Oracle® Entitlements Server 10*g* (10.1.4.3)

Administrative Reference

September 2008

ORACLE

ProductName 10*g* (10.1.4.3)

1. Introduction and Roadmap

2. Administrative Utilities

3. Import/Export Utilities

4. Configuration Files

5. WLESblm.conf Reference

6. Out-of-Box Attribute Retrievers

7. Provider Extensions

8. Audit Events

9. BLM Configuration API Security Providers Reference

CHAPTER 1

Introduction and Roadmap

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

Note: Oracle Entitlements Server was previously known as BEA Aqualogic Enterprise Security. Some items, such as schema objects, paths, and so on may still use the term “ALES."

Scope

This document provides instructions for using Oracle Entitlements Server administrative utilities, provides information about writing provider extensions, and describes how to use the Custom Extensions API.

Guide to this Document

This document is organized as follows:

- Chapter 2, “Administrative Utilities,” provides a reference for the command-line administrative utilities provided with OES.
- Chapter 3, “Import/Export Utilities,” provides instructions for importing and exporting data from the OES policy store.

- Chapter 4, “Configuration Files,” provides information about confiuration files provided when the Administration Server and SSMs are installed.
- Chapter 5, “WLESblm.conf Reference,” describes the configuration parameters in the `WLESblm.conf` configuration file. You can edit this file to configure and tune this product after installation.
- Chapter 6, “Out-of-Box Attribute Retrievers,” describes the OOTB attributes retrievers that can be used to return attributes values from external data sources. These values can then be used to determine policy outcomes.
- Chapter 7, “Provider Extensions,” describes how to write custom security provider extensions. A provider extension is a plug-in function that extends the capabilities of the existing providers.
- Chapter 8, “Audit Events,” describes how to extend the AuditContext interface. This interface provides a mechanism for passing additional audit information to Auditing providers.
- Chapter 9, “BLM Configuration API Security Providers Reference,” describes the security provider attributes and default values. This information is needed if you want to use the BLM API to configure security providers.

Related Information

Other documents that may be of interest to the reader include:

- *Policy Managers Guide*—This document explains how to write access control policies and describes how to import and export policy data.
- *SSM Installation and Configuration Guide*—This document describes how to install Security Services Modules.
- *Developing Security Providers* —This document provides security vendors and security and application developers with the information needed to develop custom security providers.
- For API documentation in Javadoc format, use the links on the *documentation home page*.

CHAPTER 2

Administrative Utilities

Oracle Entitlements Server includes a number of helpful administrative utilities. This section provides a reference to the following utilities:

Note: "Configuration Files" on page 4-1 describes which configuration files are used by a particular utility.

Observe these conventions when reading these sections:

- `OES_ADMIN_HOME` is the directory where the Administration Server is installed.
- angle brackets <> indicate required variable arguments
- square brackets [] indicate optional variable arguments

install_ales_schema

Installs the policy database schema into the database server. If the schema already exists, it will be replaced, including existing policy. On UNIX, the program prompts you to input the arguments. Make sure the current working directory is `BEA_HOME\ales32-admin\bin` before running the tool.

Usage

```
BEA_HOME\ales32-admin\bin\install_ales_schema.bat <db-username>
<db-password>
BEA_HOME\ales32-admin/bin/install_ales_schema.sh
```

Options

db-username
Login ID, usually same as owner

db-password
Password for the db-username

Example

```
install_ales_schema.bat username password
```

asipassword

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

Usage

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

OES_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 (`BEA_HOME/ales32-shared/keys/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 (`BEA_HOME/ales32-shared/keys/password.key`) is used if you do not supply a different value for this option.

Example

In this example, the command is issued from the bin directory:

```
asipassword admin ../keys/password.xml ../keys/password.keycd keys
```

asisignal

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

Usage

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

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

Options

-action ping, comtest

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

-action status

Get the server status. Could be INITING or READY.

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

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

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

-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 ASIAuthorizer format to XACML. It reads policies from an input file in policyloader format, translates rules to XACML, and stores the XACML rules to an output file.

Usage

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

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

Options

-in

The input policy file name. If not provided, read standard input, until EOF is detected.

-out

The output policy file name. If not provided, print to standard output.

Example

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

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 Administration Server is running.

Note: The Apache and IIS SSMs use a different version of this script. See "Usage" on page 2-5.

During the enrollment process, you will be asked for the administrator's username and password to connect to the 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

For all SSMs *except* the IIS and Apache SSMs:

```
BEA_Home\ales32-shared\bin\enroll.bat <demo|secure>
BEA_Home/ales32-shared/bin/enroll.sh <demo|secure>
```

For the IIS and Apache SSMs:

```
BEA_Home\ales32-ssm\<ssm_type>\instance\<instance_name>\adm\enroll.bat
<demo|secure>
BEA_Home/ales32-ssm/<ssm_type>/instance/<instance_name>/adm/enroll.sh
<demo|secure>
```

where
<*ssm_type*> is `apache-ssm` or `iis-ssm`.
<*instance_name*> is the SSM instance name.

Options

demo

Enrolls the SSM instance and verifies Administration Server certificate using the demo CA certificate from the `DemoTrust.jks` key store. 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 `cacerts` file in the `BEA_HOME/`*`jdk-version`*`/jre/lib/security` directory. If this option is specified, the tool verifies matching of the Administration Server host with the one from the certificate.

Example

```
enroll demo
```

unenroll

Note: This tool has been deprecated and applies only to the Web Server SSM in this release.

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 Administration Server is running.

During the un-enrollment process, you will be asked for the administrator's username and password to connect to the 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 . 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/`*`jdk-version`*`/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

Import/Export Utilities

This section provides detailed information about tools used to import/export data from the database:

PolicyIX

The Policy Propagation Import/Export (policyIX) tool can be used to import and export policies and also export SSM configurations — it does not support importing SSM configurations. It is commonly used to move policy data from one database to another and to export an SSM configuration to an XML file for use when the SSM is deployed without an SCM.

Note: This section does not cover the use of policyIX for imports/exports between OES and Oracle Enterprise Repository. For that information, see Storing and Versioning Policy with Oracle Enterprise Repository in the *Integration Guide*.

SSL Requirements

To sign exported configuration files, PolicyIX uses the SSL infrastructure specified during the Administration Server installation, . Specifically, the `policyIX.bat` file invokes the tool with `-Dales.policyTool.signer=wles-admin`. The `ales.policyTool.signer` property is a

required Java property that specifies the alias of the signing key in the identity keystore, which must be equal to the Administration server machine name.

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

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

After you export the configuration data, you must manually copy the XML configuration file and signature file to the SSM configuration directory, *`BEA_HOME/ales32-ssm/<ssm-type>/instance-name/`*`config`.

If you do not use the default name (`wles.securityrealm.xml`) for the configuration file, set the `wles.realm.filename` property in the SSM instance's `/config/security.properties` file. See the *SSM Installation and Configuration Guide* for additional information about the `security.properties` file.

Requirements and Limitations

- PolicyIX does not import or export user passwords. After an import, user passwords must be set using the Entitlements Administration Application.
- In this release, policyIX can be used to import policies that have been exported from ALES 2.6 and ALES 3.0 — but no versions prior to ALES 2.6. When these policies are imported, they will be located under `RootOrg/DefaultOrg/DefaultApp`.
- Prior versions of the policyIX configuration file (`config.xml`) are not supported. For more information about this file, see "PolicyIX Configuration File" on page 3-4.
- On WebLogic Server 8.15, this release of policyIX does not support importing/exporting policies. As a workaround, the policy propagation servlet can be modified to support imports/exports. For instructions, see "Modify Policy Propagation Servlet" on page 3-13.

Export an SSM Configuration

Use the following command to export an SSM configuration:

```
policyIX.bat <config_id> -exportConfig <config.xml> <exportfile.xml>
-passwdPrompt
```

For example, the following command exports a SSM configuration named `java2ssm` to a file named `java2ssmconfig.xml`.

```
policyIX.bat java2ssm -exportConfig ..\config\policyIX_config.xml
java2ssmconfig.xml
```

Element	Description
<*config_id*>	The name of the SSM configuration to be exported.
-export	A static parameter required for exports.
<*config.xml*>	The configuration file used by policyIX. For details, see "PolicyIX Configuration File" on page 3-4.
<*exportfile.xml*>	(Optional) The name of the file to contain the exported SSM configuration.
-passwdPrompt	(Optional) When included in the command, you will be prompted for the password. If not specified, the password must be specified in <*config.xml*>.

Export Policies

Use the following command to export policies from the database to an XML file:

```
policyIX.bat -export <config.xml> <exportfile.xml> -passwdPrompt
```

Element	Description
-export	A static parameter required for exporting policies.
<*config.xml*>	Configuration file used by policyIX. The <`export_configuration`> element in this file specifies the policies and identities to export and the product version of the SSM. See details, see "export_configuration" on page 3-5.
<*exportfile.xml*>	The name of the file that will contain the exported policies.
-passwdPrompt	(Optional) When included in the command, you will be prompted for the password. If not present on the command line, the password must be specified in <*config.xml*>.

Import Policies

Use the following command to load policies from an XML file to the database:

```
policyIX.bat -import -disableTransaction <config.xml> <importfile.xml>
-passwdPrompt
```

Element	Description
-import	A static parameter required for importing policies.
-disableTransaction	(Optional) Prevents loading of policies in a single transaction. This is recommended when loading a large policy set. It improves loading performance, but loaded policy data will not be rolled back if the loading fails.
<*config.xml*>	The configuration file used by policyIX. The `<import_configuration>` element must specify how to handle policy duplicates. For details, see "import_configuration" on page 3-7.
<*importfile.xml*>	This file will contain the policies to be imported. This is commonly a file that was exported for a database using policyIX. If the file contains multi-byte characters, it must be must be UTF-8 encoded.
-passwdPrompt	(Optional) When included in the command, you will be prompted for the password. If not specified, the password must be specified in <*config.xml*>.

PolicyIX Configuration File

Table 3-1 describes the possible elements in the policyIX configuration file.

Table 3-1 Configuration File Elements

Element	Description
policy_propagation	Parent element. `<policy_propagation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://policypropagation.ales.com/xmlbean" targetNamespace="http://policypropagation.ales.com/xmlbean">` **Child Elements**: configuration
configuration	Container element. **Child Elements:** export_configuration import configuration server_configuration aler_configuration
export_configuration	This is a container element that is used when the policyIX command includes the `-export` switch. It's child elements specify the policies and identities to export and the product version of the SSM. **Child Elements:** clipping_scope <withIdentity> <target_ssm_version>

Table 3-1 Configuration File Elements

Element	Description
clipping_scope	Specifies the object containing the policies to export. You may specify one or more Organizations and/or Applications. To export all policies under all applications in MyOrg: *<clipping_scope value="RootOrg!MyOrg"/>* To export all policies under the MyApp application: *<clipping_scope value="RootOrg!MyOrg!MyApp"/>* To export all policies under the App1 and App2 applications: *<clipping_scope value="RootOrg!MyOrg!App1,RootOrg!MyOrg!App2"/>* The scope can also be specified on the command line using the `-clippingscope` switch. This overrides the value specified in the configuration file. Example: `policyIX.bat -export -withIdentity all -clippingScope RootOrg!MyOrg cfg.xml exp.xml`
withIdentity	(Optional) Specifies what identities will be exported: *<withIdentity value="direct"/>* If not specified or the clipping scope is an application, no identities are exported. **Values:** `direct` = export all identities defined in the parent Organizaton `all` = export identities defined in the parent Organizaton and those the Organization is inheriting. This can also be specified on the command line using the `-withIdentity` switch. Using this switch overrides the value specified in the configuration file. Example: `policyIX.bat -export -withIdentity all -clippingScope RootOrg!MyOrg cfg.xml exp.xml`

Table 3-1 Configuration File Elements

Element	Description
target_ssm_version	Specifies the release version of the SSM that will use the exported configuration. This option is required *<target_ssm_version value="3.x"/>* **Values:** `2.x` = ALES 2.x `3.x` = ALES 3.0 and OES
import_configuration	This is a container element that is used when the policyIX command includes the `-import` switch. Its child element specifies how to handle policy duplicates. **Child Elements:** policy_load_procedure
policy_load_procedure	Specifies what to do when the database contains a duplicate policy. *<policy_load_procedure value="override"/>* **Values:** `override` = add the policy without removing the existing policy `delete_existing` = delete the existing policy before importing the policy.
server_configuration	This container element is used for all policyIX commands. Its child elements specify how to connect to OES. **Child Elements:** server_property

Table 3-1 Configuration File Elements

Element	Description
server_property	The following name/value pairs that specify how to connect to OES. server_property name="server_host_name" value="smysore02"/> server_property name="server_port" value="7010"/> server_property name="blm_server_port" value="7011"/> server_property name="print_info" value="false"/> server_property name="userID" value="admin"/> server_property name="userPassword" value="password"/> **Values:** `server_host_name` = Machine name or IP address of admin server `server_port` = Admin server port (default = 7010) `blm_server_port` = BLM server port (default = 7011) `print_info` = If set to true then policy propagation progress will be sent to standard console output `userID` = the OES administrator user name (default = admin). `userPassword` = the OES admin password (default = *password*). Not required if the policyIX command includes the `-passwordPrompt` switch

Table 3-1 Configuration File Elements

Element	Description
aler_configuration	Container used when the policyIX command includes the `-exportToALER` or `-importFromALER` switch. For information about OES to OER imports and exports, see Storing and Versioning Policy with Oracle Enterprise Repository in the *Integration Guide*. **Child Elements:** aler_property
aler_property	Name/value pairs that specify how to connect to Oracle Enterprise Repository. <aler_property name = "server_version" value="3.0"/> <aler_property name = "server_url" <value="http://localhost:7101/aler/services/FlashlineRegistry"/> <aler_property name = "userName" value="admin"/> <aler_property name = "userPassword" value="admin"/> <aler_property name = "assetName" value="ALESPolicyAsset"/> <aler_property name = "assetDescription" value = "export ALES to ALER"/> <aler_property name = "importAssetVersion" value="1"/> **Parameters/Values:** `version` = A OER server version of `3.0` or `2.6`. `server_url` = The URL for accessing the OER server. `userName` = The user name to use to connect to OER. `userPassword` = The user password of the above user. `assetName` = The name of the asset to export/import. `assetDescription` = (Optional) Asset description for an export `importAssetVersion` = Asset version to import (required with, only valid if the `-importFromAler` switch)

XML Policy Data Files

When using policyIX to import policies, the policies to be imported must be specified in the required XML format. This section describes the required format and provides an example.

In addition to the information provided here, there are two ways understand the XML requirements:

- Perform a policy export using policyIX, the generated export file will contain the correct XML structure and format.
- A XSD schema can be downloaded from http://www.oracle.com/technology/products/id_mgmt/oes/ales_policy_propagation_schema.zip.

Importing a Role Policy

The XML shown in Figure 3-1 could be used to import the following policies:

```
grant(//role/myrole, //resources/myresource, //MyOrg/MyDir/RJones/) if
true;
grant( post, //myresource, //role/myrole/) if true;
```

Figure 3-1 Sample XML for Policy Imports

```
<?xml version="1.0" encoding="UTF-8"?>
<xb:policy_propagation
xmlns:xb="http://policypropagation.ales.com/xmlbean">
<xb:policy_propagation_data_v2>
  <xb:scopes>
    <xb:application_entry value="RootOrg!MyOrg!MyApp">
       <xb:resources>
          <xb:resource_entry value="//resources/myresource"/>
       </xb:resources>
       <xb:actions>
          <xb:action_entry value="post"/>
       </xb:actions>
       <xb:roles>
          <xb:role_entry value="myrole" parent=""/>
       </xb:roles>
       <xb:policies>
         <xb:membership_rule_entry>
            xb:policy_effect value="grant"/>
            xb:policy_roles>
                <xb:policy_role_entry value="myrole"/>
            </xb:policy_roles>
            <xb:policy_resources>
              <xb:policy_resource_entry value="//resources/myresource"/>
            </xb:policy_resources>
            <xb:policy_subjects>
             <xb:policy_user_entry name="RJones" directory="MyDir"
scope="MyOrg"/>
            </xb:policy_subjects>
```

```
            </xb:membership_rule_entry>
            <xb:authorization_policy_entry>
              <xb:policy_effect value="grant"/>
              <xb:policy_actions>
                  <xb:policy_action_entry value="post"/>
              </xb:policy_actions>
              <xb:policy_resources>
                 <xb:policy_resource_entry value="//resources/myresource"/>
               </xb:policy_resources>
               <xb:policy_subjects>
                 <xb:policy_role_entry value="myrole"/>
               </xb:policy_subjects>
            </xb:authorization_policy_entry>
                  </xb:policies>
      </xb:application_entry>
      </xb:scopes>
  </xb:policy_propagation_data_v2>
  </xb:policy_propagation>
```

Table 3-2 describes the XML elements used in the above example.

Table 3-2 XML Import Elements

Element	Description
policy_propagation	Parent container <xb:policy_propagation xmlns:xb="http://policypropagation.ales.com/xmlbean">
policy_propagation_data_v2 or policy_propagation_data	Parent container for all ALES policies. When importing OES policies use `<policy_propagation_data_v2>`. When importing ALES 3.0 policies use `<policy_propagation_data>`.
scopes	Parent container for one or more organization_entry and application_entry elements.
application_entry	Parent container for the application where the policies will be imported. `<xb:application_entry value="RootOrg!MyOrg!MyApp">`

resources	Specifies the resources being secured by the policy.
	`<xb:resources>`
	`  <xb:resource_entry value="//resources/myresource"/>`
	`</xb:resources>`
	This definition must include the initial `//resources/` string followed by the resource hierarchy. Not required if the resources already exist.

Table 3-2 XML Import Elements

Element	Description

actions
action_entry
	Specifies the policy actions.
	`<xb:actions>`
	`  <xb:action_entry value="post"/>`
	`</xb:actions>`
	This is not required if the action already exists in the application.

roles
role_entry
	Specifies roles used in the policy definition.
	`<xb:roles>`
	`  <<xb:role_entry value="myrole" parent=""/>`
	`</xb:roles>`
	Not required if the roles already exist. The `parent=` parameter must specify the role hiearchy.

policies	This container element contains the policies to import and can include one or more `<xb:membership_rule_entry>` and `<xb:authorization_policy_entry>` elements.
	`<xb:policies>`
	`  <xb:membership_rule_entry>`
	`  .`
	`  </xb:membership_rule_entry>`
	`  <xb:authorization_policy_entry>`
	`  .`
	`  </xb:authorization_policy_entry>`
	`</xb:policies>`

Table 3-2 XML Import Elements

Element	Description

membership_rule_entry	Specifies a role policy to import.
	`<xb:membership_rule_entry>`
	`<xb:policy_effect value="grant"/>`
	`<xb:policy_roles>`
	`<xb:policy_role_entry value="myrole"/>`
	`</xb:policy_roles>`
	`<xb:policy_resources>`
	`<xb:policy_resource_entry value="//resources/myresource"/>`
	`</xb:policy_resources>`
	`<xb:policy_subjects>`
	`<xb:policy_user_entry name="RJones" directory="MyDir" scope="MyOrg"/>`
	`</xb:policy_subjects>`
	`</xb:membership_rule_entry>`

authorization_policy_entry	Specifies an authorization policy to import.
	`<xb:authorization_policy_entry>`
	`<xb:policy_effect value="grant"/>`
	`<xb:policy_actions>`
	`<xb:policy_action_entry value="post"/>`
	`</xb:policy_actions>`
	`<xb:policy_resources>`
	`<xb:policy_resource_entry value="//resources/myresource"/>`
	`</xb:policy_resources>`
	`<xb:policy_subjects>`
	`<xb:policy_role_entry value="myrole"/>`
	`</xb:policy_subjects>`
	`</xb:authorization_policy_entry>`

Modify Policy Propagation Servlet

By default, policyIX on WebLogic Server 8.15 supports the export of configuration data — but not policy data. This section describes how to manually modify the policy propagation servlet so that policyIX will also import/export policy data.

1. Unpack the following WAR file:

   `BEA_Home\ales32-admin\asiDomain\applications\asi.war`

2. Modify `web.xml` to include the following servlet and listener definitions:

```
<listener>
    <listener-class>com.ales.policypropagation.servlet.PropagationSessi
    onListener</listener-class>
</listener>


<servlet>
    <servlet-name>policyPropagationServlet</servlet-name>
    <servlet-class>com.ales.policypropagation.servlet.PolicyPropagation
    Servlet</servlet-class>
        <init-param>
            <param-name>file.max.upload.size</param-name>
            <param-value>104857600</param-value>
        </init-param>
        <init-param>
            <param-name>session.inactive.internal</param-name>
            <param-value>2</param-value>
        </init-param>
        <init-param>
            <param-name>output.bulksize</param-name>
            <param-value>200</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>policyPropagationServlet</servlet-name>
    <url-pattern>/policypropagation</url-pattern>
</servlet-mapping>
```

3. Rebuild `asi.war` with the updated `web.xml` file and update the stage file located in `BEA_HOME/ales32-admin/asiDomain/asiAdminServer/stage/asiconsole/asi.war` directory.

4. Modify `BEA_HOME/ales32-admin/config/WLESWebLogic.conf` as shown below.

   Note: Entries you must add are shown in ***bold italics***.

   wrapper.java.classpath.1=@bea.home@
   wrapper.java.classpath.2=@admin.home@/lib/asi-weblogic.jar
   wrapper.java.classpath.3=@admin.home@/lib/jsafe.jar
   wrapper.java.classpath.4=@admin.home@/lib/jsafeJCE.jar
   wrapper.java.classpath.5=@admin.home@/lib/asn1.jar
   wrapper.java.classpath.6=@admin.home@/lib/certj.jar
   wrapper.java.classpath.7=@admin.home@/lib/saaj.jar
   wrapper.java.classpath.8=@java.home@/lib/tools.jar

wrapper.java.classpath.9=@weblogic.home@/server/lib/weblogic_sp.jar
wrapper.java.classpath.10=@admin.home@/lib/xmlpublic.jar
wrapper.java.classpath.11=@weblogic.home@/server/lib/weblogic.jar
wrapper.java.classpath.12=@weblogic.home@/server/lib/webservices.jar
wrapper.java.classpath.13=@admin.home@/lib/wls-v8-rt.jar
wrapper.java.classpath.14=@admin.home@/lib/api.jar
wrapper.java.classpath.15=@admin.home@/lib/framework.jar
wrapper.java.classpath.16=@admin.home@/lib/scmapi.jar
wrapper.java.classpath.17=@admin.home@/lib/log4j.jar
wrapper.java.classpath.18=@admin.home@/lib/asi_classes.jar
wrapper.java.classpath.19=@admin.home@/lib/ssladapter.jar
wrapper.java.classpath.20=@admin.home@/lib/asitools.jar
wrapper.java.classpath.21=@admin.home@/lib/process.jar
wrapper.java.classpath.22=@admin.home@/lib/webservice.jar
wrapper.java.classpath.23=@admin.home@/lib/providers/ales/jconn3.jar
wrapper.java.classpath.24=@weblogic.home@/server/lib/jconn2.jar
wrapper.java.classpath.25=@admin.home@/lib/wrapper.jar
wrapper.java.classpath.26=@admin.home@/lib/asisignal.jar
wrapper.java.classpath.27=@admin.home@/lib/CR330583_414.jar
wrapper.java.classpath.28=@admin.home@/lib/CR336221_414.jar
wrapper.java.classpath.29=@admin.home@/lib/CR338979_414_jdk1.4.jar
wrapper.java.classpath.30=@admin.home@/lib/kodo-runtime.jar
wrapper.java.classpath.31=@admin.home@/lib/jca1.0.jar
wrapper.java.classpath.32=@admin.home@/lib/jdbc2_0-stdext.jar
wrapper.java.classpath.33=@admin.home@/lib/jdo.jar
wrapper.java.classpath.34=@admin.home@/lib/jta-spec1_0_1.jar
wrapper.java.classpath.35=@admin.home@/lib/openjpa.jar
wrapper.java.classpath.36=@admin.home@/lib/commons-collections-3.2.jar
wrapper.java.classpath.37=@admin.home@/lib/commons-lang-2.1.jar
wrapper.java.classpath.38=@admin.home@/lib/commons-pool-1.3.jar
wrapper.java.classpath.39=@admin.home@/lib/serp.jar
wrapper.java.classpath.40=@admin.home@/lib/persistence-api.jar
wrapper.java.classpath.41=@admin.home@/lib/quarkparser.jar
wrapper.java.classpath.42=@admin.home@/lib/backport-util-concurrent.jar
wrapper.java.classpath.43=@admin.home@/lib/org.mortbay.jetty.jar
wrapper.java.classpath.44=@admin.home@/lib/javax.servlet.jar
wrapper.java.classpath.45=@admin.home@/lib/org.apache.jasper.jar
wrapper.java.classpath.46=@admin.home@/lib/sslserver.jar
wrapper.java.classpath.47=@admin.home@/lib/sslclient.jar
wrapper.java.classpath.48=@admin.home@/lib/axis.jar
wrapper.java.classpath.49=@admin.home@/lib/commons-logging-1.0.4.jar

wrapper.java.classpath.50=@admin.home@/lib/commons-discovery-0.2.jar
wrapper.java.classpath.51=@admin.home@/lib/pdsoap.jar
wrapper.java.classpath.52=@admin.home@/lib/wsdl4j-1.5.1.jar
wrapper.java.classpath.53=@admin.home@/lib/antlr.jar
wrapper.java.classpath.54=@admin.home@/lib/xmlbeans-2.3.0/xbean.jar
wrapper.java.classpath.55=@admin.home@/lib/jsr173_1.0_api.jar
wrapper.java.classpath.56=@weblogic.home@/server/lib/ojdbc14.jar
wrapper.java.classpath.57=@admin.home@/lib/ld-server-core.jar
wrapper.java.classpath.58=@admin.home@/lib/wlsdo.jar
wrapper.java.classpath.59=@admin.home@/lib/wlxbean.jar
wrapper.java.classpath.60=@admin.home@/lib/xbean.jar
wrapper.java.classpath.61=@admin.home@/lib/xqrl.jar
wrapper.java.classpath.62=@admin.home@/lib/ld-client.jar
wrapper.java.classpath.63=@db.jdbc.driver.loc@
wrapper.java.classpath.64=@admin.home@/config
wrapper.java.classpath.65=@admin.home@/lib/alesAssetSchemaObjects.jar
wrapper.java.classpath.66=@admin.home@/lib/policyPropagation.jar
wrapper.java.classpath.67=@admin.home@/lib/policyPropagationALER30.jar
wrapper.java.classpath.68=@admin.home@/lib/configSchemaObjects.jar
wrapper.java.classpath.69=@admin.home@/lib/policyPropagationALER26.jar
wrapper.java.classpath.70=@admin.home@/lib/policySchemaObjects.jar
wrapper.java.classpath.71=@admin.home@/lib/commons-fileupload.jar
wrapper.java.classpath.72=@admin.home@/lib/managementapi.jar
wrapper.java.classpath.73=@admin.home@/lib/jdom.jar

5. Log in to the Administration Console and create a resource named `policypropagation` under the following existing resource:

   `ASI/asiconsole/url/asi`

6. Add the following policy to grant POST, GET actions on the `policypropagation` resource:

   ```
   grant( [//priv/GET,//priv/POST],
   //app/policy/ASI/asiconsole/url/asi/policypropagation, //role/Everyone)
   if true;
   ```

7. Open `policyIX.sh|bat` in an editor and add the following java option:

   `-Dadmin.server.type=wls81`

Policyexporter

Policyexporter is used to output data from the policy store to plain-text policy files. These policy files can be imported back to the same or to another policy store using the policyloader as described in "Policyloader" on page 3-18.

The plain-text policy files include the following management object types:

— Resources, resource attributes, and resource bindings
— Actions, action bindings, and action groups
— Roles
— Authorization and role policies
— Identities, including user, group, and membership
— Declarations, constants and evaluation functions
— SSM configuration data and providers' configuration data

The export process also generates two additional files (object_config, objattr_config) that contain SSM configuration data. These files also get loaded and are similar to object and objattr respectively in format. These files are split so as to differentiate policy elements from configuration elements. However, object_config and objattr_config can be merged into object and objattr respectively, if needed.

Requirements and Limitations

Observe the following requirements and limitations:

- A directory must be set up to hold the exported policies. Any existing policy files in that directory will replaced or deleted.
- To perform the export, you need access to the policy database. In general, you must be the policy owner or a database administrator.
- Policyexporter exports all policies in the database.
- Policylexporter does not export user passwords.
- Policyexporter does not support Organizations, Application, Role attributes, and Meta objects.

Performing a Policy Export

To perform a policy export using policyexporter:

1. Set up a target directory in which to store the policy files.

   The directory name must contain no white space and cannot be not write-protected. The required free space depends on the size of the policy being exported.

2. Make sure the database client is configured to connect to the database and that you have the access rights to perform an export.

   To perform an export, you must be a database administrator or the policy owner.

3. Ensure that you can run the tools from the Administration Server's `/bin` directory. These scripts need to locate files relative to this directory.

4. Execute `policyexporter.bat|.sh` as described in "Running Policyexporter" on page 3-18.

Running Policyexporter

To run policyexporter, perform the following steps:

1. Open a command window and change to the Administration Server's `\bin` directory.

2. Ensure that the current path (.) is included your `PATH` and that the client environment is set up properly.

3. Run `policyexporter.bat|.sh <directory>`

   where `<directory>` is the full path to the target directory where the policy should be exported.

Notes:

- The export saves configuration information to the following files: `object_config` and `objattr_config`.

Policyloader

The Policy Import tool (`policyloader.bat|sh`) is a Java utility that can be used to import and export policy data. To import policies, it reads text files containing the policy definitions — each policy element is stored in a separate file, referred to as a policy file. For information on the specific format of these policy elements, see Advanced Topics in the *Policy Managers Guide*.

PolicyIX imports/exports support the following objects:

— Organizations and Identities
— Applications
— Resources and resource attributes
— Actions including action groups
— Declarations, including attribute definitions, evaluation functions, and constants.
— Roles, role hierarchies, role attributes
— Policies including Membership Rules and Authorization policies
— Resource binding
— SSM configuration (export only)
— Meta objects

Policyloader uses multi-threading processing and is optimized for fast imports and distribution of large policies. In addition, policy imports are transactional: all policies are loaded, or none.

Note: When running the Policy Import tool on a large policy, the number of records processed may not be synchronized. If multiple threads are used to import the data, when one thread completes before the other cannot be determined. If the threads are set too high, a message may appear indicating that the number of records processed is not synchronized. This is normal and is not a problem for the Policy Import tool.

Requirements and Limitations

- If you import a file containing multi-byte characters, the file must be UTF-8 encoded.
- Policyloader does not import user passwords. After an import, user passwords must be set using the Entitlements Administration Application.
- Policyloader does not support Organizations, Applications, Role attributes, and Meta objects. All policies are imported under the default application. All identities are imported under the default organization.

Performing a Policy Import

To perform a import using policyloader:

1. It is assumed that policy data files will be created by exporting policies from a database using the policyexporter utility as described in "Policyexporter" on page 3-17.

   In addition, policy data files may be manually created as described *Advanced Topics* in the *Policy Managers Guide*.

2. Set up a configuration file to define your policy load. For instructions, see "Policyloader Configuration File" on page 3-20.

   You can use the `../examples/common/data/policy/template/load.conf` file as a template.

3. Run the tool as described in "Running Policyloader" on page 3-25.

4. Check for errors in log file as described in "Check for Errors" on page 3-26.

5. If necessary, rerun policyloader in `-recover` mode as described in "Recover Mode" on page 3-26.

Policyloader Configuration File

The configuration file consists of name/value pairs specifying information needed by policyloader. The file paths in the configuration file depend on the directory from which you run the Policy Import tool. You may use the full path filename to avoid directory dependency. Spaces are allowed between parameters and between new lines. Parameter names are case insensitive. Table 3-3 lists the parameters you need to configure for the Policy Import tool.

A sample file configuration file is shown in "Policyloader Configuration File" on page 3-20.

Table 3-3 Configuration Parameters

Parameter	Description
`Action`	The action to be performed. Supported values are `LOAD` and `REMOVE` (case insensitive). `REMOVE` = Removes the policy from the database. `ADD` = Loads the policy data into the database.
`ApplicationNode`	The application in the database that holds the administration policy (default = admin). If commented out, the default value of `admin` is used.
`BLMContext Retries`	Specifies the number of times retries should take place. This can be used in the event the Administration server is still starting up when the import is being executed. In most cases the Administration server is always running. Default: 100.
`BLMContext Interval_ms`	Specifies the amount of time (in milliseconds) to wait between context retries. DEFAULT: 10 (100 milliseconds)

Table 3-3 Configuration Parameters (Continued)

Parameter	Description
`BulkSize`	The number of records to send at one time in a thread. Default: 200. The optimal value is between 50 and 300. Setting this value too high may slow the import process slows and also higher `RequestTimeout` and `ConnectionTimeout` values. When there are multiple threads importing policy data, each processing a number of records, the number of records processed may result in an "out-of-sync" message. However, it does not harm the data when importing the policy. The policy import tool switches to single thread when importing some policy elements, such as resources and declarations, as the later records have dependency on earlier records.
`ConsoleDisplay`	Specifies whether to hide console interaction or not (yes/no). If you want to run the policy loader in the background as a batch process, set to `no`. Default: `yes` `no =` Error messages are not displayed on the console and the user is requested to enter their `Username` and `Password` if they are missing in the configuration file. `yes` = Error messages are displayed on the console. This parameter must be enabled if you want to type in your password on the command prompt, rather then use the one specified in the `password.xml` or in the configuration file.
`Debug`	Specifies whether you want to log debug information. Default: 0 `0` = Does not log debug information. `1` = Sends debug information to the file defined by: `ErrorLogFile.`
`Domain`	Always should be set to `asi`.
`ErrorLogFile`	Specifies the name of error log file that will be generated if errors occur. Default: `error.log`.

Table 3-3 Configuration Parameters (Continued)

Parameter	Description
`Mode`	The mode of operation. Values are `INITIAL` or `RECOVER` (case insensitive). Use INITIAL mode the first time you run the Import Policy Tool to load a set of policy files. If you encounter errors in the initial load attempt, check the `ErrorLogFile` for a description of the error, correct the errors in the generated error file(s) (an error file is produced for each policy file that fails), and rerun the Import Policy Tool again, but this time in the RECOVER mode. This way the tool only attempts to load the generated error files. If the tool fails again, fix the errors, and run it again in RECOVER mode. Repeat until no errors are encountered. **Note:** This parameter can also be passed in as a command-line parameter `-recover` or `-initial`. Values for this parameter on the command line override values specified in the configuration file.
`PasswordFile`	The encrypted password file. This file can be generated using the `asipassword` utility. For details, see xxxxx. If this and the `PasswordKeyFile` parameters are not specified and console display is enabled, you will be prompted for these values.
`PasswordKey File`	The private key used to decrypt the password stored in the above file.
`Policy DirectoryPath`	The directory from which to import policy files. For example: `../examples/policy`. The path may be relative. Default: "`.`" (for relative)
`Policy Distribution`	Indicates if the policies should be flagged for distribution when imported into the database. `YES` = flag the policies for distribution `NO` = do not flag for distribition. Policies flagged for distribution can be distributed using the administration console or the Entitlements Administration Application.
`RequestTimeout`	Specifies the time (in milliseconds) to wait for the server to respond. Should be longer for loading large files. May set to infinite (`ASI.INFINITE`) for very large files. Default: 600000

Table 3-3 Configuration Parameters (Continued)

Parameter	Description
`RunningThread`	Number of threads running concurrently to process the policy import. The value depends on the capacity of the database server. Commonly, the optimal value is 2 - 4 or larger for a high capacity database server. Default: 3. If set too high, performance may be slowed. The error logs will indicate this by `database busy` messages.
`Username`	(Optional) The OES admin username (default = `admin`). If not specified and `ConsoleDisplay` parameter is *enabled*, you will be prompted for this value. **Note:** This user must have the privilege to import policy.

Sample Configuration File

Listing 3-1 provides sample configuration file.

Note: Be sure to use forward slashes (/) when specifying the policy file directory path.

Listing 3-1 Sample Configuration File

```
# Copyright (c) 2004-2008 Oracle and/or its affiliates. All rights reserved.
##########################################################################

## In addition, asi.properties is read in from the /config directory.
Parameters set here will override values defined there.
#### Enterprise Domain Name (DEPRICATED - Should always be asi)
Domain asi

#### A OES admin user name. Must be the same as stored in database.
Username admin

#### Encrypted password file
PasswordFile C:/bea/ales32-shared/keys/password.xml

#### Password key file
PasswordKeyFile C:/bea/ales32-shared/keys/password.key
```

```
#### The application node holding the administration policy.
#### If commented out it assumes the dafult value of "admin".
ApplicationNode admin

#### Milliseconds to wait for server to respond. For large files
#### increase or use ASI.INFINITE
RequestTimeout 600000

#### Number of times retries should take place(DEFAULT 100)
BLMContextRetries 10

#### Milliseconds to wait between retries (DEFAULT 100ms)
BLMContextInterval_ms 10000

#### Number of concurrent threads.
RunningThread 2

#### Number of entries loaded in a single load (200 here)
BulkSize 200

#### Directory of policy files. May be a relative path.
PolicyDirectoryPath C:/bea/ales32-admin/data/adminPolicy/standardBase

#### Indicate if policies should be distributed. Default = YES
PolicyDistribution flush

#### File where all error messages are logged.
ErrorLogFile C:/bea/ales32-admin/log/loader.log

#### Action to be performed: load or remove.  Default = LOAD
#Action REMOVE

#### Policy import mode: INITIAL or RECOVER. Default = INITIAL
#### This parameter can also be passed in as a command parameters.
#### Command line parameters override values in this file.
#Mode RECOVER

#### Uncomment to see debug information, default = 0 (no debug)
#Debug 1

#### Uncomment to hide console interaction (yes/no), default = yes
#### If you want to run loader in background/in batch process, set this to no
ConsoleDisplay no
```

Running Policyloader

Once the configuration file and the policy data files are set up, run policyloader follows:

1. Execute: `policyloader.bat|sh <load.conf>`

   where `<load.conf>` is the name of the configuration file.

   You may also use the following options:

   `-disableTransaction`
   Prevents loading of policies in a single transaction. This is recommended when loading a large policy set. It improves loading performance, but loaded policy data will not be rolled back if the loading fails.

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

   `-initial`
   Run in initial mode (there should be no versioned files in the policy directory).

   `-recover`
   Run in recover mode to revert to an earlier policy set. For details, see "Recover Mode" on page 3-26.

   `-load`
   (Default) Load policy from the files specified in the configuration file.

   `-remove`
   Removes the policies from the database.

2. Check for errors in log file.

Notes:

- If an error occurs and the policyloader terminates; you must restart the import.
- Policy files are processed in a predefined order. If a file is not found, the next file is processed. Records successfully imported are committed to the database.
- To nullify any changes made to the database, run policyloader again to make the necessary changes.
- A `Object Exists Error` message indicates that a duplicate policy entry was created.
- If an error other than `Object Exists Error` occurs, a file is generated about the error and the error message is logged. See the next section for more information.

Check for Errors

If an error occurs, the Policy Loader terminates; you must restart the Policy Import tool. The name of the error file is defined in the your Policy Import tool configuration file by the `ErrorLogFile` parameter. In addition, to distribute policy you need distribution privileges granted to you.

Also, because the Policy Import tool is multi-threaded and each thread writes out to the log when it is complete, you cannot guarantee the order in which each load completes.

The Policy Import tool processes policy files according to a predefined order, and if the policy file is not found, it tries to load the next policy file in the proper order. Records imported successfully are committed to the database. After the import process begins, you cannot go back within the same process and edit changes you have made. If you want to change what you have done, you have to start a new import process. After the import process is complete, you may run the removal operation to reverse the import process.

Recover Mode

When the policyloader encounters an error (other than `Object Exists Error`), it generates a file named `<filename>.<version>` (for example, `object.1` , `object.2`) and logs an error message. These files contain only the problematic lines from policy data files. You can now correct the mistakes in these files and re-run the policy loader in the recover mode.

Recover mode can be run in one of two ways:

- By setting the configuraiton file's Mode parameter to `RECOVER`.
- Adding `-recover` to the policyloader command line.

In recovery mode, policyloader will only try to load the highest version files that have not already been loaded. If you corrected `priv.1` and there are still problems, then the loader will generate `priv.2` with just the lines that failed. You can then correct `priv.2` and rerun the policy loader in the recover mode. Perform additional iterations until policyloader does not generate any new version files and the error log file does not indicate any outstanding errors.

Load_Adminpolicy

Loads the administrative policy that determines administrative access to access the Administration Console and Entitlements Administration Application. This utility is automatically called during the installation process and it is unlikely you will ever need to run it independently.

Usage

```
OES_ADMIN_HOME\bin\load_adminpolicy.bat
```

Example

```
load_adminpolicy.bat
```

CHAPTER 4

Configuration Files

This section provides information about confiuration files provided when the Administration Server and SSMs are installed. These files are provided in the `config` directory of the Administration Server, SCM, and SSM instance.

Administration Server Configuration Files

Table 4-1 describes the installed Administration server configuration files.

Table 4-1 Admin Configuration Files

Tool	Config File	Explanation
Admin Examples	admin_install.properties	This file is similar to a Java properties file and can be read to determine the parameters specified during the install of the Admin Server.
Admin installer when run in silent mode	silent_install_admin.xml	This file captures the parameters specified during installation of the Admin Server and can be used for silent installs of similar configurations.
annotation_transform.bat|sh	annotation_config.properties annotation_transform.xml	The annotation_transform tool invokes annotation_transform.xml ant script and gets its configuration from annotation_config.properties. This tool is only needed for annotated policy files created via Eclipse.
policyloader.bat|sh	asi.properties	Policyloader uses the file to initialize the BLM API for commuunication with the BLM server.
policyIX.bat|sh	policyIX_config.xml	policyIX_config.xml must be updated before used as input to the policyIX tool.
BLM WebApp	WLESblm.properties	Used to configure the BLM WebApp. The BLM looks for this file based on the setting for the Java option ales.blm.home and also in /config/WLESblm.propeties .
WebService interface to BLM	blm.wsdl pd.wsdl	These files used to compile a Web service client that will be able to talk to the BLM server via SOAP.

Table 4-1 Admin Configuration Files

Tool	Config File	Explanation
install_ales_schema.bat|sh uninstall_ales_schema.bat|sh upgrade_ales_schema.bat|sh databaseloader.bat|sh	database.properties	Contains the JDBC URL specified during install and is used by persistence layer to connect to the database.
set-env.bat|sh set-wls-env.bat|sh (WLS 8.1) WLESWeblogic.conf (WLS 8.1) WLESTomcat.conf	log4j.properties	Referenced in the set-env files used when configuring an WebLogic 8.1 SSM. This file controls the log4j logging for the SSM.
set-wls-env.bat|sh (WLS 9.2\10.0) WLESWeblogic.conf (WLS 9.2\10.0)	log4j.wls9.properties	Similar to log4j.properties, but specific to a WLS SSM.
set-env.bat|sh set-wls-env.bat|sh WLESWeblogic.conf WLESTomcat.conf	WLESarme.properties	Used to configure the ASI Authorization provider.
WLESWebLogic.bat|sh	WLESWebLogic.conf	Used by the Wrapper tool to start WebLogic server.
WLESTomcat.bat|sh	WLESTomcat.conf	Used by the Wrapper tool to start the Tomcat server.
propagateInitialCache.bat|sh	asiadmin.xml	Used by the propagateInitialCache tool. The tool runs only for bootstrap purposes upon install to properly initialize the SCM and Admin SSM. It is automatically run as part of database schema install or via “WLESadmin.bat|sh init”.
load_adminpolicy.bat|sh	load.standardbase.conf	Load.standardbase.conf file is used by the load_adminpolicy tool to load the initial Admin policy after a fresh install.

Table 4-1 Admin Configuration Files

Tool	Config File	Explanation
	security.properties	Contains configuration properties for an SSM. By default, the runtime looks for a property file called 'security.properties' in the working directory. Only applicable to SSM running on Tomcat and WLS 8.1.
	SSM.properties	Can be used to determine the location of SCM and Admin install directories.

SCM Configuration Files

Table 4-2 describes SCM configuration files.

Table 4-2 SCM Configuration Files

Tool	Config File	Explanation
	scm_install.properties	Can be read to determine parameters used when the server was installed.
WLESscm.bat|sh	WLESscm.conf	Used by the Wrapper tool to start the SCM server.
asisignal.bat|sh WLESscm.conf	log4j.properties	Referenced from the startup scripts of the tools.
WLESscm.conf	kernel.xml	Config file for the Phoenix Java container framework that is used for creating the SCM process.
WLESscm.conf	java.policy	Configures the security policy for the SCM Java process.

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

SSM Common Configuration Files

Table 4-3 describes which configuration files are used by the SSM instance. Most files are common between various types of SSM instances; those that are specific to an SSM are described in the explanation column. Most files are located in the `config` directory but when this is not the case the directory is listed.

Table 4-3 Common SSM Config Files

Tool	Config File	Explanation
	SSM_HOME/adm/ssm_install.properties	Can be read to determine parameters selected during SSM install. This file is located in SSM_HOME/adm.
	SSM_HOME/adm/silent_install.xml	Can be used for silent installs with similar configurations. This file located in SSM_HOME/adm.
SSM Examples	adm/ssm_instance.properties	Can be read to determine parameters selected during creation of the SSM instance. This file is located in the INSTANCE_HOME/adm.
SSM instance wizard when run in silent mode	adm/silent_instance.xml	Can later be used for silent installs with similar configurations. This file is located in the INSTANCE_HOME/adm.
annotation_transform.bat|sh	annotation_config.properties annotation_transform.xml	The annotation_transform tool invokes annotation_transform.xml ant script and gets configuration from annotation_config.properties. This tool is needed only for annotated policy files created via Eclipse. These files are located in the SSM instance's `config` directory.
policyloader.bat|sh	asi.properties	Policyloader uses this file to initialize the BLM API for communicating with the BLM server.
policyIX.bat|sh	policyIX_config.xml	Must be updated before being used as input to policyIX tool.

Table 4-3 Common SSM Config Files

Tool	Config File	Explanation
set-env.bat|sh enroll.bat|sh unenroll.bat|sh WLESarme.properties	log4j.properties	Referenced from the startup scripts of the tools.
set-env.bat|sh	WLESarme.properties	Used to configure the ASI Authorization provider.
	shortcut.xml	Internal file used on Windows to control the shortcut menu items.
	SSM.properties	Can be used to determine location of SCM and Admin install directories.

Web Serivce SSM Configuration Files

The files shown in Table 4-4 are specific to the Web Service SSM.

Table 4-4 Web Service SSM Configuration Files

Tool	Config File	Explanation
	access_control-xacml-2.0-context-schema-os.xsd access_control-xacml-2.0-policy-schema-os.xsd xacml.wsdl	XML schema and WSDL files required when creating a WS SSM XACML client to connect to the WS SSM XACML WebService endpoint.
	ssm-soap-types.xsd SSM-SOAPWS.wsdl	XML schema and WSDL files required when creating a WS SSM client to connect to the WS SSM server.
WLESws.bat|sh	WLESws.wrapper.conf	Used by Wrapper tool that starts the Web service server.
	security.properties	Properties file for Web service server.

Table 4-4 Web Service SSM Configuration Files

Tool	Config File	Explanation
WLESws.wrapper.conf	kernel.xml	Config file for Phoenix Java container framework used for creating Web service Java process.
WLESws.wrapper.conf	java.policy	Configures the security policy for the Web service Java process.

WLS SSM Configuration Files

The files shown in Table 4-5 are specific to the WLS SSM.

Table 4-5 WLS SSM Configuration Files

Tool	Config File	Explanation
	DefaultAuthorizerInit.ldift	Used by the WLS DefaultAuthorizer Provider. Must be copied to the WLS domain if you plan to configure the WLS DefaultAuthorizer and ASI Authorizer providers together for the same SSM configuration.
	XACMLAuthorizerInit.ldift (only WLS9.x\10.0)	Used by the WLS XACMLAuthorizer Provider. Must be copied to the WLS domain if you plan to configure the WLS XACMLAuthorizer and ASI Authorizer providers together for the same SSM configuration.
WLESWebLogic.bat|sh	WLESWebLogic.conf	Used by the Wrapper tool that starts the WebLogic server.
	security.properties	Properties file for the WLS SSM server. Only applicable to SSM running WLS 8.1.

CHAPTER 5

WLESblm.conf Reference

Configuration parameters for the Business Logic Manager (BLM) are stored in the `WLESblm.conf` file, located in the `BEA_HOME/ales32-admin/config` directory. In most cases, this configuration can be accomplished using the installation program and the Administration Console. However, you may want to change default configurations by editing the `WLESblm.conf` file. This section provides a reference to the `WLESblm.conf` parameters.

Required Parameters

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

Table 5-1 Required Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin.domain	The enterprise domain on which BLM is running.	asi
BLM.wlesadmin.location	Location (Must be DEFAULT)	DEFAULT
BLM.wlesadmin.ASIPolicyARMEAddresses	The address of the ARME as a URL. The BLM directs authorization requests to this URL.	https://hostname:7012
BLM.wlesadmin.trustedPeerKeyStore	The file that contains the trusted Peer certificates in PEM format.	BEA_HOME/ales32-admin/ales32-shared/keys/peer.pem
BLM.wlesadmin.trustedCAKeyStore	The file that contains the trusted CA certificates in PEM format.	BEA_HOME/ales32-admin/ales32-shared/keys/trust.pem
BLM.wlesadmin.identityKeyStore	The file that contains the server's own certificate in PEM format.	BEA_HOME/ales32-admin/ales32-shared/keys/wles-admin.pem
BLM.wlesadmin.identityAlias	The alias that will be used to retrieve the identity private key	wles-admin
BLM.wlesadmin.passwordfile	Location of the password file that contains encrypted passwords indexed with an alias. The alias and the private key file are required to retrieve the password.	BEA_HOME/ales32-admin/ales32-shared/keys/password.xml
BLM.wlesadmin.passwordkeyfile	The password key is the master key that is required to retrieve any passwords from the password file.	BEA_HOME/ales32-admin/ales32-shared/keys/password.key
BLM.wlesadmin.configkeyfile	The config key is the master key that is required to decrypt any attributes set as sensitive in the database.	BEA_HOME/ales32-admin/ales32-shared/keys/config.key

Miscellaneous Configuration Parameters

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

Table 5-2 Miscellaneous Configuration Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin.port	The BLM's listening port. The BLM runs on HTTP/SOAP. The default value is the default SOAP port, 80.	80
BLM.wlesadmin. adminPolicyRoot	The admin policy root is created when you install the Administration Server. If, after installation, you make any change to the tree structure, you need to update this parameter as well. You do not need to change this parameter unless you are making changes to the security policies that protect the administration resources.	//app/policy/ASI/admin
BLM.wlesadmin. defaultdirectory	Used by BLM to locate the Administrator user when the user's directory is not provided by the BLM client at the time of making connection. This directory stores the administration server user and user groups that are used to boot the server and BLM API login. By default, admin user IDs are maintained in the asi admin directory and custom identities for application-related users would be stored in a directory other than the asi directory. You do not need to change this unless you are making changes to the default admin policy.	asi
BLM.wlesadmin. AuditWebserviceURL	The URL of the Web Service host to which BLM directs authorization audit events. You do not need to change this parameter unless you have changed the IP address and port on which the Audit Web Service runs.	https://127.0.0.1:7014

Table 5-2 Miscellaneous Configuration Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin.AuditRetries	Number of times the server will try to send audit events to the Audit Web Service before giving up. This must be an integer greater than 0. If the server cannot connect to the Audit Web Service, no exception is thrown, but a debug message will note the failure.	2
BLM.wlesadmin.contextsize	When the BLM reaches a number of connections equal to the `contextsize` value, including the connections that have already timed out, the BLM will try to drop the timed-out connections that have not been accessed for a number of seconds equal to or greater than the `sessionTimeout` value. Set a lower value for more frequent clean-up as compared to default value of 40.	40
BLM.wlesadmin. sessionTimeout	When the BLM has a number of connections equal to the `contextsize` value, it will try to drop connections that have not been accessed for a number of seconds equal to or greater than the `sessionTimeout` value not been accessed for a number of seconds equal to or greater than the `sessionTimeout` value.	7200 seconds (2 hours)

Table 5-2 Miscellaneous Configuration Parameters

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

Table 5-2 Miscellaneous Configuration Parameters

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

Logging Configuration Parameters

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

Table 5-3 Logging Configuration Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin.logLevel	Determines which events get logged. Valid values are integers from 0 to 63. The value is interpreted as a bitfield. Add the levels together to determine the value. The following log levels are defined: 0 (error) Always be log errors 1 (log) Enable log output. 2 (dbg) Enable debug output. 4 (eviction) Log any session eviction that takes place to free up idle connections. 8 (exceptions) Exceptions thrown by the BLM server. 16 (comTest calls) Server heartbeat calls to check that server is functioning correctly. 32 (soap calls) All IN/OUT SOAP messages at the transport level.	0
BLM.wlesadmin.logfile	Logging location.	BEA_HOME/ales32-admin/log/WLESblm.log
BLM.wlesadmin.errfile	Logging location for error log entries.	BEA_HOME/ales32-admin/log/WLESblm.log
BLM.wlesadmin.dbgfile	Logging location for debug log entries.	BEA_HOME/ales32-admin/log/WLESblm.log
BLM.wlesadmin.DbgOut	Logging location for C++ client debug entries, which do not have log levels assigned.	BEA_HOME/ales32-admin/log/WLESblm.log

Table 5-3 Logging Configuration Parameters

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

Database Configuration Parameters

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

Table 5-4 Database Configuration Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin.dbsystem	The database system used by the client. Valid values are: ORACLE92, ORACLE90, ORACLE81 SYBASE125, SYBASE120, SYBASE119 In addition, for backwards compatibility, the value ORACLE is treated as ORACLE81 and the value SYBASE is treated as SYBASE125.	ORACLE92
BLM.wlesadmin.dbserver	The database server name (the database service name for Oracle).	ASI.DB.EXAMPLE.COM
BLM.wlesadmin.dbname	Database name. This parameter is only applicable in Sybase and is ignored in Oracle.	sspolicy
BLM.wlesadmin.dbpolicyowner	Username of the policy owner. Generally, this will be the same as the `dblogin` user.	
BLM.wlesadmin.dblogin	Database login ID. This user ID must be granted database permissions. Usually it is the schema (policy) owner which has all the permissions.	
BLM.wlesadmin.dbpoolsize	Number of database connections in shared pool to be allocated for the BLM. Consult database administrator before setting this larger than 20, since there is typically a limited number of connections configured in the database server.	20

Table 5-4 Database Configuration Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin. dbconnidletimeout	If a database connection is idle for this number of seconds, it is disconnected. Use this is to make sure the BLM does not retain unused database connections for a long period.	600 seconds
BLM.wlesadmin.sqldebug	Enables or disables database SQL debugging (bit wise). In a production environment, set it to 0 or 1. In order for SQL debug logging to function, the BLM.wlesadmin.logLevel dbg bit must be set. 0 - hard database error 1 - soft database error (recoverable) 2 - SQL debugging 4 - Stored procedure debugging Add the levels together to come up with the value.	0
BLM.wlesadmin. fetchnumrows	Indicates how many delta elements should be returned from the database as part of the query resultset as opposed to loading all of the results at once. The subsequent get on the 1001th item would fetch the next 1000 results and so forth. To process the results, the collection configuration would take two passes with 500 items on the first and 500 on the next pass and so on. This value is a trade-off between latency during administration actions and latency at evaluation.	1000

CPP API Configuration Parameters

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

Table 5-5 CPP API Parameters

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

Distribution Parameters

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

Table 5-6 Distribution Parameters

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

Table 5-6 Distribution Parameters

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

Table 5-6 Distribution Parameters

Parameter	Description	Default or Example Value
PD.secondaryEnabled	Indicates whether a second server has been set-up to enable failover. This flag is used by the policy distributor. If set to true, the policy distributor on the primary server will temporarily give up control to the policy distributor on the second server to process the distribution request queue when the primary server detects that the network to the ARME is down.	False
PD.minFailedARMEDistributions	Defines the minimum number of ARME instances that is considered sufficient to indicate that the network from the local server to the ARMEs could be down when policies can not be distributed. When this number is exceeded, the policy distributor on the primary server will temporarily give up control (of processing the distribution request queue) to the second server if `PD.secondaryEnabled = true`. This value should be strictly greater than 1.	10
PD.networkDownWaitTimeout	Defines the time period (in milliseconds) for which the policy distributor on the primary server will give up control of processing the distribution request queue to a different (second) server if the network to the ARME instances is detected to be down. After this time period has lapsed, the primary server will resume control of processing the distribution queue.	600000 (10 minutes)
PD.lockUpdatePeriod	Defines the time period (in milliseconds) at which the policy distributor updates the DB lock entry while it holds the lock to indicate that the lock is active.	90000 (1.5 minutes)

Table 5-6 Distribution Parameters

Parameter	Description	Default or Example Value
PD.missedLockUpdates	Defines the number of missed updates of the DB lock entry that is considered sufficient to indicate that the lock is faulty. If the lock entry hasn't been updated for more than (PD.missedLockUpdates)*(PD.lockUpdatePeriod) milliseconds, it will be removed. This value should be strictly greater than 1. Please ensure that the time (PD.missedLockUpdates)*(PD.lockUpdatePeriod) is sufficiently large (at least ~5 minutes is suggested).	3
PD.lockRemoverMonitoringPeriod	Defines the time period (in milliseconds) at which the faulty lock remover thread will monitor the distribution lock DB entry to detect a faulty distribution lock (a lock left behind due to a server failure in the middle of a distribution) and remove it automatically. This value should be strictly greater than PD.lockUpdatePeriod.	180000 (3 minutes)

Default Timeout Parameters

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

Table 5-7 Default Timeout Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin. defaultSendTimeoutMs	When BLM sends a request, specifies the milliseconds to time out if it cannot send data.	10000
BLM.wlesadmin. defaultRecvTimeoutMs	When BLM makes a request to another server, specifies milliseconds to wait before disconnecting.	10000
BLM.wlesadmin. defaultConnectTimeoutMs	When the transport cannot connect to another server, specifies the milliseconds to wait before giving up.	10000

Override Timeout Parameters

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

Table 5-8 Override Timeout Parameters

Parameter	Description	Default or Example Value
BLM.wlesadmin.connectTimeout	When the transport cannot connect to the ARME, specifies the milliseconds to wait before giving up	10000
BLM.wlesadmin.sendTimeout	When BLM sends a request to the ARME, specifies the milliseconds to time out if it cannot send data.	10000
BLM.wlesadmin.requestTimeout	When a BLM makes request to the ARME, specifies milliseconds to wait before disconnecting.	10000

Table 5-8 Override Timeout Parameters

BLM.wlesadmin.relocateOnError	Controls whether to keep using the same connection (0) or relocate (1) if errors occur while communicating with the ARME.	1
BLM.wlesadmin.maxRetries	Maximum number of retries for the same ARME before relocation takes place	10

CHAPTER 6

Out-of-Box Attribute Retrievers

This section describes the out-of-box (OOTB) retrievers provided with Oracle Entitlements Server.

Note: For information about developing custom attribute retrievers, see "Attribute Retrievers" on page 7-4.

Overview

The OOTB attribute retrievers are used by ASI Authorization and ASI Role Mapping providers to retrieve attributes for use in policies. After an attribute retriever is implemented, it will retrieve the required attribute values (as specified in policy constraints) for use during policy evaluation.

To use an attribute retriever, you define its configuration, attributes, and attribute handling logic (or query) on the authorization or role mapping provider being used by the SSM instance. With the exception of the WebLogic Server 9.x/10.0 SSM, this is performed in the Administration Console. For WLS SSMs, this is performed using the WebLogic console.

When defining the attribute retriever's query, you may use OES system attributes. For example, the following use of the `%sys_user%` system attribute will retrieve the current user's "user_type" from the data source.

```
SELECT user_type FROM customer WHERE user_id=%sys_user%
```

In order to be included in policy constraints, each attribute retrieved from an external source must be defined as a dynamic attribute. To do this in the Entitlements Administration Application, navigate to the application containing the policies. Then use the application's **Extensions** > **Dynamic Attributes** tab.

Setting Up OOTB Attribute Retrievers

To set up an attribute and configure an attribute retriever:

1. Use the Administration Console to add the configuration settings and attribute properties on the provider's **Attribute Retriever** and **Attribute** tabs. These settings are described in the following sections:

   "RDBMS Attribute Retrievers" on page 6-2
   "LDAP Attribute Retrievers" on page 6-4
   "Service Data Objects (SDO) Attribute Retrievers" on page 6-5
   "ALES Identity Attribute Retrievers" on page 6-7

2. Distribute the changes to the SCM (or the XML file if no SCM is used).

3. Restart the SSM instance.

RDBMS Attribute Retrievers

RDBMS Attribute Retrievers retrieve attribute values from an RDBMS database. Table 6-1 indicates the properties used to configure a RDBMS Attribute Retriever.

Table 6-1 RDBMS Attribute Retriever Configurations

Property	Description
Conn Idle Timeout	Idle timeout in seconds for ALES database connection.
Description	Short description of the attribute retriever.
Driver	Defines the Java class name of the RDBMS database JDBC Driver.

Table 6-1 RDBMS Attribute Retriever Configurations

Property	Description
Database Name	Database instance name
Failed Server Retry Sec	Time in seconds after which a previously failed primary server is retried.
Login / Password	Login and password to access the database.
Name	The attribute retriever name in OES.
Pool Size	The database pool size that the Attribute Retriever use to access the RDBMS
System	The OES database type (Oracle 10, 92, 90, blank, Sybase 15, Sybase 125, Sybase, Pointbase, MS Sql Server, DB2)
Server	Defines the RDBMS database server URL. For failover, specify a comma separated list of primary and backup server URLs. If failure occurs when accessing the first server, then the next one is used. jdbc:oracle:thin:@smysore02.amer.acme.com:1521:orcl

Table 6-2 indicates the properties used to define the attributes used with a RDBMS attribute retriever.

Table 6-2 Attributes of a RDBMS Attribute Retriever

Property	Description
Name	The attribute name in OES.
Description	(Optional) A short description.
Retriever	For this attribute retriever type, the value is always: `RDBMSAttributeRetriever`

Attribute Query	The query to retrieve the attribute. May include OES system attributes.
	Examples:
	`SELECT name FROM table WHERE name = %sys_user%`
	`SELECT user_type FROM customer WHERE user_id=%sys_user%`

Table 6-2 Attributes of a RDBMS Attribute Retriever

Property	Description
Use Cache	Select checkbox to locally cache this attributes value.
TTL	If Use Cache is selected, specify the seconds after which the cache will expire.

LDAP Attribute Retrievers

LDAP Attribute Retrievers retrieve attribute values from a LDAP database. Table 6-3 indicates the properties used to configure a LDAP Attribute Retriever.

Table 6-3 LDAP Attribute Retriever Configurations

Property	Description
Description	Short description of the attribute retriever.
Failed Server Retry Sec	Time in seconds after which a previously failed primary server is retried.
Host	The host name or IP address of the LDAP server. For failover, please specify another host name separated by a comma similar to host1,host2. Please note that the ports and other parameters of the failover server is assumed to be same as the primary server.
Name	The attribute retriever name in OES.
Principal / Credential	The Distinguished Name (DN) of the LDAP user used to connect to the LDAP server and the user's credential.
Port	The LDAP server listening port
System	The LDAP type: iPlanet Open LDAP Active Directory Novell LDAP

Table 6-4 indicates the properties used to define the attribute whose value is returned by a LDAP attribute retriever.

Table 6-4 Attributes of a LDAP Attribute Retriever

Property	Description
Name	The attribute name in OES.
Description	(Optional) A short description.
Retriever	For this attribute retriever type, the value is always: `LDAPAttributeRetriever`
Attribute Query	The entry to retrieve the attribute. May include OES system attributes. Example: `uid=%sys_user%,cn=employees,ou=enterprisesecurity,ou=security,dc=amer,dc=bea,dc=com`
LDAP Filter	If needed, specify a filter to narrow the query scope. Default: `&(objectclass=*)(transactioncode= %sys_rule_obj_q%` May include OES system attributes.
Use Cache	Select checkbox to locally cache this attributes value.
TTL	If Use Cache is selected, specify the seconds after which the cache will expire.

Service Data Objects (SDO) Attribute Retrievers

Service Data Objects (SDO) is a specification that identifies a unified framework for data application development. It works with data from multiple data sources in the form of physical or Logical data services from ODSI (formerly AquaLogic Data Services Platform (ALDSP)). The ODSI services can in turn depend on multiple data services to retrieve data.

Table 6-5 indicates the properties used to configure a SDO Attribute Retriever.

Table 6-5 SDO Attribute Retriever Configurations

Property	Description
Description	Short description of the attribute retriever.
Failed Server Retry Sec	Time in seconds after which a previously failed primary server is retried.

Table 6-5 SDO Attribute Retriever Configurations

Property	Description
Initial Context Factory	Initial Context factory to use. For example, if the WebLogic initial context factory is used, the value is: `weblogic.jndi.WLInitialContextFactory` This class must be available in the SSM classpath.
Login / Password	Login and password to access the dataservice.
Name	The attribute retriever name in OES.
Service Lookup	The service to lookup, for example: `ld:DataServices/CustomerManagement/CustomerProfile`
SDOApplication	Name of the dataservice application to connect to.
Server URL	The server URL in the format `t3://<host>:<port>` For failover, specify additional URLs separated with a comma. For example: `t3://<host1>:<port1>,t3://<host2>:<port2>`

Table 6-6 indicates the properties used to define the attribute whose value is returned by a SDO attribute retriever.

Table 6-6 Attributes of a SDO Attribute Retriever

Property	Description
Name	The attribute name in OES.
Description	(Optional) A short description.
Retriever	For this attribute retriever type, the value is always: `SDOAttributeRetriever`

Table 6-6 Attributes of a SDO Attribute Retriever

Property	Description
Attribute Query	The SDO function name that retrieves the attribute. May be a method name that obtains the data from a logical or physical service Example: `getCustomer`
Parameter	A comma-separated list of parameters to use for SDO method name based on method signature. May include OES system attributes. Example: `%sys_user%,customer_id`

ALES Identity Attribute Retrievers

An ALES Identity Attribute Retriever retrieves the value of an identity attributes from the OES database. Table 6-7 indicates the properties used to configure a ALES Identity Attribute Retriever.

Table 6-7 RDBMS Attribute Retriever Configurations

Property	Description
Cache All Attributes TTL	Cache All Attributes TTL
Conn Idle Timeout	Idle timeout in seconds for ALES database connection.
Description	Short description of the attribute retriever.
Driver	The Java class name of the OES database JDBC Driver.
Database Name	Database instance name
Failed Server Retry Sec	Time in seconds after which a previously failed primary server is retried.
Login / Password	Login and password to access the database.
Name	The attribute retriever name in OES.
Pool Size	Database pool size.

Table 6-7 RDBMS Attribute Retriever Configurations

Property	Description
System	The OES database type (Oracle 10, 92, 90, blank, Sybase 15, Sybase 125, Sybase, Pointbase, MS Sql Server, DB2)
Server	The database server URL. For failover, specify a comma separated list of primary and backup server URLs. If failure occurs when accessing the first server, then the next one is used. `jdbc:oracle:thin:@smysore02.amer.acme.com:1521:orcl`

There is no need to create attributes for an ALES Identity Retriever. Simply create identity attributes using the Entitlements Administration Application and then use those attributes in the policy constraints as needed.

CHAPTER 7

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 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. These plug-ins can be used with the ASI Authorization, ASI Role Mapping, Log4j Audit Channel, and Database Authentication providers.

While the out-of-box security providers 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. Attribute retrievers retrieve embedded data from complex data objects or external data sources. Attribute converters convert context data to an internal attribute

format. Resource converters convert WebLogic Server and AquaLogic Enterprise Security data to an internal resource format.

- You can use Java extensions plug-ins to add 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 new evaluation functions or newly-available dynamic attributes.
- You can use the audit plug-ins to help format audit events that are generated by the Security Framework, the runtime API, or custom implementations.
- You can use the database authentication plug-in with the Database Authentication provider to customize authentication features.

Note: If you using the WLS SSM:

- The security provider JAR files in the WLS SSM (in directory `SSM/lib/providers/wls/v9`) are not compatible with those in the WLS 8.1 SSM (in directory `SSM/lib/providers`). For building classes to use with the WLS SSM, all JAR files should be placed in `SSM/lib/providers/wls/v9` directory and the JARs must be added to the CLASSPATH.
- Configuration of the ASI Authorization and the ASI RoleMapper providers needs to be done using both the Administration Console and the WebLogic Server Administration Console.

JAR Required for Provider Extensions

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

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

Authorization and Role Mapping Extensions

This product supports the use of Java-based plug-ins and language extensions with the out-of-box security providers such as `ASIAuthorizer` and `ASIRoleMapper`. You can use these plug-ins to augment authorization and role mapping processing logic.

The following sections describe plug-ins in detail:

Plug-in Types

Four types of Java-based plug-ins are supported: attribute retrievers, evaluation functions, resource converters, and attribute converters. These plug-ins can only be used with ASI Authorization and ASI Role Mapping providers.

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

1. Write a Java class to implement the interface. The following sections provide descriptions of each type of plug-in interface:
   - Attribute Retrievers
   - Custom Attribute Retrievers
   - Evaluation Functions
   - Resource Converters
   - Attribute Converters
2. Build a Jar file and place it in the SSM's `/lib/providers` directory (for the WLS SSM, use `/lib/providers/wls/v9` directory). If you use other 3rd party libraries in support of your plug-in, also copy those Jar files to the same location.
3. You can use Log4j libraries to insert debug statements in your plug-ins. The example found in `<BEA_HOME>\java-ssm\examples\AttributeRetriever` illustrates use of Log4j debugging messages.
4. Refer to the following topics in the Console Help and use the Administration Console to register the Java plug-ins in the desired security providers for the desired SSMs:
   - Configuring an ASI Authorization Provider
   - Configuring an ASI Role Mapping Provider

Attribute Retrievers

Attribute retrievers are used by ASI Authorization and ASI Role Mapping providers to retrieve attributes for use during runtime evaluation of policy. The out-of-the-box attribute retrievers can be configured in the Administration Console (for the WLS SSM this must be done using the WebLogic console).

There is no need to create an attribute retriever to retrieve values from LDAP or a database store. Such retrievers can be directly configured via the ASIAuthorizer provider's **Configuration** tab in the Administration Console.

Custom Attribute Retrievers

Dynamic attributes are often used to write policy constraints. In such cases the value of the dynamic attribute can come either from the application context or from an external source. If the external source is not a database store or LDAP, you must write a custom attribute retriever.

Examples:

- A custom attribute retriever could be written to obtain the value of a stock quote from a financial site via HTTP.
- If you use custom Authentication providers that store user attributes directly in the subject, a custom attribute retriever could be used to extract this information and make it available during policy evaluation.

`AttributeRetriever` is an interface in the `com.bea.security.providers.authorization` package that you can use to implement plug-ins for retrieving attributes.

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 7-1 lists and describes the methods provided by the `AttributeRetrieverV2` interface.

Table 7-1 AttributeRetrieverV2 Interface Methods

Method	Description
String[] getHandledAttributeNames()	Returns the names of attributes handled by attribute retriever. An attribute retriever may return at least one attribute name in this method. If the method returns null or an empty value, this attribute retriever will be called when any dynamic attribute has to be resolved.
Object getAttributeValue(String name,RequestHandle requestHandle,Subject subject,Map roles,Resource resource,ContextHandler contextHandler)	Retrieves the value of the named attribute. Additional authorization request data is made available to allow for more complex attribute retrieval. The parameters are as follows: • **name** — name of the needed attribute • **subject** — subject associated with the request • **RequestHandle** — the provider configuration parameters associated with the request, through which the function can get the values of other attributes if required. The `com.bea.security.providers.authorization.asi.ARME.evaluator.RequestHandle` interface is described in RequestHandle.getAttribute() Method. • **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 Valid Java return types are `String` and `String[]`.

RequestHandle.getAttribute() Method

The `com.bea.security.providers.authorization.asi.ARME.evaluator.RequestHandle` object has the `getAttribute()` and `appendReturnData()` methods. These are defined as follows:

```
public AttributeElement getAttribute(String name, boolean typeCheck)
throws ArmeRuntimeException, BadParameterException,
CredvarException,BoolEvalInternalException, NotReadyException;
public void appendReturnData(String name, Object data);
}
```

Of the two methods, the `RequestHandle.getAttribute()` is of most interest to `AttributeRetrieverV2` implementations. This method gets the named attribute and its value, and optionally type-checks the value. It returns the attribute name and value as a `com.wles.util.AttributeElement` object. For example if your attribute retriever needed to make an HTTP request to obtain the stock quote, then the application context can contain the URL of the stock quote service and you would use `getAttribute()` to obtain this value. This function will also help in obtaining the value of all the runtime system attributes (sys_*) such as `sys_dir`, `sys_app_q` etc. Please refer to the *Advanced Topics* section in the *Policy Managers Guide* for a full list of system attributes.

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

```
try {
AttributeElement attrElement =
requestHandle.getAttribute("sys_subjectgroups", true);
Object value = null;

//It must be a list attribute
if(!attrElement.isList()) {
return null;
}
//transfer AttributeElement value to a list of String type object.
List subjectGroupList = (List)attrElement.getValueAs(String.class);
value = String.valueOf(subjectGroupList.size());
return value;
} catch (Exception e) {
//failed to retrieve attributes.
return null;
}
```

The `AttributeRetrieverV2` implementation can use the `RequestHandle.getAttribute()` method to get the value for user and resource attributes. However, when this method is used to get values of dynamic attributes that may be handled by other Attribute Retrievers (including built-in database and LDAP), it is possible that the `ASIAuthorizer` may or may not have computed it yet when the custom `AttrbuteRetriever` was called. In order to make sure that required attributes are present, always list them before the attribute in the policy constraint via

the use of sys_defined(<attr>) function. For example, the attribute retriever depends on the URL attribute being present in the RequestHandle. If this attribute is also computed by another attribute retriever, instead of being passed in as an application context, then the rule constraint would have to look something like this: `IF sys_defined(URL) and BEAS_quote >= 19;`

`RequestHandle.getAttribute()` and `RequestHandle.appendReturnData()` are also applicable to Evaluation functions, which are described in the next section.

ASI providers always evaluate the minimum number of required policies and attributes are retrieved only when a policy that references them is evaluated. Therefore, not all attribute retrievers are guaranteed to be called for all policy evaluations. OES attributes retrievers are not called unless the policy has been explicitly evaluated and the attribute can not be obtained from the application context or is not an associated user attribute.

Configuring a Custom Attribute Retriever

To configure a custom attribute retriever, perform the following steps:

1. Implement a custom attribute retriever and create a JAR file.

   The `com.bea.security.providers.authorization.asi.AttributeRetrieverV2` interface is in `<BEA_HOME>\ales32-ssm\<ssm-type>\lib\providers\ASIAuthorizer.jar` (for WLS SSM, use `<BEA_HOME>\ales32-ssm\wls-ssm\lib\providers\wls\v9\ASIAuthorizer.jar`). Include this file and `<BEA_HOME>\ales32-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 SSM's installation directory (either the WLS or Java SSM). For example, the WLS SSM default directory is `<BEA_HOME>\ales32-ssm\wls-ssm`.

3. Do one of the following:

   – (For all SSMs except the WLS SSM) In the left pane of the Administration Console, click the ASI Authorization provider configured for the SSM instance and select the **Attribute Retrievers** tab in the right pane.Then create a new custom attribute retriever for the plugin using the JAR file you just created. Select **CustomAttributeRetriever** as the type.

   – (WLS SSM Only) In the WebLogic Server Administration Console, select **Home > Summary of Security Realms > asiadmin > Providers > ASI Authorization Provider** and create a new attribute retriever for the plugin using the JAR file you just created. Select **CustomAttributeRetriever** as the type.

4. In the left pane, select the **Deployment** node. Then select the **Configuration** tab in the right pane and deploy the configuration change to the SSM.
5. Restart the SSM.

Configuring Caching for Custom Attributes

Caching options can be specified for all attribute values returned by a custom attribute retriever. To set these options, select the Custom Attribute Retriever's **Cache All Attributes** checkbox and specify how long the values should be cached in the **Cache All Attributes TLL** field.

It is also possible to specify a caching requirement for individual attributes. When this is done, the caching parameters take precedence over those defined for the custom attribute retriever.

To specify a caching requirement for an attribute, perform the following steps:

1. On the ASI Authorization provider's **Attributes** tab, click on **Configure a new Attribute**.
2. Enter the attribute name and click **Create**.
3. In the **Retriever** field, select the custom attribute retriever that returns the value for this attribute.
4. Select the **Use Cache** checkbox.
5. In the **TTL** field, specify how long the values should be cached.

Evaluation Functions

You can use a named evaluation function to augment built-in authorization and role mapping logic. This method is invoked when the policy contains a custom evaluation function with a matching name. For example:

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

where `myFunc()` is the custom evaluation function name.

A custom evaluation function is implemented as a method in a class that should also implement `init()` and `shutdown()` methods. This class may contain one or more custom evaluation functions. You can choose any name for custom evaluation function so long as the corresponding method name matches the name that is used in a policy. Custom evaluation functions can be passed arguments consisting of constants or names of other attributes, including dynamic attributes.

Note: Since all evaluation functions share a common namespace, two functions cannot have the same name.

Custom Initialization/Shutdown Interface

This section describes the interface for writing custom evaluation functions.

The `com.bea.security.providers.authorization.asi.InitializationShutdownFunction` interface can be used to implement custom initialization and/or shutdown steps that are specific to your evaluation functions.

The `init()` method is invoked during the Authorization or Role Mapping provider initialization and can be used to initialize some plug-in specific data, such as a connection pool. The `shutdown()` method is invoked during the Authorization or Role Mapping provider shutdown and can be used to free data allocated in the "`init()`" method.

When the `init()` method is called, it is passed all configuration parameters for Authorization provider or Role Mapping provider. For the complete list of available parameters, see javadocs for `InitializationShutdownFunction`.

Table 7-2 lists and describes the methods provided by the `InitializationShutdownFunction` interface required to implement an Evaluation Function.

Table 7-2 InitializationShutdownFunction Interface

Method	Description
void init(Map config)	This method is called during the Authorization or Role Mapping provider's initialization and is passed the list of configuration parameters for the provider's
void shutdown()	This method is invoked during the Authorization or Role Mapping provider shutdown

Table 7-2 InitializationShutdownFunction Interface

Method	Description
Evaluation Function (Not part of InitializationShutdownFunction interface)	
public boolean some_method_name(RequestHandle requestHandle, Object[] args, Subject subject, Map roles, Resource resource, ContextHandler contextHandler)	This method receives the name of the argument (can be an attribute) passed in during constraint evaluation. Additional request information is made available via `requestHandle` to allow the function to obtain the value of the named attribute. The parameters are as follows: • **requestHandle** — The attributes container associated with the request, through which the method can get required attribute values and also return data. See RequestHandle.getAttribute() Method for a description of how to get values for attributes. See RequestHandle.appendReturnData() Method for a description of how to return data. • **args** — An array of method arguments. Each element is either `<code>null</code>`, or a String • **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 should return `True` or `False`.

RequestHandle.appendReturnData() Method

The `RequestHandle.appendReturnData()` method can be used to create a named response attribute with a specified value. It is equivalent to using the `report_as()` function from inside a custom evaluation function.

Table 7-3 RequestHandle.appendReturnData()

Method	Description
void appendReturnData (java.lang.Stringname, java.lang.Object data) throws RuntimeException	Parameters: **`name`** — Name of return attribute **`data`** — Attribute value to be returned if the rule evaluates to *true*. Throws: `RuntimeException` — may throw `RuntimeException` if there is problem in converting to the Response object.

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

RequestHandle.getAttribute() Method

When the evaluation function is called, the runtime system passes in the literal string as an argument to the function as part `args[0]`. To obtain the value of the attribute passed in, make use of the `RequestHandle.getAttribute()` method. This method gets the named attribute and its value, and optionally type-checks the value. It returns the attribute name and value as a `com.wles.util.AttributeElement` object. The `AttributeElement` object represents an attribute name/value pair for a single element or a list.

The following code fragment provides an example.

```
try {
AttributeElement strLength = requestHandle.getAttribute((String)args[0],
true);
if(strLength != null) {
if(strLength.isList()) {
// string_longer_then: first argument not a single value
return false;
}
intCompLength =
Integer.parseInt((String)strLength.getValueAs(String.class));

} else {
```

```
// numerical constant will be passed in as is.
try {
intCompLength = Integer.parseInt((String)args[0]);
} catch(NumberFormatException ne) {
//value format is an error, and there is no attribute in the requestHandle.
throw new MissingAttributeException("missing attribute: " + args[0],
(String)args[0]);
}
}
} catch(Exception e) {
//caught exception while getting attribute
throw new RuntimeException("failed while getting attribute: " +
(String)args[0] + ". Exception: " + e.getMessage());
}
```

It is also possible to use dynamic attributes as arguments to an evaluation function such that the dynamic attribute is handled by a Custom Attribute Retrievers. When the evaluation function calls `requestHandle.getAttribute((String)args[0], true)`, the value will be returned after the custom attribute retriever is called.

Configuring a Custom Evaluation Function

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

1. Implement a custom evaluation function plug-in and create a JAR file.

   The `com.bea.security.providers.authorization.asi.InitializationShutdownFunction` class is in `BEA_HOME\ales32-ssm\<ssm-type>\lib\providers\ASIAuthorizer.jar` (for WLS SSM, use `BEA_HOME\ales32-ssm\wls-ssm\lib\providers\wls\v9\ASIAuthorizer.jar`). Include this file and `<BEA_HOME>\ales32-ssm\<ssm-type>\lib\asi_classes.jar` in the classpath when compiling the custom evaluation function.

2. Place the Jar file in the SSM's `/lib/providers` directory (for the WLS SSM use `/lib/providers/wls/v9`).

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

For the WLS SSM, use the WebLogic Administration Console to configure the Authorization and Role Mapping providers.

4. Repeat step 3 to register the extensions plug-in with the ASI Role Mapping provider.
5. In the left pane, click **Deployment** and select the **Configuration** tab. Then deploy the configuration change to the SSM.
6. Restart the SSM.

Resource Converters

Resource converters are used by ASI Authorization and ASI Role Mapping providers to convert application-specific resources to an internal OES resource format that is recognized. Out-of-box resource types will convert WLS-style and OES-style resource types.

To add support for additional resource types, define a resource converter to allow the ASI Authorization provider to protect the new resource types. `ResourceConverter` is an interface in the `com.bea.security.providers.authorization.asi` package.

Table 7-4 lists and describes the methods provided by the `ResourceConverter` interface.

Table 7-4 ResourceConverter Interface Methods

Method	Description
String[] getHandledTypes()	This method is called when the plug-in is instantiated to register the resource types supported by this custom resource converter. The Security Framework represents resource types internally as strings such as `<http>` or `<wlp>`. The return parameter should be a list of strings that contains the names of the resource types that this custom resource converter is going to handle.

Table 7-4 ResourceConverter Interface Methods

Method	Description
AccessElement convertResource(Resource resource, ContextHandler contextHandler) throws ResourceConversionException	This method is called when the ASI Authorizer or Role Mapper encounters resources types that match what was registered via `getHandledTypes()` function. This method would have the knowledge to extract information from the `Resource` and `ContextHandler` objects to obtain the OES representation of resource and privilege. Additional information that can be obtained is the application name and input attributes extracted from the `Resource` or `ContextHandler`: Use the following guidelines when dealing with application name and where to place the resource names in the resource hierarchy: • If no application name is specified, the resource is assumed to be a child of the /shared resource node as specified in the provider configuration. For example, `App_name=""` then `ALES_res= //app/myapp/shared/res1/res2` • If an unqualified application name is specified, the resource is assumed to be a child of the application name, which is a child of the default deployment parent node. For example, `App_name="trading"` and `App_deployment_parent="myapp"` then `ALES_res=//app/myapp/trading/res1/res2` • If a fully-qualified application name is present, the resource is assumed to be a child of that application name. For example, `App_name="//app/app1/trading"` then `ALES_res="//app/app1/trading/res1/res2"`. If the resource converter is unable to obtain enough information to create AccessElement from input parameters, it should throw a `ResourceConversionException` indicating to the provider that this resource cannot be converted into a equivalent OES format.
Object getAttributeValue(Resource resource, String name,ContextHandler contextHandler)	This method must obtain the value of a attribute that is passed in as an argument. It is the task of `ResourceConverter` developer to determine how the `ResourceConverter` obtains the value. The plug-in may return null if the value is not found.

Configuring a Custom Resource Converter

To configure a custom resource converter, 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 create a JAR file.

   The `com.bea.security.providers.authorization.asi.ResourceConverter` class is present in `<BEA_HOME>\ales32-ssm\<ssm-type>\lib\providers\ASIAuthorizer.jar` (for WLS SSM, use `<BEA_HOME>\ales32-ssm\wls9-ssm\lib\providers\wls\v9\ASIAuthorizer.jar`). Include this file and `<BEA_HOME>\ales32-ssm\<ssm-type>\lib\asi_classes.jar` in the classpath when compiling the custom resource converter.

2. Copy the JAR file in the `/lib/providers` directory (or `/lib/providers/wls/v9` if using the WLS SSM) in the SSM's installation directory. For example, the default directory for the WLS SSM is `<BEA_HOME>\bea\ales32-ssm\wls-ssm`.

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

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

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

6. Restart the SSM.

Attribute Converters

Attribute converters are used by ASI Authorization and ASI Role Mapping providers to convert Java attribute types to the internal attribute types, and vice versa. Out-of-box attribute converters that convert between Java and internal types are provided. For a list of supported attribute types, see the *Attribute Declarations* in the *Policy Managers* guide.

To add new attribute types, implement a `TypeConverter` interface to handle the conversion. Attribute types are the also called 'credential' types. They are listed in the Administration Console integer, date, string etc. These are Java equivalents, but are represented as internal attribute types.

`TypeConverter` is an interface in the `com.wles.util package`.

Table 7-5 lists and describes the `TypeConverter` interface.

Table 7-5 TypeConverter Interface Methods

Method	Description
Class getType()	Returns the Java class type which this converter converts. For an `IntegerConverter` this call would return `Class.forName("java.lang.Integer");`
String getASITypeName()	Returns the type name. For an `IntegerConverter` this call would return `"integer";"`
String convertToASI(Object javaFormat) throws UnsupportedTypeException	Converts a java object into a string.
Object convertFromASI(String asiFormat) throws TypeConversionException	This method converts a 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 create a JAR file.

   The `com.wles.util.TypeConverter` class is present in `<BEA_HOME>\ales32-ssm\<ssm-type>\lib\asi_classes.jar`. Include this file in the classpath when compiling the custom attribute converters.

2. Copy the JAR file in `/lib/providers` directory (or `/lib/providers/wls/v9/ASIAuthorizer.jar` for the WLS SSM) in the SSM's installation directory (either the WLS or Java SSM). For example, the WLS SSM default directory is `<BEA_HOME>\bea\ales32-ssm\wls-ssm`.

3. In the left pane of the Administration Console, click the ASI Authorization provider configured for the SSM instance and select the **Advanced** tab in the right pane. Then enter

fully-qualified name of the custom converter in the **Attribute Converters** field and click **Apply**. If you are using the WLS SSM, configuration changes to the ASI Authorization provider must also be made using the WebLogic Server Administration Console.

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

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 7-18
- "Audit Plug-in Renderer Class" on page 7-19

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 7-19 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 it in `/lib/providers` (or `/lib/providers/wls/v9/ASIAuthorizer.jar` for the WLS SSM) in the SSM's installation directory. For example, the WLS SSM default location is: `<BEA_HOME>\ales32-wls-ssm\lib\providers`.
3. Use the Administration Console to register the audit plug-in for the desired Log4j Audit Channel provider. For instructions, see *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 7-6 describes a sample `AuditEventRenderer` class.

Table 7-6 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 *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 want to implement a plug-in by following the guidelines provided with the plug-in.

A custom database authentication plug-in must also extend the `DBMSPlugin` abstract class (located in the `com.bea.security.providers.authentication.dbms.DBMSPlugin` package). The `DBMSPlugin` abstract class implementation must include the methods described in Table 7-7.

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

Table 7-7 lists and describes the methods provided by the `DBMSPlugin` abstract class.

Table 7-7 DBMSPlugin Methods

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

Table 7-7 DBMSPlugin Methods

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

Table 7-7 DBMSPlugin Methods

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

The `options` object is a map containing optional information for use by the plug-in. The most common options and 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 8

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 8-1. This table also indicates the sub-interfaces that implement the `AuditContext` interface. These interfaces are documented in the *Security Provider SSPI API Reference.*

Table 8-1 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 8-2 lists WebLogic 9.x audit events.

Table 8-2 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
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

Table 8-2 WebLogic 9.x Audit Events

Audit Event Name	Interface Class	Interfaces Implemented	
		Audit Event	Audit Context
Role Deployment Audit Event	weblogic.security.spi.AuditRoleDeploymentEvent	Yes	No
Role Mapping Audit Event	weblogic.security.spi.AuditRoleEvent	Yes	Yes

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

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

What Events are Audited?

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

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

Each audit record has the following format:

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

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

To be audited, you can select which severity the audit event must equal or be greater than; and you can select the types of `AuditEvents` by setting the Custom Audit Events attribute. If an `AuditEvent` implements or is an instance of any of the classes listed, then you can audit it. Only new custom events need to be listed here. The default events already exist and are controlled by selecting either: `DISABLED`, `WITH_CONTEXT`, or `WITHOUT_CONTEXT` on the Details tab for the Log4j Auditor page in the Administration Console. For a list of audit events, see "Audit Events" on page 8-1.

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

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

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

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

Custom Audit Context Extensions

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

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

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

Adding Application Context from the BLM API

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

An Audit Context is a name=value pair that contains additional audit data that is made available to the Audit provider. Like the Audit Context, the Application Context is also a name=value pair data structure, and it contains additional application-specific audit data that is appended to the Audit Context when audit messages are written.

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

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

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

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

Audit Event Interfaces and Audit Events

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

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

- AuditAtnEvent
- AuditAtzEvent
- AuditMgmtEvent
- AuditCredentialMappingEvent
- AuditPolicyEvent
- AuditRoleDeploymentEvent

- AuditRoleEvent
- BLM Management Events
- ProviderAuditRecord

For a list of the events that are audited for the default Admin policy, see "BLM Management Events" on page 8-12.

AuditAtnEvent

Authentication audit events are posted by the security framework. Table 8-3 describes the conditions under which the event is posted and severity level of the event.

Table 8-3 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 8-3 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
Security Framework	A user has been locked by the user lockout manager.	Failure
Security Framework	A user has been unlocked	Success
Security Framework	A user lockout has expired	Success

AuditAtzEvent

Authorization audit events are posted by the security framework. Table 8-4 describes the conditions under which the events are posted and severity level of the event.

Table 8-4 Authorization Audit Events

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

AuditCredentialMappingEvent

Credential Mapping audit events are posted by the security framework. Table 8-5. describes the condition under which the events are posted and severity level of the event.

Table 8-5 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 8-6 describes the conditions under which the events are posted and lists the event severity level.

Table 8-6 Audit Policy Events

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

Table 8-6 Audit Policy Events (Continued)

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

AuditRoleDeploymentEvent

The security framework posts audit role deployment events when roles are deployed to or undeployed from a role mapping provider. Table 8-7 describes the conditions under which the events are posted and lists the event severity level.

Table 8-7 Audit Role Deployment Events

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

Table 8-7 Audit Role Deployment Events (Continued)

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

AuditRoleEvent

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

Table 8-8 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

BLM Management Events

Table 8-9 lists and describes the BLM management events that are audited.

Table 8-9 BLM Management Audit Events

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

Table 8-9 BLM Management Audit Events (Continued)

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

Table 8-9 BLM Management Audit Events (Continued)

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

Table 8-9 BLM Management Audit Events (Continued)

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

Table 8-9 BLM Management 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, resource, subject_name, 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, constraint	Modify a delegate policy. The “action”, “resource”, and “subject_name” attributes are lists.
Policy/Action/Role/ Instance	create	action	Create a new role.
	delete	action	Delete a role.
	rename	action, new_name	Rename a role.
Policy/Action/ Privilege/Instance	create	action	Create a privilege.
	delete	action	Delete a privilege.
	rename	action, new_name	Rename a privilege.

Table 8-9 BLM Management Audit Events (Continued)

Policy Element	Action	Type	Event Description
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.
	addMember	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.
	modify	title, owner, effect_type, subjects, actions, resources, delegator	Modify a policy query.
	Read		Read policy inquiry.
Policy/Repository	deploy Update	resource, directory	Deploy a policy update. The “resource” is the distribution node; all nodes below it may be affected. This check is made for each chosen distribution point
	deploy Structural Change	deleted_directories, deployed_engines, deleted_engines, deleted_bindings, deleted_applications	Deploy a structural change.

Table 8-9 BLM Management Audit Events (Continued)

Policy Element	Action	Type	Event Description
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.
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.
Transaction	begin	transaction	Begin transaction
	commit	transaction	Commit transaction
	rollback	transaction	Rollback transaction

AUDITBASE

Table 8-10 indicates the Auditbase events posted by the PD during policy distribution.

Table 8-10 Auditbase Events

Component	Description	Severity
PD	Requesting initial policy succeed.	Success
PD	Requesting initial policy failed	Failure
PD	Sending begin policy update succeed.	Success
PD	Sending begin policy update failed	Failure
PD	Sending prepare to commit succeed	Success
PD	Sending prepare to commit failed	Failure
PD	Sending commit succeed	Success
PD	Sending commit failed	Failure
PD	Processing async policy distribution request succeeded	Success
PD	Processing async policy distribution request failed	Failure
PD	Processing async structural change distribution request succeeded	Success
PD	Processing async structural change distribution request failed	Failure
PD	Enqueuing distribution request succeeded	Success
PD	Enqueuing distribution request failed	Failure

Distribution Status Request

As indicated in Table 8-11, the adminstration console posts distribution status request events after user distributes policy from console and distribution result page is rendered.

Table 8-11 Distribution Status Request Events

Component	Description	Severity
Administration Console	Posted after the following event occurs: Distribute policy from admin console and distribution result page is rendered.	Information

Distribution

As indicated in Table 8-12, the administration console posts distribution events when user distributes structural change from console.

Table 8-12 Distribution Status Request Events for Structural Change

Component	Description	Severity
Administration Console	Posted after the following event occurs: Distribute structural change from the administration console.	Information

Examples

When OOTB ALES log4j Audit Provider is configured, following message can be found in `secure_audit.log`, text in bold is event type:

1. Audit events from BLM:

```
2008-08-26 11:16:46,712 [JettySSLListener1-1] SUCCESS ASI_AUDIT -
BLMManagementEvent -- BLMManagementEvent/SUCCESS/Created, grant,
//role/Admin, //app/policy/ASIRecovery, //user/asi/bbb/, , true,
R...
```

2. Audit events from administration console:

```
2008-08-26 11:17:10,368 [[ACTIVE] ExecuteThread: '0' for queue:
'weblogic.kernel.Default (self-tuning)'] INFORMATION ASI_AUDIT - Policy
Distribution -- Policy was distributed for the following resources

2008-08-26 11:17:10,384 [[ACTIVE] ExecuteThread: '0' for queue:
'weblogic.kernel.Default (self-tuning)'] INFORMATION ASI_AUDIT -
Policy Distribution --    //app/policy

2008-08-26 11:17:10,415 [[ACTIVE] ExecuteThread: '0' for queue:
'weblogic.kernel.Default (self-tuning)'] INFORMATION ASI_AUDIT -
Distribution Status Request -- Distribution Status :
10,//user/asi/system/,10,computing update,Tue Aug 26 11:17:10 CST 2008

2008-08-26 11:17:10,493 [[ACTIVE] ExecuteThread: '0' for queue:
'weblogic.kernel.Default (self-tuning)'] INFORMATION ASI_AUDIT -
Distribution Status Request -- Distribution Status :
8,//user/asi/system/,100,distribution done,Tue Aug 26 10:44:07 CST 2008

Distribution Result(ARME.admin.server.asi.cding02, //bind/asiadmin,
cding02, true)

2008-08-26 11:17:10,509 [[ACTIVE] ExecuteThread: '0' for queue:
'weblogic.kernel.Default (self-tuning)'] INFORMATION ASI_AUDIT -
Distribution Status Request -- Distribution Status :
7,//user/asi/system/,100,distribution done,Tue Aug 26 10:41:18
CST 2008
...
```

3. Audit events from PD:

```
2008-08-26 11:17:12,493 [Thread-28] SUCCESS ASI_AUDIT - AUDITBASE --
AUDITBASE/SUCCESS/PD, performing sendBeginPolicyUpdate operation to
ARME.admin.server.asi.cding02 with policy id 9 succeeded.

2008-08-26 11:17:12,493 [Thread-28] SUCCESS ASI_AUDIT - AUDITBASE --
AUDITBASE/SUCCESS/PD, performing prepareToCommit operation to
ARME.admin.server.asi.cding02 with policy id 10 succeeded.

2008-08-26 11:17:12,493 [Thread-28] SUCCESS ASI_AUDIT - AUDITBASE --
AUDITBASE/SUCCESS/PD, performing commit operation to
ARME.admin.server.asi.cding02 with policy id 10 succeeded.

2011-03-30 12:19:12,877 [JettySSLListener1-1] SUCCESS AUDITBASE -
AUDITBASE -- AUDITBASE/SUCCESS/PD, accepting the initial policy request
from ARME.admin.server.asi.VPOPIC-LAP with policy id 2 succeeded.

2011-03-30 14:35:00,717 [JettySSLListener1-1] SUCCESS AUDITBASE -
AUDITBASE -- AUDITBASE/SUCCESS/PD, processing async policy distribution
request with distribution tracking id 4201 succeeded.

2011-03-30 14:42:19,581 [JettySSLListener1-1] FAILURE AUDITBASE -
AUDITBASE -- AUDITBASE/FAILURE/PD, processing async policy distribution
request failed.
```

```
2011-03-30 14:35:07,748 [JettySSLListener1-1] FAILURE AUDITBASE -
AUDITBASE -- AUDITBASE/FAILURE/PD, processing async structural change
distribution request failed.

2011-03-30 14:48:40,648 [JettySSLListener1-1] SUCCESS AUDITBASE -
AUDITBASE -- AUDITBASE/SUCCESS/PD, enqueuing distribution request with
distribution tracking id 4214 succeeded.
```

ProviderAuditRecord

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

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

In the providers, the `instanceof` operator can be used to distinguish between the WLS Security Framework interfaces and Oracle Entitlements Server interfaces. For example:

```
if ( myauditrecord instanceof com.bea.security.spi.ProviderAuditRecord){

// This is an audit record that uses the enhanced SSPI.
} else {
// This is a WLS audit record. You must test further for more object
// types and handle them explicitly.
}
```

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

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

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

```
public void writeEvent(AuditEvent event){
// write the event out to the sample auditor's
// log file using the event's "toString" method.
// followed by the string version of the application
// context name value pairs.
ProviderAuditRecord par = (ProviderAuditRecord) event;
ContextHandler ch = par.getContext();
String[] names = ch.getNames();
StringBuffer ctxReader = new StringBuffer();
for ( int i=0; i < names.length; i++ ) {
String value = (String) ch.getValue(names[i]);
ctxReader.append(names[i]).append("=").append(value);
}
log.println(event + “Context = “ + ctxReader.toString());
}
```

Other Audit Events

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

AuditApplicationVersionEvent

Table 8-13 describes the conditions under which the event is posted and severity level of the event.

Table 8-13 Application Version Audit Event

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

Table 8-13 Application Version Audit Event

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

AuditCertPathBuilderEvent and AuditCertPathValidatorEvent

These events are posted by the CertPathBuilder providers.

AuditLifecycleEvent

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

Table 8-14 Audit Life Cycle Events

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

Additional Audit Event Information

Some implementations of the `AuditEvent` interface contain additional information that can be accessed by the providers and security framework. All interfaces that extend the

`weblogic.security.spi.AuditEvent` interface or all the implementations of that interface have the following information available:

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

The following sections provide additional information about specific audit events:

- "Authentication - AuditAtnEvent" on page 8-26
- "Authorization - AuditAtzEvent" on page 8-27
- "AuditCredentialMappingEvent" on page 8-27
- "Policy Events - AuditPolicyEvent" on page 8-28
- "Role Mapping - AuditRoleEvent" on page 8-29
- "AuditRoleDeploymentEvent" on page 8-29
- "Predicate Events - AuditPredicateEvent" on page 8-31

Authentication - AuditAtnEvent

The `AuditAtnEvent` interface provides an interface for audit providers to determine the instance types of the extended authentication event type objects. Table 8-15 describes the event properties.

Table 8-15 Authentication - AuditAtnEvent Event Type Property Values

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

Table 8-15 Authentication - AuditAtnEvent Event Type Property Values

Event Type Property	Description
IMPERSONATEIDENITY	Represents the impersonate identity authentication type.
VALIDATEIDENTITY	Represents the validate identity authentication type.

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

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

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

Authorization - AuditAtzEvent

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

This event has the following information available:

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

AuditCredentialMappingEvent

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

- The resource for which the credentials are requested causing this event

- The requestor subject for the operation
- The subject of the initiator of the operation
- The string representation of the initiator of the operation
- The credential types requested in the operation
- Object array of credentials returned by the operation
- The ContextHandler object.

Policy Events - AuditPolicyEvent

The AuditPolicyEvent interface determines the instance types of extended Authorization event type objects. Table 8-16 describes the event subtypes.

Table 8-16 Policy Event- AuditPolicyEvent

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

All AuditPolicyEvents have:

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

Policy Deployment - AuditPolicyDeployEvent

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

- Role names associated with this policy deploy event

Policy Undeployment - AuditPolicyUndeployEvent

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

Role Mapping - AuditRoleEvent

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

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

- The Subject that is attempting to access the resource associated with this `AuditRoleEvent`.
- The resource attempting to be accessed by the subject associated with this `AuditRoleEvent`.
- Role objects that are being manipulated by the action being audited.
- The ContextHandler object, as described in "ContextHandler Object" on page 8-31.

AuditRoleDeploymentEvent

The `AuditRoleDeploymentEvent` provides an interface for auditing providers to determine the instance types of extended Role deployment event type objects. Table 8-17 describes the event subtypes.

Table 8-17 Role Deployment - AuditRoleDeploymentEvent

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

Role Deployment - AuditRoleDeployEvent

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

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

Role Undeployment - AuditRoleUndeployEvent

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

- Resource for which the role is being undeployed.
- The name of the role being undeployed.

- The sublect that is attempting to undeploy the role.

Role Delete App Event

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

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

Role Deployment Start and End Events

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

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

Predicate Events - AuditPredicateEvent

The `AuditPredicateEvent` event is used by Auditing providers to determine the instance type of extended predicate event type objects. A predicate event occurs when a policy expression is either registered or unregistered in the Administration Console. Table 8-18 describes the event subtypes.

Table 8-18 Predicate Events - AuditPredicateEvent

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

ContextHandler Object

A `ContextHandler` is a class that obtains additional context and container-specific information from the resource container, and provides that information to security providers making access or role mapping decisions. The `ContextHandler` interface provides a way for an application or container to pass additional information to a Security Framework call, so that a security provider can obtain contextual information beyond what is provided by the arguments to a particular

method. A `ContextHandler` is essentially a name/value list and as such, it requires a security provider to know what names to look for. In other words, use of a `ContextHandler` requires close cooperation between the resource container and the security provider. Each name/value pair in a `ContextHandler` is known as a context element, and is represented by a `ContextHandler` object.

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

Policy Administration Messages

When a policy is modified or deployed using the Entitlements Administration Application or BLM Java API, informative messages are audited. The following information is available in these messages:

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

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

Using Custom Audit Providers

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

CHAPTER 9

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 9-1 describes the attributes supported by this provider.

Table 9-1 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 9-1 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 9-2 describes the attributes supported by this security provider.

Table 9-2 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	"{shared.dir}/keys/demoProviderTrust.jks"	The location of the Trusted Keystore stored in the TrustedCAKeystoreType keystore format. {shared.dir} 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 9-2 ALESIdentityAsserter

Attribute Name	Default Value	Description	List
TrustedKeystore	"{shared.dir}/keys/demoProviderTrust.jk"	The Location of the Trusted Keystore stored in the TrustedKeystoreType keystore format. {shared.dir} 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 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 9-3 describes the attributes supported by this security provider.

Table 9-3 ALESIdentityCredential Mapper

Attribute Name	Default Value	Description	List
mgmtinterface	"com.bea.security.providers.credentials.alesidentity.ALESIdentityCredentialMapper"		N
TrustedKeystore	"{shared.dir}/keys/demoProviderTrust.jks"	The keystore to be used to get the certificate chain to sign the Identity Assertion. {shared.dir} 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 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 9-4 describes the attributes supported by this security provider.

Table 9-4 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 9-5 describes the attributes supported by this security provider.

Table 9-5 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 9-6 describes the attributes supported by this security provider.

Table 9-6 AsiAuthorizer

Attribute Name	Default Value	Description	List
mgmtinterface	"com.bea.security.providers.authorization.asi.ASIAuthorizer"		N
AdvancedConfigurationProperties		Specifies additional advanced configuration parameters.	Y
Directory	RootOrg!AdminDir	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 9-6 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 9-6 AsiAuthorizer (Continued)

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

ASIRoleMapperProvider

ASIRoleMapperProvider extends weblogic.management.security.authorization.RoleMapper. Table 9-7 describes the attributes supported by this security provider.

Table 9-7 ASIRoleMapperProvider

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

Table 9-7 ASIRoleMapperProvider (Continued)

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

DatabaseAuthenticator

DatabaseAuthenticator extends com.bea.security.providers.authentication.dbms.DBMSAuthenticator.

DatabaseCredentialMapper

DatabaseCredentialMapper extends weblogic.management.security.credentials.CredentialMapper. Table 9-8 describes the attributes supported by this security provider.

Table 9-8 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 9-8 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 9-8 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 9-8 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 9-8 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 9-8 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 9-8 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} username of the requesting user {1} username username of the target user {2} remote username {3} normalized form of the resource {4} normalized form of the action or default if none is defined {5} name of the provider configuration {6} record id being updated (numeric) {7} encrypted password.	N

Table 9-8 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} username of the requesting user {1} username or alias of target user {2} remote username {3} normalized form of the resource {4} normalized form of the action or default if none is defined {5} name of provider configuration {6} encrypted password.	N
SharedSecret		A secret passphrase 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 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 9-9 describes the attributes supported by this security provider.

Table 9-9 DefaultAuthenticator

Attribute Name	Default Value	Description	List
mgmtinterface	weblogic.security.providers.authentication.DefaultAuthenticator		N
MinimumPasswordLength	“8”	Minimum number of characters required in a password.	N
SupportedImportFormats	{“DefaultAtn”}	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		Users and groups to import into this Authentication provider’s database. If none are specified, all are imported.	Y
SupportedExportFormats	{“Default”}	Format of the file to export. The list of supported export formats is determined by this Authentication provider.	Y
SupportedExportConstraints	{“users”,”groups”}	Users and groups to export 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
MaxGroupMembershipSearchLevel	“0”	Specifies how many levels of group membership can be searched. Valid only if GroupMemberShipSearching is set to limited	N
UseRetrievedUserNameAsPrincipal	“false”	Specifies if username retrieved from LDAP should be used as the principal in the subject.	N

DefaultAuthorizer

DefaultAuthorizer extends weblogic.management.security.authorization.DeployableAuthorizer. Table 9-10 describes the attributes supported by this security provider.

Table 9-10 DefaultAuthorizer

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

DefaultCredentialMapper

DefaultCredentialMapper extends weblogic.management.security.credentials.DeployableCredentialMapper. Table 9-11 describes the attributes supported by this security provider.

Table 9-11 DefaultCredentialMapper

Attribute Name	Default Value	Description	List
mgmtinterface	weblogic.security.providers.credentials.DefaultCredentialMapper		N
SupportedImport Formats	{“DefaultCreds”}	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		Credential maps to import into this Credential Mapping provider’s database. If none are specified, all are imported.	Y
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”}	Credential maps to export from this Credential Mapping provider’s database. If none are specified, all are exported.	Y

DefaultRoleMapper

DefaultRoleMapper extends weblogic.management.security.authorization.DeployableRoleMapper. Table 9-12 describes the attributes supported by this security provider.

Table 9-12 DefaultRoleMapper

Attribute Name	Default Value	Description	List
mgmtinterface	weblogic.security.providers.authorization.DefaultRoleMapper		
SupportedImportFormats	{“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.	
SupportedImportConstraints		Security roles that to import into this Role Mapping provider’s database. If none are specified, all are imported.	
SupportedExportFormats	{“DefaultRoles”}	The format of the file to export. The list of supported export formats is determined by this Role Mapping provider.	
SupportedExportConstraints		Security roles to export from this Role Mapping provider’s database. If none are specified, all are exported.	

IPlanetAuthenticator

IPlanetAuthenticator extends com.bea.security.providers.authentication.LDAPAuthenticator. Table 9-13 describes the attributes supported by this security provider.

Table 9-13 IPlanetAuthenticator

Attribute Name	Default Value	Description	List
mgmtinterface	"com.bea.security.providers.authentication.IplanetAuthenticator"		N
GroupFromNameFilter	"(|(&(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
StaticMemberDNAttribute	"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
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 9-14 describes the attributes supported by this security provider.

Table 9-14 LDAPAuthenticator

Attribute Name	Default Value	Description	List
mgmtinterface	com.bea.security.providers.authentication.LDAPAuthenticator		N
UserObjectClass	“person”	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 it 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
UserBaseDN	“ou=people, o=example.com”	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))”	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

Table 9-14 LDAPAuthenticator (Continued)

Attribute Name	Default Value	Description	List
AllUsersFilter		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"	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))	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
AllGroupsFilter		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"	Name of the LDAP object class that stores static groups.	N
StaticGroupNameAttribute	"cn"	Attribute of a static LDAP group object that specifies the name of the group.	N
StaticMemberDNAttribute	"uniquemember"	Attribute of a static LDAP group object that specifies the distinguished names (DNs) of the members of the group.	N

Table 9-14 LDAPAuthenticator (Continued)

Attribute Name	Default Value	Description	List
StaticGroupDNsfromMemberDNFilter	(&(uniquemember=%M)(objectclass=groupofuniquenames))	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		LDAP object class that stores dynamic groups.	N
DynamicGroupNameAttribute		Attribute of a dynamic LDAP group object that specifies the name of the group.	N
DynamicMemberURLAttribute		Attribute of the dynamic LDAP group object that specifies the URLs of the members of the dynamic group.	N
AutomaticFailoverEnabled	“false”	Option to enable automatic failover when using the LDAP server.	N
BackupHost	“localhost”	Host name or IP address of the backup LDAP server.	N
BackupPort	“389”	Port number on which the backup LDAP server is listening.	N
BackupSSLEnabled	“false”	Option to enable SSL when connecting to the backup LDAP server.	N
BackupPrincipal		Distinguished Name (DN) of the LDAP user authorized to connect to the backup LDAP server.	N
BackupCredential		Credential (generally a password) used to authenticate the backup LDAP user defined in the Principal attribute.	N

Table 9-14 LDAPAuthenticator (Continued)

Attribute Name	Default Value	Description	List
PrimaryRetryInterval	“3600”	Number of 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”	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. 0 indicates only direct group memberships will be found. Positive number indicates the number of levels to go down.	N
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 9-15 describes the attributes supported by this security provider.

Table 9-15 Log4jAuditor

Attribute Name	Default Value	Description	List
mgmtinterface	com.bea.security.providers.audit.Log4JAuditor		N
Severity	GLOBAL ERROR INFORMATION SUCCESS WARNING FAILURE	Lowest level at which auditing is initiated. Treated as follows by the Log4j Audit Channel provider. INFORMATION SUCCESS WARNING ERROR FAILURE For example, if log4j severity threshold is ERROR (default setting), all audit events with severity ERROR and FAILURE are audited. Different audit events can be 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 9-15 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, 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 9-15 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. If set to `WITHOUT_CONTEXT`, all events are audited. Add custom AuditEvents using weblogic.security.spi.AuditEvent interface.	
AuditEvent	"WITHOUT_CONTEXT"	Events of type weblogic.security.spi.AuditEvent. If set to `WITHOUT_CONTEXT`, all events are audited.	N
AuditAuthenticationEvent	"WITHOUT_CONTEXT"	Events of type weblogic.security.spi.AuditAtnEvent.	N
AuditAuthorizationEvent	"WITHOUT_CONTEXT"	Events of type weblogic.security.spi.AuditAtzEvent.	N
AuditRoleEvent	"WITHOUT_CONTEXT"	Events of type weblogic.security.spi.AuditRoleEvent.	N
AuditProviderRecordEvent	"WITHOUT_CONTEXT"	Events of type com.bea.security.spi.ProviderAuditRecord	N
AuditManagementEvent	"WITHOUT_CONTEXT"	Events of type weblogic.security.spi.AuditMgmtEvent	N

Table 9-15 Log4jAuditor (Continued)

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

NovellAuthenticator

NovellAuthenticator extends com.bea.security.providers.authentication.LDAPAuthenticator. Table 9-16 describes the attributes supported by this security provider.

Table 9-16 NovellAuthenticator

Attribute Name	Default Value	Description	List
mgmtinterface	"com.bea.security.providers.authentication.NovellAuthenticator"		N
UserNameAttribute	"cn"	Attribute of an LDAP user object that specifies the name of the user.	N
UserFromNameFilter	"(&(cn=%u)(objectclass=person))	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))	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 9-16 NovellAuthenticator

Attribute Name	Default Value	Description	List
StaticGroupObjectClass	“groupofnames”	Name of the LDAP object class that stores static groups.	N
StaticGroupDNsfromMemberDNFilter	“(&(uniquemember=%M)(objectclass=groupofnames))	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 9-17 describes the attributes supported by this security provider.

Table 9-17 NTAuthenticator

Attribute Name	Default Value	Description	List
mgmtinterface	"weblogic.security.providers.authentication.NTAuthenticator"		N
DomainControllers	"localanddomain"	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]"}	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 9-17 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. **BADDCRetryAlwaysString** indicates 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. Used only when BadDomainControllerRetry setting is configured to use delay (BADDCRetryDelayString). 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 9-17 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 9-17 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** Windows NT domain name is never inserted into the principal name. **MAP NTDomain Name UPNString** Windows NT domain name is inserted into the principal name using the style domain\\name. **MAP NTDomain Name Never String** 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 9-18 describes the attributes supported by this security provider.

Table 9-18 OpenLDAPAuthenticator

Attribute Name	Default Value	Description	List
mgmtinterface	com.bea.security.providers.authentication.OpenLDAPAuthenticator		N
UserNameAttribute	“cn”	Attribute of an LDAP user object that specifies the name of the user.	N
UserBaseDN	“ou=people, dc=example, dc=com”	Base distinguished name (DN) of the tree in the LDAP directory that contains users.	N
UserFromNameFilter	“((cn=%u)(objectclass=person))”	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”	Base distinguished name (DN) of the tree in the LDAP directory that contains groups.	N
GroupFromNameFilter	“((cn=%g)(objectclass=groupofnames))”	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”	Name of the LDAP object class that stores static groups.	N

Table 9-18 OpenLDAPAuthenticator (Continued)

Attribute Name	Default Value	Description	List
StaticMemberDNAttribute	"member"	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))"	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 9-19 describes the attributes supported by this security provider.

Table 9-19 PerfDBAuditor

Attribute Name	Default Value	Description	List
mgmtinterface	com.bea.security.providers.audit.PerfDBAuditor		N
PerformanceStatisticsInterval	5	Performance statistics gathering interval, in minutes.	
PerformanceStatisticsDuration	0	Length of circular buffer in minutes . Must be greater than or equal to the interval. A value of 0 means unlimited duration.	
EnablePerformanceStatistics	true	Enables/disables performance-gathering counters.	
JDBCDriverClassName	oracle.jdbc.driver.OracleDriver	Java class name of the JDBC Driver.	
JDBCConnectionURL		The connection string for the authentication database.	

Table 9-19 PerfDBAuditor

Attribute Name	Default Value	Description	List
DatabaseUserLogin		User id to access the authentication database.	
DatabaseUserPassword		User password to access the authentication database.	
JDBCConnectionProperties		Optional parameters for configuring the JDBC Connection. Legal values are determined by the JDBC Driver. These properties are entered as NAME=VALUE.	
AuthenticationStatisticsTable	PERF_ATH_STAT	Database table for collecting authentication statistics.	
AuthorizationStatisticsTable	PERF_ATZ_STAT	Database table for collecting authorization statistics.	
AuthorizationAttrStaticsTable	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 9-20 describes the attributes supported by this security provider.

Table 9-20 ResourceDeploymentAuditor

Attribute Name	Default Value	Description	List
mgmtinterface	com.bea.security.providers.audit.ResourceDeploymentAuditor		N
ResourceDeploymentEnabled	“true”	If “true” the audit provider publishes resources to the Administration Application.	N

Table 9-20 ResourceDeploymentAuditor

Attribute Name	Default Value	Description	List
ResourceDeploymentNamingAuthority	"RESOURCEDEPLOYMENT"	Naming authority of audit events to process as resource deployment audit events.	N
SessionEvictionCapacity	"40"	Number of sessions to actively maintain. Once the limit is reached, old sessions are dropped and automatically reestablished when needed.	N
SessionEvictionPercentage	"25"	Percentage of the sessions to drop when the eviction capacity is reached.	N
SessionLifetime	"120000"	MJaximum number of milliseconds a session can use before it is discarded. A value of 0 indicates that sessions are indefinite.	N
SessionMaxUses	"100"	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"	Root of the resource tree where new resources are published.	N
SharedResourcesParent	"shared"	Root of the resource tree where new shared resources are published. This item may be relative to the value specified by ApplicationDeploymentParent	N
ResourceConverters		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. OES includes resource converters for the standard WebLogic Server resource types.	Y

Table 9-20 ResourceDeploymentAuditor

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

SAMLCredentialMapper

SAMLCredentialMapper extends weblogic.management.security.credentials.CredentialMapper. Table 9-21 describes the attributes supported by this security provider.

Table 9-21 SAMLCredentialMapper

Attribute Name	Default Value	Description	List
mgmtinterface	"com.bea.security.providers.credentials.saml.SAMLCredentialMapper"		N
TrustedKeystore	"{shared.dir}/keys/demoProviderTrust.jks"	Keystore used to get the Certificate chain to sign the SAML Assertion with. {shared.dir} 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"	TYPE of keystore that is specified in TrustedKeystore.	N
TrustedCertAlias	"demo_provider_trust"	Cert alias to be used to sign the SAML Assertion.	N
TrustedCertAliasPasswd	"password"	Password to use for the CertAlias specified to retrieve the private key from the keystore.	N
NotBeforeOffset	"120"	Number of seconds in the past to make an assertion valid to allow for clock skew.	N
NotAfterOffset	"300"	Number of seconds in the future to make an assertion valid.	N

Table 9-21 SAMLCredentialMapper (Continued)

Attribute Name	Default Value	Description	List
IssuerURI	“https://www.bea.com”	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 9-22 describes the attributes supported by this security provider.

Table 9-22 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 9-22 SAMLIdentityAsserter (Continued)

Attribute Name	Default Value	Description	List
TrustedCAKeystore	"{shared.dir}/keys/demoProviderTrust.jks"	Location of the Trusted Keystore stored in the TrustedCAKeystoreType keystore format. {shared.dir} 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	{shared.dir}/keys/demoProviderTrust.jks"	Location of the Trusted Keystore stored in the TrustedKeystoreType keystore format. {shared.dir} 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 9-22 SAMLIdentityAsserter (Continued)

Attribute Name	Default Value	Description	List
TrustedCAKeystoreType	“JKS”	Type of keystore to which the trustedCAKeystore is configured.	N
TrustedKeystoreType	“JKS”	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 9-23 describes the attributes supported by this security provider.

Table 9-23 SinglePassNegotiateIdentityAsserter

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

X509IdentityAsserter

X509IdentityAsserter extends weblogic.management.security.authentication.IdentityAsserter. Table 9-24 describes the attributes supported by this security provider.

Table 9-24 X509IdentityAsserter

Attribute Name	Default Value	Description	List
mgmtInterface	"com.bea.security.providers.authentication.X509IdentityAsserter"		N
SupportedTypes	AuthenticatedUser X.509 CSI.PrincipalName CSI.ITTAnonymous CSI.X509CertChain CSI.DistinguishedName	Token types supported by the Identity Assertion provider.	Y
UserNameMapperClassName		Name of the Java class that maps X.509 digital certificates and X.501 distinguished names to AquaLogic Enterprise Security user names.	N
TrustedClientPrincipals		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"	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 9-25 describes the attributes supported by this security provider.

Table 9-25 XACMLAuthorizer

Attribute Name	Default Value	Description	List
mgmtInterface	com.bea.security.providers.authorization.xacml.XACMLAuthorizer		N
PolicyDirectory	"xacmlpolicy"	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		XACML policy that is provisioned to the XACML authorization provider via the SCM.	Y