassword enterprise_domain [policyowner]
```

```
ALES_ADMIN_HOME/bin/install_schema_oracle.sh
```

Options

server
Oracle database server name

dblogin
Login ID, usually same as owner

password
Password for the dblogin user

enterprise_domain
ALES enterprise domain name. Default is `asi`.

policyowner
(Optional) Owner of the policy. Defaults to dblogin it if not provided

-s
Silent mode. Install with no confirmation screen.

Example

```
>install_schema_oracle.bat DBSERVER wles password asi
```

install_schema_sybase

Installs the ALES policy database schema into a Sybase 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.

The `SYBASE` environment variable must be set. The `SYBASE_OCS` environment variable must be set for the Sybase 12 open client. Also make sure your PATH environment variable includes the current directory and the `\bin` and `\dll` subdirectories of the Sybase open client. On UNIX, make sure that `LD_LIBRARY_PATH` is also set correctly.

Usage

```
ALES_ADMIN_HOME\bin\install_schema_sybase.bat [-s] <server> <database>
<dblogin> <dbpassword> <enterprise_domain> [policyowner]
```

```
ALES_ADMIN_HOME/bin/install_schema_sybase.sh
```

Options

dbserver
Sybase database server name

database
Policy database name

dblogin
Login ID, usually same as owner

password
Password for the dblogin user

enterprise_domain
ALES enterprise domain name. Default is `asi`.

policyowner
(Optional) Owner of the policy. Defaults to dblogin it if not provided

-s
Silent mode. Install with no confirmation screen.

Example

```
>install_schema_sybase.bat GODZILLA sspolicy wles password asi
```

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 ARME and BLM process both look into the `password.xml` for the correct password to connect to the ALES database.

If you enable the metadirectory for the ASI Authorization Provider, then remember to set the password for the SSM instance using this tool before restarting the ARME process.

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|restart|shutdown|log|wait|waitready|status] [-msg msg_to_log]
[-reps 1] [-interval 1000] [-?] [-dbg]
```

```
ALES_ADMIN_HOME/bin/asisignal.sh  -url server_url [-action
ping|comtest|restart|shutdown|log|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 restart
Restart the server.

-action shutdown
Shut down the server.

-action log
Send a message for server to log. The text of the message is specified by the `-msg` option.

-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 ARME 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
```

lockdown

Lock down an Administration Server, SCM, or SSM instance with permissions for certain users and groups. It sets the directory permissions based upon the users and groups entered during install. These users and groups are used so that adequate file system security can be enforced for the ALES installation.

Usage

```
ALES_ADMIN_HOME\bin\lockdown.bat

ALES_ADMIN_HOME/bin/lockdown.sh

ALES_SCM_HOME\bin\lockdown.bat

ALES_SCM_HOME/bin/lockdown.sh

ALES_SSM_INSTANCE\adm\lockdown.bat

ALES_SSM_INSTANCE/adm/lockdown.sh
```

When files are changed by users other than `asiadmin/scmuser`, such as root, you should run this tool to change the file owner and groups to the users and group names selected during install (user `asiadmin/scmuser` and groups `asiadgrp/asiusers`). These file permissions need to be updated when you apply a cumulative patch to an existing ALES installation as root.

Example

```
>lockdown.bat
```

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
```

CHAPTER 3

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 C++ language 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.

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

- "Authorization and Role Mapping Extensions" on page 3-2
- "Custom Audit Plug-ins" on page 3-11
- "Database Authentication Plug-in" on page 3-12

asi_classes JAR Required for Provider Extensions

The `asi_classes.jar` 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 3-3
- "Configuring a Custom Attribute Converter" on page 3-8

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 3-1 shows the types of Java-based plug-ins that each security provider supports.

Table 3-1 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).

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

- "Using the Java-based Plug-in Interfaces" on page 3-3
- "Resource Converter" on page 3-4
- "Attribute Retriever" on page 3-6
- "Attribute Converter" on page 3-8

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 3-4
- "Attribute Converter" on page 3-8
- "Attribute Retriever" on page 3-6

2. Use the Java class to create a JAR file and place the JAR file in the `/lib/providers` 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\ales22-ssm\wls-ssm\lib\providers`.
3. 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 3-2 lists and describes the methods provided by the `ResourceConverter` interface.

Table 3-2 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`*`\ales22-ssm\<ssm-type>\lib\providers\ASIAuthorizer.jar`. Include this file, and the *`BEA_HOME`*`\ales22-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 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\ales22-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.
4. Repeat step 3. to register the Resource Converter 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.

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.

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

Table 3-3 AttributeRetriever Interface Methods

Method	Description
`String[] getHandledAttributeNames()`	This method returns the names of attributes handled by this object. An empty or `null` value indicates that the retriever is considered capable of handling any attribute name.
`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.

Configuring a Custom Attribute Retriever

Note: This release of AquaLogic Enterprise Security includes an attribute retriever example, in `BEA_HOME\ales22-ssm\java-ssm\examples\AttributeRetriever`.

To configure a custom attribute retriever, you must implement the attribute retriever and 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` class is in the *`BEA_HOME`*`\ales22-ssm\<`*`ssm-type`*`>\lib\providers\ASIAuthorizer.jar`. Include this file, and the *`BEA_HOME`*`\ales22-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 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\ales22-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.
4. Repeat step 3. to register the Attribute Retriever 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.

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

Table 3-4 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 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\ales22-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.
4. Repeat step 3. to register the Attribute Converter 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.

Using Language Extensions

The ASI Authorization and ASI Role Mapping providers support the use of C++ plug-ins for custom rule extensions for evaluation and credential functions. The functions available for use are described in "Policy Language Custom Extension Library API Reference" on page 5-1.

This section contains a description of how to create an extension library for the ASI Authorization and ASI Role Mapper engine (ARME) used by the ASI Authorization provider. This example works with BEA AquaLogic Enterprise Security, Version 2.2.

The product installation includes an example of code and build commands for an extension library located in the `/examples` subdirectory in the Administration Application installation directory.

For instructions for building, deploying, and using extensions, see the following sections:

- "Building an Extension" on page 3-9
- "Deploying the Extension" on page 3-10

- "Using the Extension" on page 3-11

Building an Extension

To build the extension library, you need to have the following header files, installed in the `asi` subdirectory:

- armeapi.h
- AttributeValue.h
- session.h
- defs.h
- exception.h

The extension needs to be linked with the following libraries, included with the ARME executable:

- utilmd.lib or libutil.so
- pluginmd.lib or libplugin.so

The compiler used must generate library binaries compatible with the compiler used to compile the ARME server.

- On Windows platform: Microsoft Visual Studio .NET 2003
- On Linux Red Hat, AS 2.1 or 3.0: gcc 2.96
- On Solaris: Sun Forte 9, or a binary compatible mode for later versions

Deploying the Extension

To deploy the extension, do the following:

1. Place the compiled library (for example, `arme_extension.dll`) into the same path accessible by the ARME process.
2. Configure the initialization function in the ARME local configuration file, using the following syntax:

   ```
   <ARME.tag>.plugin[1..4] <path>/<DLL plug_in filename>(initialize 'arg')
   ```

   For example:

   ```
   ARME.alesadmin.plugin1 arme_extension.dll(initialize 'test')
   ```

This example assumes that the `arme_extension.dll` is in the path for the ARME process. In this example, `initialize` is the name of the routine in the extension library that is called when the library is loaded to perform initialization. `<ARME.tag>` is a parameter passed in the command line of the ARME process. This parameter defines the scope for the configuration parameters used from a local file. You may use an empty scope for these keywords; that is, just `plugin[1..4]`.

Using the Extension

The extension library adds credential functions (custom dynamic attributes) and evaluation functions. To use them in your policy, you need to add declarations for them. For example, if an extension library defines the `custom-attribute` credential function, you need to add a declaration for a `custom-attribute` in the Administration Console with a dynamic type, and an appropriate data type (string, integer, and so on.). Then, you can use this attribute in policy constraints.

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 3-11
- "Audit Plug-in Renderer Class" on page 3-12

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 3-12 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 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\ales22-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 3-5 lists and describes a sample `AuditEventRenderer` class.

Table 3-5 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 implement the `DBMSPlugin` Interface (located in the `com.bea.security.providers.authentication.dbms.DBMSPlugin` package). The `DBMSPlugin` Interface implementation must include the methods described in Table 3-6.

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 and use the Administration Console to configure the Database Authentication provider to use the plug-in.

Table 3-6 lists and describes the methods provided by `the DBMSPlugin` interface.

Table 3-6 DBMSPlugin Interface 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 3-6 DBMSPlugin Interface 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`".

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 4

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 4-1. Table 4-1 also indicates which sub-interfaces implement the `AuditContext` interface. These interfaces are documented in the *BEA AquaLogic Enterprise Security Provider SSPI API Reference*.

Table 4-1 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 4-2 lists WebLogic 9.x audit events.

Table 4-2 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 4-2 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

The providers implement the appropriate `AuditEvent` interfaces and post those events to the Audit provider. 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 4-1.

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 ARME 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

In this release of AquaLogic Enterprise Security, 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 data structure 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 seven major types of audit events:

- AuditAtnEvent
- AuditAtzEvent
- AuditMgmtEvent
- AuditCredentialMappingEvent

- AuditPolicyEvent
- AuditRoleDeploymentEvent
- AuditRoleEvent

For a list of the events that are audited for the default Admin policy, see "Admin Policy Audit Events" on page 4-12.

AuditAtnEvent

Authentication audit events are posted by the security framework. Table 4-3 describes the conditions under which the event is posted and severity level of the event.

Table 4-3 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
Security Framework	Posted after an unsuccessful identity assertion (when username returned from identity assertion callback handler is null or zero length).	Failure

Table 4-3 Authentication Audit Events (Continued)

Component	Description	Severity
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

AuditAtzEvent

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

Table 4-4 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
Security Framework	Posted if access is not allowed to resource.	Failure

AuditCredentialMappingEvent

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

Table 4-5 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 4-6 describes the conditions under which the events are posted and lists the event severity level.

Table 4-6 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
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

AuditRoleDeploymentEvent

The security framework posts audit role deployment events when roles are deployed to or undeployed from a role mapping provider. Table 4-7 describes the conditions under which the events are posted and lists the event severity level.

Table 4-7 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
Security Framework	Posted after each unsuccessful role undeployment from a role mapping provider.	Failure

AuditRoleEvent

The WebLogic Role Mapping provider posts audit role events when roles are created, deleted, or updated. Table 4-8 describes the conditions under which the events are posted and lists the event severity level.

Table 4-8 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 4-9 lists and describes the administration policy events that are audited.

Table 4-9 Admin Policy Audit Events

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

Table 4-9 Admin Policy Audit Events (Continued)

Policy Element	Action	Type	Event Description
Declaration/Evaluation Function	create	declaration	Create an evaluation function.
	delete	declaration	Delete an evaluation function.
	rename	declaration, new_name	Rename an evaluation function.
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.

Table 4-9 Admin Policy Audit Events (Continued)

Policy Element	Action	Type	Event Description
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.
	cascade Delete	subject_name	Cascade a user and all policies associated with the user.
	rename	subject_name, new_subject_name	Rename a user.
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.
	addMemb er	subject_name, member_subject	Add a member to a group.
	remove Member	subject_name, member_subject	Remove a member from a group.
Identity/Subject/ Attribute Assignment/ Single	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.

Table 4-9 Admin Policy Audit Events (Continued)

Policy Element	Action	Type	Event Description
Identity/Subject/ Attribute Assignment/List	create	attribute, value, subject_name	Set a value to a currently unset vector subject attribute.
	delete	attribute, value, subject_name	Unset a currently set vector subject attribute.
	modify	attribute, value, subject_name, new_value	Modify the value of a currently set vector 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.
	cascade Delete	resource	Cascade delete of a resource. This includes deletion of all child resources and associated policies.
	rename	resource, new_name	Rename a resource.
Resource/Attribute Assignment/Single	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.

Table 4-9 Admin Policy Audit Events (Continued)

Policy Element	Action	Type	Event Description
Resource/Attribute Assignment/List	create	attribute, resource, value	Set a value to a currently unset vector resource attribute.
	delete	attribute, resource, value	Unset a currently set vector resource attribute.
	modify	attribute, resource, value, new_value	Modify the value of a currently set vector resource attribute
Resource/Metadata/ IsApplication	modify	resource, value, new_value	Toggle the “is application” resource metadata.
Resource/Metadata/ IsDistributionPoint	modify	resource, value, new_value	Toggle the “is distribution point” resource metadata.
Resource/Metadata/ Logical Name	create	logical_name, resource	Create a logical name for a resource.
	delete	logical_name, resource	Delete the logical name of a resource.
	rename	logical_name, resource, new_name	Rename the logical name of a resource.
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, new_action, new_resource, new_subject_name, new_constraint	Modify a grant policy. The “action”, “resource”, and “subject_name” attributes are lists.

Table 4-9 Admin Policy Audit Events (Continued)

Policy Element	Action	Type	Event Description
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, action_type, resource, subject_name, subject_type, constraint, new_effect, new_action, new_action_type, new_resource, new_subject_name, new_subject_type, new_constraint	Modify a deny policy. The “action”, “resource”, and “subject_name” attributes are lists.
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, delegator, constraint, new_action, new_resource, new_subject_name, new_delegator, new_constraint	Modify a delegate policy. The “action”, “resource”, and “subject_name” attributes are lists.

Table 4-9 Admin Policy Audit Events (Continued)

Policy Element	Action	Type	Event Description
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.
Policy/Analysis/ Inquiry Query	create	title, owner, effect_type, subjects, actions, resources, delegator	Create a new policy query.
	delete	title, owner	Delete a policy query.
	modify	title, owner, effect_type, subjects, actions, resources, delegator	Modify a policy query.
	execute	title, owner, effect_type, subjects, actions, resources, delegator	Execute a policy query. If this is an unsaved query “title” and “owner” is set to an emptystring.

Table 4-9 Admin Policy Audit Events (Continued)

Policy Element	Action	Type	Event Description
Policy/Analysis/ Verification Query	create	title, owner, actions, resources	Create a new policy verification query.
	delete	title, owner	Delete a policy verification query.
	modify	title, owner, actions, resources	Modify a policy verification query.
	execute	title, owner, actions, resources	Execute a policy verification query. If this is an unsaved query “title” and “owner” is set to an emptystring.
Policy/Repository	deploy Update	resource, directory	Deploy a policy update. The “resource” is the distribution node; all nodes below it may be effected. 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 /ARME	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 4-9 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.
Infrastructure/ Management/Console	login		Login to the ALES administration console.
Infrastructure/ Management/BulkManager	login		Login to the ALES policy loader.

Additional Audit Event Interfaces

The following sections describe additional audit event interfaces:

- "ContextHandler Object" on page 4-24
- "PolicyAdministrationEvent" on page 4-25

Authentication - AuditAtnEvent

The `AuditAtnEvent` interface provides an interface for audit providers to determine the instance types of the extended authentication event type objects. Table 4-10 describes the event properties.

Table 4-10 Authentication - AuditAtnEvent

Event 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.

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`
- Authorization - `AuditAtzEvent`

There are both pre- and post-authorization access control checks; each of which generates pre- and post-operation audit write events. The `AuditAtzEvent` event interface is used to report events that result when access is allowed on a resource. The Audit Channel provider is called on both the pre- and post-operation cases. The exceptions reported using this event must derive from the `java.security.GeneralSecurityException`.

When this event is generated, the following information associated with this `AuditAtzEvent` is 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 Access Decision to obtain information associated with the context of the request. This `ContextHandler` object is also available with this `AuditAtzEvent`. For more information about the `ContextHandler` object, see "ContextHandler Object" on page 4-24.

Policy Deployment - AuditPolicyDeployEvent

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

- The subject whose action is being audited
- The severity of this event
- The resource that is the target of action being audited
- A string array of role names for this policy
- 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.

Policy Undeployment - AuditPolicyUndeployEvent

The `AuditPolicyUndeployEvent` event interface is used when the Authorization Manager `undeployPolicy` method is called. When this event is generated, the following information is available:

- The subject whose action is being audited
- The severity of this event
- The resource that is the target of action being audited
- A string array of role names for this policy

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

Policy Events - AuditPolicyEvent

The `AuditPolicyEvent` event interface determines the instance types of extended Authorization event type objects. Table 4-11 describes the event subtypes.

Table 4-11 Policy Event- AuditPolicyEvent

Event Subtype	Description
DEPLOY	Indicates that a policy deployment event occurred.
UNDEPLOY	Indicates that a policy undeployment event occurred.
UPDATE	Indicates that a policy was updated.

Role Mapping - AuditRoleEvent

The `AuditRoleEvent` event provides an interface for auditing providers to determine the instance types of extended Role Mapping event type objects. Table 4-12 describes the event subtypes.

Table 4-12 Role Mapping - AuditRoleEvent

Event Subtype	Description
DEPLOY	Indicates that a role mapping deployment event occurred.
UNDEPLOY	Indicates that a role mapping undeployment event occurred.
UPDATE	Indicates that a role mapping was updated.

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`
- The `ContextHandler` object

The resource container that handles the type of resource being requested (for example, with WebLogic Server, the EJB container receives the request for an EJB resource) constructs a `ContextHandler` object that may be used by an Authorization provider Access Decision to obtain information associated with the context of the request. This `ContextHandler` object is also available with this `AuditAtzEvent`. For more information about the `ContextHandler` object, see "ContextHandler Object" on page 4-24.

Role Deployment - AuditRoleDeployEvent

The `AuditRoleDeployEvent` event provides a interface used by the role mapping service to determine the instance types of extended Role Mapping deployment event type objects.

Role Undeployment - AuditRoleUndeployEvent

The `AuditRoleUndeployEvent` event provides a interface used by the role mapping service to determine the instance types of extended Role Mapping undeployment event type objects.

Predicate Events - AuditPredicateEvent

The `AuditPredicateEvent` event provides a interface for 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 4-13 describes the event subtypes.

Table 4-13 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 `ContextElement` 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.

PolicyAdministrationEvent

The `PolicyAdministrationEvent` event is used when AquaLogic Enterprise Security policy is modified or deployed using the AquaLogic Enterprise Security Administration console or bulk loader. When this event is generated, the following information is available:

- The subject whose action is being audited
- The severity of this event
- The resource that is the target of action being audited
- Detailed information regarding the change being made

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.

Policy Language Custom Extension Library API Reference

The ASI Authorization and Role Mapper providers use an external server process to evaluate authorization decisions. This process is called the Authorization and Role Mapping Engine (ARME).

This section describes the Application Programming Interface (API) for writing custom extension libraries (plug-ins) for this process to enhance features available through the policy language, such as routines for dynamic computation of an attribute value (credential function) or custom predicate (evaluation function).

This section covers the following topics:

Plug-In Extension Function Pointers

The following plug-in function pointers are described in this API:

- “registerCustomCredentialFunction() - Register Credential Function” on page 5-6
- “registerCustomEvaluationFunction() - Register Evaluation Function” on page 5-6
- “registerShutdownFunction() - Register Shutdown Function” on page 5-7

These extension functions (called plug-ins) take two kinds of parameters: required and optional. Parameters also have one of two modes: in or out. Input parameters pass data to an extension. Output parameters receive data from an extension.

*CredFunc() - Custom Credential Function Pointer

Description

Pointer to a function that computes the value of a credential function. Your plug-in function is internally referenced by this pointer. The `registerCustomCredentialFunction()` function assigns the pointer to your function. For example, this function declares a credential function:

```
bool MyCredentialFunction (Session &sess, const char *cvarname)
```

The ARME then connects a function pointer (assign a handle) to your function:

```
returnCode = registerCustomCredentialFunction( “MyCredentialFunctionsName”,
MyCredentialFunction);
```

Syntax

```
bool (*CredFunc)(Session &sess, const char *cvarname);
```

Parameters

`*cvarname` is a required input parameter, that points to a null-terminated character string (a character array) and contains the name of a credential function.

`&sess` is a required output parameter that references the address of a Session object and a session that contains the computed value of the credential function.

Returns

`TRUE` if the computation is successful

`FALSE` if the computation is unsuccessful

Example

```
bool GetAccountID(Session &sess, const char *cvarname)

{
...
}
```

See Also

`*EvalFunc()`

*EvalFunc() - Custom Evaluation Function Pointer

Points to a function that computes the value of an evaluation function. Your plug-in function is internally referenced by the ARME through this pointer. The `registerCustomEvaluation Function()` function assigns the pointer to your plug-in function. For example, you could declare a evaluation function like this:

```
TruthValue MyEvalFunction(Session &sess,const char *fname, char **argv);
```

The ARME then connects a function pointer (assign a handle) to your function:

```
returnCode = registerCustomEvaluationFunction(“MyEvaluationFunctionsName“,
MyEvalFunction);
```

Syntax

```
TruthValue (*EvalFunc)(Session &sess,const char *fname, char **argv);
```

Parameters

`&sess` is a required output parameter that references the address of a `Session` object and a session that contains the computed value of the credential variable.

`*fname` is a required input parameter that points to a null-terminated character string (a character array) and contains the name of an evaluation function.

`**argv` is a required input parameter and is a pointer to a null-terminated array (also a pointer) of null-terminated strings (character arrays), containing a list of arguments for the function.

Returns

`TruthValue` enumerated list, containing one of three constants:

`TV_TRUE` is true

`TV_FALSE` is false

`TV_UNKNOWN` is an error or undefined result

Example

```
TruthValue isValidAccountID(Session &sess, const char* fname, char **argv)
{...}
```

See Also

`*CredFunc()`

*ShutdownFunc () - Custom Shutdown Function Pointer

Points to a function that is called when the ARME is about to be shutdown. Put all cleanup code inside this function that you want the plug-in to perform before the shutdown; for example, open windows handlers, open file pointers, and database connections. Your plug-in function is internally referenced by the ARME by this pointer. The `registerShutdownFunction()` function assigns the pointer to your plug-in function. For example, to declare a shutdown function, use the following:

```
MyShutdownFunction ()
```

Assign a handle to your function to connect the ARME to your function pointer:

```
returnCode = registerShutdownFunction(MyShutdownFunction);
```

Syntax

```
void (*ShutdownFunc)();
```

Parameters

None

Returns

Nothing

Example

```
void MyShutdownFunction()
```

```
{
// Code that needs to run for the plugin to release
// any resources or open connections that it created
}
```

See Also

`*PluginInitFunc()`

*PluginInitFunc() - Plug-in Initialization Function Pointer

Points to a function that initializes an ARME plug-in. The name of this function must appear in the ARME configuration file. For example, for the following statement, include the function name `initMyplugins()` in the ARME configuration file.

```
ARME.Instance1.plugin1 F:/BEA/MyPlugins/ Plugin01.dll(initMyplugins
'plugin1')
```

Syntax

```
void (*PluginInitFunc)(const char *argp);
```

Parameters

`*argp` is a required input parameter and is a pointer to a null-terminated array of null-terminated character strings that contains a list of arguments for the initialization.

Returns

Nothing

Example

```
void initPlugin(const char *pluginName)
{
...
}
```

registerCustomCredentialFunction() - Register Credential Function

Registers a credential function with a ARME.

Syntax

```
bool registerCustomCredentialFunction(const char *credname,CredFunc
credfunc);
```

Parameters

`*credname` is a required input parameter and points to a null-terminated character string that contains the name of a custom credential function.

`credFunc` is a required input parameter used to register the credential function.

Returns

`TRUE` if the registration is successful

`FALSE` if the registration is unsuccessful

Example

```
returnCode = registerCustomCredentialFunction("MyCredentialFunctionsName",
MyCredentialFunction);
```

See Also

`registerCustomEvaluationFunction()`

registerCustomEvaluationFunction() - Register Evaluation Function

Registers a custom evaluation function with a ARME.

Syntax

```
bool registerCustomEvaluationFunction(const char *evalname, CredFunc
evalfunc);
```

Parameters

`*evalname` is a required input parameter and points to a null-terminated character string that contains the name of an evaluation function.

`evalFunc` is a required input parameter used to register the evaluation function.

Returns

`TRUE` if the registration is successful

`FALSE` if the registration is unsuccessful

Example

```
returnCode = registerCustomEvaluationFunction(“MyEvaluationFunctionsName”,
MyEvalFunction);
```

See Also

`registerCustomCredentialFunction()`

registerShutdownFunction() - Register Shutdown Function

Registers a shutdown function.

Syntax

```
bool registerShutdownFunction(ShutdownFunc shutdownfunc);
```

Parameters

`ShutdownFunc` is a required input parameter used to register the function.

Returns

`TRUE` if the registration is successful

`FALSE` if the registration is unsuccessful

Example

```
returnCode = registerShutdownFunction(MyShutdownFunction);
```

See Also

`registerCustomEvaluationFunction()`

`registerCustomCredentialFunction()`

Session Class

A session object keeps all data related to a single access or role query. All attributes used during evaluation are accessible through the session object.

Subject and resource attributes are always accessible. A dynamic attribute, either computed by an evaluation function or passed in, may be accessible, but this is not guaranteed as they are loaded as needed. A attribute used as an argument for an evaluation function is computed before that function is invoked.

The following session objects are described in this API:

Session::SetAttribute() - Append AttributeValue Object

Sets an attribute value for a credential attribute by appending an `AttributeValue` object to the named credential.

Syntax

```
bool setAttribute(const char *name, AttributeValue *value, bool overwrite)
```

Parameters

`*name` is a required input parameter that points to a null-terminated character string (a character array), referencing the name of the credential attribute. This parameter must be passed a `const char *` and not as a `char *` variable. Simply placing the name directly into the function (in quotation marks) does not work.

`*value` is a required input parameter that points to an `AttributeValue` object and contains the value of the credential. Allocate this object with a new operator because the session object takes ownership of the object.

WARNING: Passing in a pointer to a local object or deleting this pointer inside your plug-in may lead to unexpected results, most likely terminating the ARME process.

`overwrite` is a required input parameter Boolean value (`TRUE` or `FALSE`), that determines whether to overwrite existing attributes of the same name. During evaluation, values set with this call take precedence over static or dynamic values defined for the same credential attribute. There are no safeguards to prevent modifying the values of these credential attributes. You should set the value of a credential function only for the attribute to which it is registered.

Returns

`TRUE` if operation was successful

`FALSE` if it was not successful

Example

```
AttributeValue* SomeAttributeValue = new AttributeValue(false);

const char *CredName = “SongCount”;

const char *Value = “small“;

SomeAttributeValue.setValue(Value);

retcode = sess.setAttribute(CredName, SomeAttributeValue, true);
```

WARNING: Some `AttributeValue` methods may cause an exception and memory leak if you call them before calling `setAttribute()`.

See Also

`getAttribute()`

Session::getAttribute() - Get AttributeValue Object from Attribute

Gets an attribute value for a credential attribute by requiring that you pass it the address of an existing `AttributeValue` object. You can use `getAttribute("my_roles", SomeAttributeValue)` to retrieve a list containing the role for a subject.

WARNING: BEA strongly recommends that you not modify the contents of the `AttributeValue` object. Some `AttributeValue` objects are shared between credentials objects and evaluation threads. Manipulating their values can lead to unexpected results.

Syntax

```
bool getAttribute(const char * name, AttributeValue *& value)
```

Parameters

`*name` is a required input parameter that points to a null-terminated character string (a character array) and references the name of a credential attribute. You must pass this parameter as a `const char *` and not a `char *` variable. Simply putting the name directly into the function in quotation marks does not work.

`*&value` is a required input parameter that points to an `AttributeValue` object and contains the value of the attribute.

Returns

`TRUE` if operation was successful

`FALSE` if it was not successful

Example

```
const char *inputvalue = "LillyLiverCount";

retcode = sess.getAttribute(inputvalue,SomeAttributeValue);
```

See Also

`setAttribute()`

Session::getEvalResult() - Get Evaluation Result

Returns a pointer to a temporary `EvalResult` object containing the data returned, along with the access decision upon the execution of the policy containing the evaluation or credential function. The plug-in does not know in advance if the policy containing the plug-in function executes (other policies that preclude it may execute first), so modifying this object does not guarantee the information is returned. BEA recommends using the `appendReturnData()` method. In many cases, you may find using the `report()` and `report_as()` functions easier and more flexible.

Syntax

```
EvalResult* getEvalResult();
```

Parameters

None

Returns

Pointer to an `EvalResult` object containing the evaluation results (output attributes) that may be returned by the executing policy.

Example

```
EvalResult *MyEvalResultPtr = getEvalResult();
```

See Also

`getEnumValue()`

Session::appendReturnData() - Return Evaluation Results

Sets the evaluation results returned through an `EvalResult` object upon successful execution of a policy containing the evaluation or credential function that references the plug-in function. It does so by copying the contents of the `AttributeValue` object that is passed to it.

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

The evaluation result is returned only if the policy actually executes. It might not 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 only looks for deny policies. When a single deny policy is found, it stops looking.

The exception to this is if the `findAllFacts` flag is enabled, which you have to consciously set. When you set the `findAllFacts` flag to true, you are telling BEA AquaLogic Enterprise Security to return the policy attributes for all policies affecting the access attempt, whether or not the policies are redundant. Setting the `findAllFacts` flag to true does change the fact that once a deny policy has executed, grant policies are not evaluated. The sole purpose is to evaluate and possibly append return results from all deny policies or all grant policies if no deny policies fired.

WARNING: Setting the `findAllFacts` flag to true dramatically slows down each access attempt because all policies are checked every time. Avoid setting the flag to true, unless there is no other way to accomplish your goal.

Each `EvaluationResult` object contains the relevant policy identification number, the name of the object and privilege in that policy, and a collection of return attributes in list form. Single values are represented as single-value lists.

WARNING: Unlike the `setAttribute()` method, this method copies the content of the data. If you allocate the `AttributeValue` object with the new operator, you have to delete the object. If you do not, a memory leak may result. You are free to pass in a pointer to a local variable, like that retrieved by the `getAttribute()` method.

Syntax

```
void appendReturnData(const char *name, const AttributeValue *data);
```

Parameters

`*name` is a required input parameter that points to a null-terminated character string (a character array) and references the name of the output credential attribute. This parameter must be passed a const char * and not a char * variable. Simply putting the name directly into the function in quotation marks does not work.

`*data` is a required input parameter that points to an `AttributeValue` object and contains the value of the attribute.

Returns

Nothing

Example

```
AttributeValue LocalValue(true);

const char *OutputAttributeName = "SomeCount";

const char *ValueSample = "shopping";

LocalValue.addValue(ValueSample);

appendReturnData(OutputAttributeName, &LocalValue);
```

See Also

```
getEvalResult()
```

Session::getDomainName() - Get Domain Name for the Session

Retrieves the name of the domain for the session object to which it belongs.

Syntax

```
const char *getDomainName() const;
```

Parameters

None

Returns

Points to a null-terminated character string (a character array) and references the name of the session domain.

Example

```
const char *CLDomainNameStr = getDomainName();
```

See Also

```
getLocationName()
```

Session::getLocationName() - Get Location Name for Session

Retrieves the name of the location for the session object to which it belongs.

Syntax

```
const char *getLocationName() const;
```

Parameters

None

Returns

Points to a null-terminated character string (a character array) and references the name of the session location.

Example

```
const char *CLLocationNameStr = getLocationName();
```

See Also

`getDomainName()`

Session::getApplicationName() - Get Application Name for Session

Retrieves the name of the application for the session object to which it belongs.

Syntax

```
const char *getApplicationName() const;
```

Parameters

None

Returns

Points to a null-terminated character string (a character array) and references the name of the session application.

Example

```
const char *CLAppNameStr = getApplicationName();
```

See Also

`getUserID()`

Session::getUserID() - Get User Name for Session

Retrieves the name of the user for the session object to which it belongs.

Syntax

```
const char *getUserId() const;
```

Parameters

None

Returns

Points to a null-terminated character string (a character array) and references the name of the session user.

Example

```
const char *CLUserNameStr = getUserId();
```

See Also

`getApplicationName()`

AttributeValue Class

Contains the methods for manipulating `AttributeValue` objects, which is how attribute values are stored for each user credential attribute.

This section describes the following `AttributeValue` objects:

`AttributeValue` objects store values in one of two ways:

- As a single string value
- As an array-like list of strings

Single Value

Single values are nothing more than one string of characters per `AttributeValue` object and are the most common type of attribute value. The methods that are exclusively for manipulating single values are:

`setValue()`

`getValue()`

Lists of Values

Lists of values are collections of strings for one `AttributeValue` object. They are accessed by their index value (in square brackets), just like a one-dimensional array. Normally, list values are not set with duplicate string values. The methods that are exclusively for manipulating lists of values are:

`addValue()`

`removeValue()`

`removeAt()`

`entries()`

There is also an undefined type of `AttributeValue` object created by a default constructor with no arguments. It becomes either a single or a list type after the first call to the `setValue()` method (single), `addValue()` method (list), or `operator=()`.

Methods Common to Both Types

Several methods are available for both single and list attributes. Most of them are for testing what type of attribute you have. These include:

```
size() (returns 1 for single value, or entries() for list value)

       operator[0] (same as getValue() for index 0), or operator[] for list
type

isList()

isSingle()

isUndefined()

       has()
```

Internal Methods

The header files contain many methods related to this class and ones that are not documented. These are used internally by the ARME and are not recommended for use. BEA cannot guarantee these methods work at all or will work in the future due to deprecation. These include `operator+=()`, `operator-=()`, all those taking non-const `char*`, enumeration-related, and type-checking-related methods.

WARNING: BEA does not recommend using any method not documented in this guide.

AttributeValue::addValue() - Add and Set a String List Attribute Value

Adds and sets a string value to a string list `AttributeValue` object. Use `isList()` to determine if the object contains a list of values. If you call this method for an undefined `AttributeValue` object, it becomes a list type.

Syntax

```
void addValue(const char *value, bool check_unique = false)
```

Parameters

`*value` is a required input parameter that points to a null-terminated character string (a character array). This parameter adds an element to the end of a string list increasing the list index value by one and must be passed a `const char *` and not a `char *` variable. Simply putting the value directly into the function in quotation marks does not work.

`check_unique` is an input parameter that defaults to false. If this parameter is set to `TRUE`, duplicate values are not added to the list.

Returns

Nothing

Example

```
const char *FavoriteSongValue = “foreveryours“;

if (FaveSongAvObj.isList())

FaveSongAVObj.addValue(FavoriteSongValue);
```

See Also

`setValue()`

`getValue()`

`isList()`

AttributeValue::AttributeValue() - Constructor

The `AttributeValue` class constructor is polymorphic. You can pass it different sets of parameters depending on your goals.

Syntax

For an undefined attribute:

```
AttributeValue();
```

For a single-value attribute:

```
AttributeValue(const char *value);
```

For an empty list of values or a single value:

```
AttributeValue(bool isList);
```

To copy an existing `AttributeValue`:

```
AttributeValue(const AttributeValue &value);
```

Parameters

`*value (String)` is a required input parameter used to create a single-value attribute that points to a null-terminated character string (a character array). This parameter references the name of a credential attribute and must be passed a `const char *` and not a `char *` variable. Simply putting the value directly into the function in quotation marks does not work.

`&value (AttributeValue Object)` is a required input parameter (for copying an `AttributeValue` object) that points to an `AttributeValue` object.

`isList` is set to `TRUE` if the attribute and contains an enumerated list of strings; set to `FALSE` for a single value attribute.

Returns

Nothing

Example

For a single-value attribute:

```
const char *MyAttrName = "SomeCount";

AttributeValue *MyAttrValPtr = new
```

```
AttributeValue(MyAttrName);
```

For a list of values:

```
AttributeValue *MyAVListPtr = AttributeValue(true);
```

To copy an existing `AttributeValue` object:

```
AttributeValue *MyAttrValPtr = new

AttributeValue(MyExistingAttrValObj);
```

See Also

`setValue()`

AttributeValue::entries() - Count Number of List Elements

Returns a count of the total number of elements in an indexed list.

Syntax

```
int entries() const
```

Parameters

None

Returns

An integer that contains the total count of elements.

Example

```
if (FaveSongAvObj.isList())int elementCount = FaveSongAVObj.entries();
```

See Also

`addValue()`

`AttributeValue::size()`

AttributeValue::getValue() - Get Single Attribute Value

Gets a single string value for an `AttributeValue` object. Use `isSingle()` to determine if the object contains a single value.

Syntax

```
const char *getValue() const
```

Parameters

None

Returns

A pointer to a constant, null-terminated character string (a character array) that contains the string value of the `AttributeValue` object. This value must be a `const char *` and not a `char *` variable.

Example

```
const char *FavoriteSongValue;

if (FaveSongAvObj.isSingle())FavoriteSongValue =

FaveSongAVObj.getValue();
```

See Also

`addValue()`

`setValue()`

`isSingle()`

AttributeValue::has() - Check If Value is Already Present in a List

Checks if a value is present in a list or if a single-value attribute is equal to it. `AttributeValue` objects of the list type should not contain redundant values.

Syntax

```
bool has(const char * value) const
```

Parameters

`*value`

is a required input parameter

Points to a null-terminated character string (a character array)

Returns

`TRUE` if the supplied value is present in a list, or is equal to the value of a single type value.

Example

```
Bool A = AttributeValue.has(“SomeValue”);
```

See Also

`addValue()`

`setValue()`

`AttributeValue::size()`

AttributeValue::IsList() - Is Attribute Value an Indexed List?

Returns `TRUE` if the `AttributeValue` object contains a list of strings or integers. The list attribute values are accessed by their index value.

Syntax

```
bool isList() const;
```

Parameters

None

Returns

`TRUE` if the attribute contains a list of values

Example

```
bool AttrIsList = MyAttrValueObj.isList();
```

See Also

`isSingle()`

AttributeValue::IsSingle() - Is Attribute Value a Single Value?

Returns `TRUE` if the `AttributeValue` object contains a single value and not a list of values.

Syntax

```
bool isSingle() const;
```

Parameters

None

Returns

`TRUE` if the attribute is a single-value type

Example

```
bool AttrIsSingle = MyAttrValueObj.isSingle();
```

See Also

`isList()`

AttributeValue::isUndefined() - Is Attribute Value an undefined object?

Returns `TRUE` if the `AttributeValue` object is constructed with a default constructor and does not have a list or single type. The type can be set by calling `setValue()`, or `addValue()` methods and becomes a single value or a list correspondingly, or by using an assignment operator. Once set, the type cannot be changed.

Syntax

```
bool isUndefined() const;
```

Parameters

None

Returns

`TRUE` if the attribute is undefined

Example

```
bool AttrIsSingle = MyAttrValueObj.isUndefined();
```

See Also

`isList()`

AttributeValue::setValue() - Set Single Attribute Value

Sets a single string value for an `AttributeValue` object. Use `isSingle()` to determine if the object is the correct type for a single value. If you call this method for an undefined `AttributeValue` object, it becomes a single type.

Syntax

```
void setValue(const char *value)
```

Parameters

`*value` is a required input parameter that points to a null-terminated character string (a character array) and sets the string value of an `AttributeValue` object. This parameter must be passed a `const char *` and not a `char *` variable. Simply putting the value directly into the function in quotation marks does not work.

Returns

Nothing

Example

```
const char *FavoriteSongValue = "foreveryours";

FaveSongAVObj.setValue(FavoriteSongValue);
```

See Also

`addValue()`

`getValue()`

AttributeValue::removeAt() - Remove Indexed List Attribute Value

Removes a value identified by index in a string or integer list `AttributeValue` object. Use `isList()` to ensure that it is a list of values.

Syntax

```
char *removeAt(int idx)
```

Parameters

`idx` is a required input parameter with an integer value that removes a string or integer element in the list by the index value of the element (index starts at 0).

Returns

A pointer to the string representing the value. This is provided as a convenience to use the value so you do not have to retrieve the value with a separate call.

WARNING: You must delete the returned value with the `delete[]` operator within the plug-in.

Example

```
//Remove the second value, which is “Apple”:

char * MrValue = someAttrValuePtr->removeValue(1);

//MrValue now points to "Apple" if the index position

//contained such value, to null if not..

//Delete value, even if null:

delete[] MrValue;
```

See Also

`addValue()`

`removeValue()`

AttributeValue::removeValue() - Remove Named List Attribute Value

Removes a named element from a string list `AttributeValue` object. `Use isList()` to determine if the object contains a list of values.

Syntax

```
char *removeValue(const char *value)
```

Parameters

`*value` is a required input parameter that points to a null-terminated character string (a character array) and identifies an element in the list by the value of the element. This parameter must be passed a `const char *` and not a `char *` variable. Simply putting the value directly into the function in quotation marks does not work.

Returns

A pointer to the sting representing the value. This is provided as a convenience to use the value so you do not have to retrieve the value with a separate call.

WARNING: To avoid a memory leak, you must delete the returned value with the `delete[]` operator within the plug-in.

Example

```
char * MrValue = someAttrValuePtr->removeValue("Apple");

//MrValue now points to "Apple" if someAttrValuePtr

//contained such value, to null if not..

//Delete value, even if null:

delete[] MrValue;
```

See Also

`addValue()`

`removeAt()`

`isList()`

AttributeValue::size() - Count Number of List Elements

This method returns a count of the total number of elements in an indexed list and returns the value 1 for single-type attributes. That is, for lists, it is equivalent to `entries()`.

Syntax

```
int entries() const
```

Parameters

None

Returns

An integer value that contains the total count of elements.

Example

```
   if (FaveSongAvObj.isList())int elementCount = FaveSongAVObj.size();
```

See Also

`addValue()`

`AttributeValue::size()`

AttributeValue [ ] Operator - Returns the Value of an Indexed String List Element

Retrieves the value of a specific list element by integer index, use the `[]` overloaded operator. For example, to retrieve the third element in the `Colors` string list, use this syntax:

```
const char *FavoriteColor = Colors[2];
```

You can use `isList()` beforehand to determine if the object contains a list of values and `entries()` or `size()` to ensure the index is valid. This method also works with single-type object. In such a case, the index is 0. When used in this way, the method is equivalent to the `getValue()` method.

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. Use Metadirectory on the Metadirectory tab must also be enabled in order to make use of user attributes.	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
UseMetadirectory	"false"	When enabled the authorization and role mapping providers access the metadirectory to gather additional user profile data. If disabled none of the other metadirectory settings has any effect.	N
RoundRobinMetadirectories	"false"	When enabled a round robin algorithm is used in accessing the configured metadirectories, otherwise they are accessed in the order listed.	N
sacTimeoutSec	"15"	Time in seconds after which a previously failed metadirectory server is retried.	N
dbServer		Defines the metadirectory database servers. If a failure occurs when accessing the first server, then the next one is used.	Y
dbSystem	"ORACLE"	Defines the type of the metadirectory database.	N
dbName		The name of the metadirectory database. Only applicable when using Sybase.	N

Table 6-6 AsiAuthorizer (Continued)

Attribute Name	Default Value	Description	List
dbLogin		Database login for the metadirectory. The password for this user must be stored in the encrypted password file within the Security Service Module installation.	N
dbpoolsize	"10"	The database pool size that the authorization and role mapping provider use to access the metadirectory	N
dbConnIdleTimeout	"600"	Idle timeout in seconds for metadirectory database connection.	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
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