



# **Setup and Maintenance Guide**

ForgeRock Access Management 5

ForgeRock AS  
201 Mission St, Suite 2900  
San Francisco, CA 94105, USA  
+1 415-599-1100 (US)  
[www.forgerock.com](http://www.forgerock.com)

---

Copyright © 2011-2017 ForgeRock AS.

## Abstract

Guide to performing setup and maintenance tasks in ForgeRock# Access Management, such as backing up and restoring, managing keystores, tuning the environment, monitoring, and others. ForgeRock Access Management provides authentication, authorization, entitlement, and federation software.



This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

To view a copy of this license, visit <https://creativecommons.org/licenses/by-nc-nd/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

ForgeRock® and ForgeRock Identity Platform™ are trademarks of ForgeRock Inc. or its subsidiaries in the U.S. and in other countries. Trademarks are the property of their respective owners.

UNLESS OTHERWISE MUTUALLY AGREED BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

DejaVu Fonts

Bitstream Vera Fonts Copyright

Copyright (c) 2003 by Bitstream, Inc. All Rights Reserved. Bitstream Vera is a trademark of Bitstream, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Bitstream" or the word "Vera".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Bitstream Vera" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL BITSTREAM OR THE GNOME FOUNDATION BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the names of Gnome, the Gnome Foundation, and Bitstream Inc., shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from the Gnome Foundation or Bitstream Inc., respectively. For further information, contact: [fonts at gnome dot org](mailto:fonts at gnome dot org).

Arev Fonts Copyright

Copyright (c) 2006 by Tavmjong Bah. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the modifications to the Bitstream Vera Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Tavmjong Bah" or the word "Arev".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Tavmjong Bah Arev" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL TAVMJONG BAH BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the name of Tavmjong Bah shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from Tavmjong Bah. For further information, contact: [tavmjong @ free . fr](mailto:tavmjong @ free . fr).

Admonition graphics by Yannick Lung. Free for commercial use. Available at [FreeCnS.Cumulus](http://FreeCnS.Cumulus).

---

# Table of Contents

Preface .....	v
1. Introducing Administration Interfaces and Tools .....	1
1.1. Web-Based AM Console .....	1
1.2. Command-Line Tools .....	6
2. Setting Up Realms .....	9
2.1. Introducing Realms .....	9
2.2. Implementing Realms Using the AM Console .....	10
2.3. Implementing Realms using the REST API .....	13
2.4. Customizing Realms .....	34
3. Setting Up Identity Data Stores .....	38
3.1. Introducing Identity Data Stores .....	38
3.2. Implementing Identity Data Stores .....	38
3.3. Customizing Identity Data Stores .....	40
4. Setting Up Policy Agent Profiles .....	49
4.1. OpenIG or Policy Agent? .....	49
4.2. Types of Agent .....	49
4.3. Creating Agent Profiles .....	50
4.4. Delegating Agent Profile Creation .....	53
4.5. Configuring Web Policy Agent Properties .....	54
4.6. Configuring Java EE Policy Agents .....	54
4.7. Configuring Version 2.2 Policy Agents .....	54
4.8. OAuth 2.0 and OpenID Connect 1.0 Client Settings .....	55
4.9. Configuring Agent Authenticators .....	63
4.10. Configuring SOAP STS Agents .....	63
5. Setting Up Keys and Keystores .....	64
5.1. Introducing Keystores .....	64
5.2. Configuring Keystores .....	67
5.3. Configuring Key Aliases .....	70
5.4. Configuring Password String Aliases .....	77
6. Setting Up Audit Logging .....	80
6.1. Introducing the Audit Logging Service .....	80
6.2. Implementing the Audit Logging Service .....	82
6.3. Implementing the Classic Logging Service .....	105
7. Setting Up the Dashboard Service .....	108
7.1. Introducing the Dashboard Service .....	108
7.2. Implementing the Dashboard Service .....	110
8. Maintaining an Instance .....	115
8.1. Backing Up and Restoring Configurations .....	115
8.2. Changing the amadmin User's Password .....	120
8.3. Monitoring Services .....	122
8.4. Tuning an Instance .....	130
8.5. Managing Sessions .....	141
8.6. Changing Host Names .....	142
9. Troubleshooting .....	145

9.1. Is the Instance Running? .....	145
9.2. Debug Logging .....	145
9.3. Recording Troubleshooting Information .....	147
10. Reference .....	156
10.1. Data Store Configuration Properties .....	156
10.2. Realm Privileges Configuration Reference .....	207
10.3. Record Control File Configuration Properties .....	208
10.4. Audit Logging File Format .....	211
10.5. Audit Logging .....	219
10.6. Logging .....	240
10.7. Monitoring .....	248
10.8. OAuth2 Provider .....	250
10.9. Dashboard .....	262
10.10. Command-line Tool Reference .....	263
A. About the REST API .....	265
A.1. Introducing REST .....	265
A.2. About ForgeRock Common REST .....	265
A.3. REST API Versioning .....	282
A.4. Specifying Realms in REST API Calls .....	287
A.5. Authentication and Logout .....	288
A.6. Using the Session Token After Authentication .....	294
A.7. Server Information .....	295
A.8. Token Encoding .....	296
A.9. Logging .....	296
A.10. Reference .....	298
B. Getting Support .....	301
B.1. Accessing Documentation Online .....	301
B.2. Joining the ForgeRock Community .....	302
B.3. Getting Support and Contacting ForgeRock .....	302
Glossary .....	303

# Preface

This guide covers how to set up and maintain core functionality such as realms, data stores and auditing and others, and also how to perform maintenance tasks in OpenAM such as backing up and restoring, tuning, and others.

This guide is written for anyone that sets up and maintains ForgeRock Access Management services for their organizations. This guide covers tasks and configurations you might repeat throughout the life cycle of a deployment in your organization.

## About ForgeRock Identity Platform™ Software

ForgeRock Identity Platform™ is the only offering for access management, identity management, user-managed access, directory services, and an identity gateway, designed and built as a single, unified platform.

The platform includes the following components that extend what is available in open source projects to provide fully featured, enterprise-ready software:

- ForgeRock Access Management (AM)
- ForgeRock Identity Management (IDM)
- ForgeRock Directory Services (DS)
- ForgeRock Identity Gateway (IG)

## Chapter 1

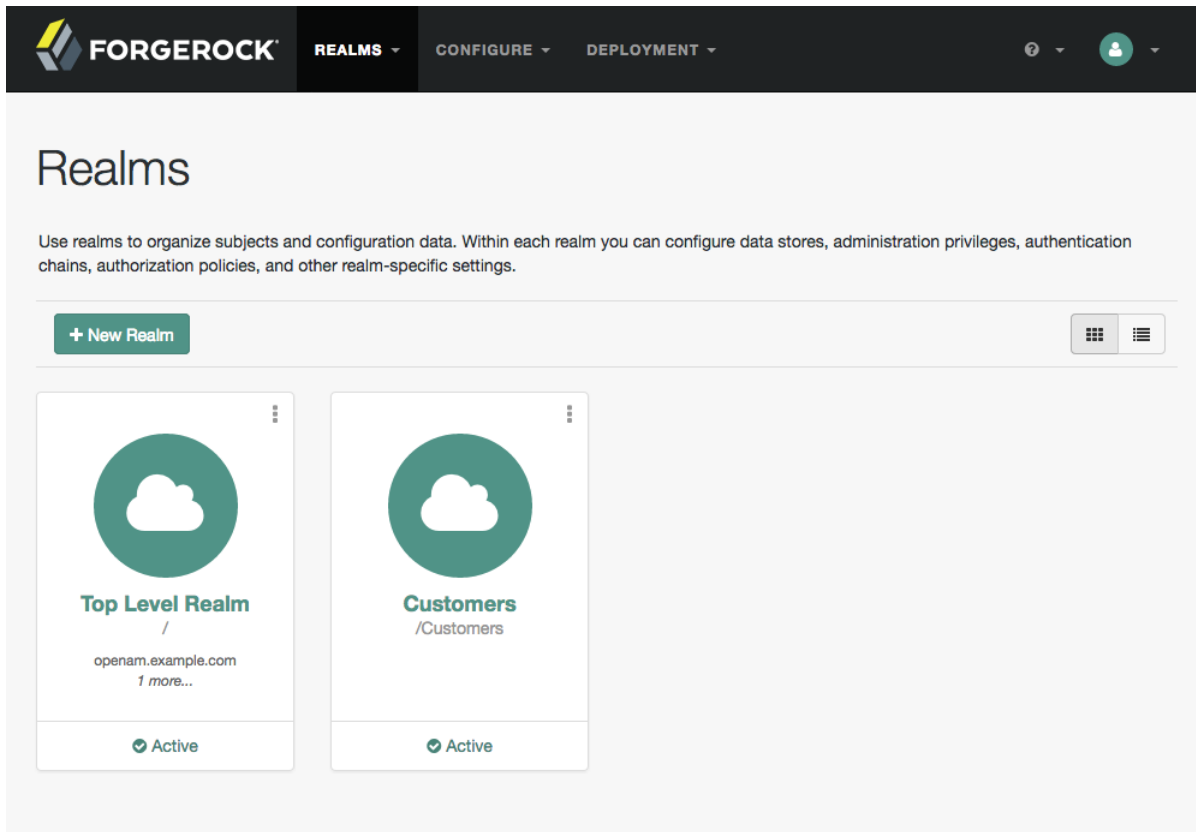
# Introducing Administration Interfaces and Tools

This chapter provides a brief introduction to the web-based UI console. It also lists and describes each command-line interface (CLI) administration tool.

## 1.1. Web-Based AM Console

After you install OpenAM, log in to the web-based AM console as OpenAM administrator, `amadmin` with the password you set during installation. Navigate to a URL, such as `http://openam.example.com:8080/openam`. In this case, communications proceed over the HTTP protocol to a FQDN (`openam.example.com`), over a standard Java EE web container port number (8080), to a specific deployment URI (`/openam`).

Figure 1.1. The AM Console



When you log in as the OpenAM administrator, `amadmin`, you have access to the complete AM console. In addition, OpenAM has set a cookie in your browser that lasts until the session expires, you logout, or you close your browser.<sup>1</sup>

When you log in to the AM console as a non-administrative end user, you do not have access to the administrative console. Your access is limited to self-service profile pages and user dashboard.

<sup>1</sup>Persistent cookies can remain valid when you close your browser. This section reflects OpenAM default behavior before you configure additional functionality.

*Figure 1.2. The AM Console for Non-Administrative Users*

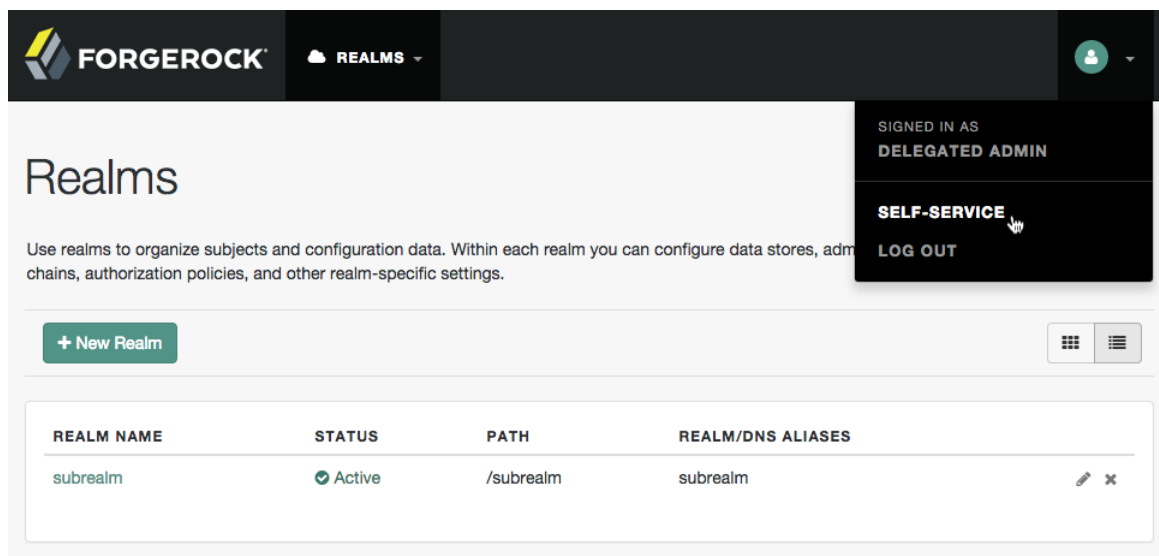
The screenshot shows the 'User profile' page in the ForgeRock AM Console. The page has a dark header with the ForgeRock logo, 'DASHBOARD', 'SHARES', and a user profile icon. The main content area is titled 'User profile' and contains two tabs: 'Basic Info' (selected) and 'Password'. The 'Basic Info' tab displays a form with the following fields: Username (scarter), First Name (Sam), Last Name (Carter), Email address (scarter@example.com), and Phone number (+1 408 555 4798). At the bottom right of the form are 'Reset' and 'Update' buttons.

Field	Value
Username	scarter
First Name	Sam
Last Name	Carter
Email address	scarter@example.com
Phone number	+1 408 555 4798

If you configure OpenAM to grant administrative capabilities to another user, then that user is able to access both the administration console in the realms they can administrate, and their self-service profile pages.



Figure 1.3. The AM Console for a Delegated Administrator



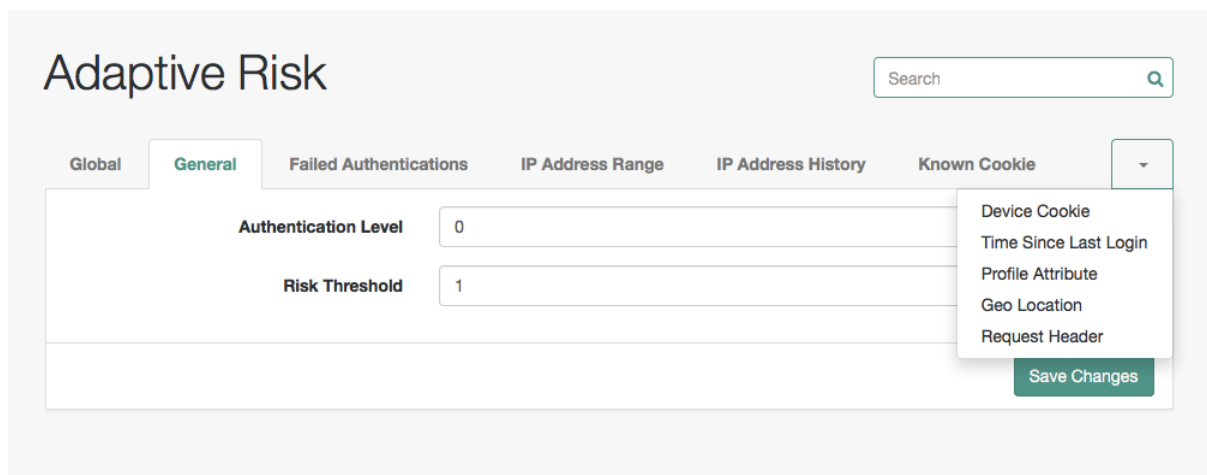
For more on delegated administration, see Section 2.4.1, "Delegating Realm Administration Privileges".

### 1.1.1. AM Console Responsiveness

The web-based console is a responsive website, which means it would resize some of its features to fit the size of your screen and the layout design.

For example, the header menu would change into a dropdown menu, and those pages with many tabs would shed most of them for a dropdown menu to the left-hand side.

Figure 1.4. AM Console Responsiveness



**Adaptive Risk**

Search

Global **General** Failed Authentications IP Address Range IP Address History Known Cookie

Authentication Level 0

Risk Threshold 1

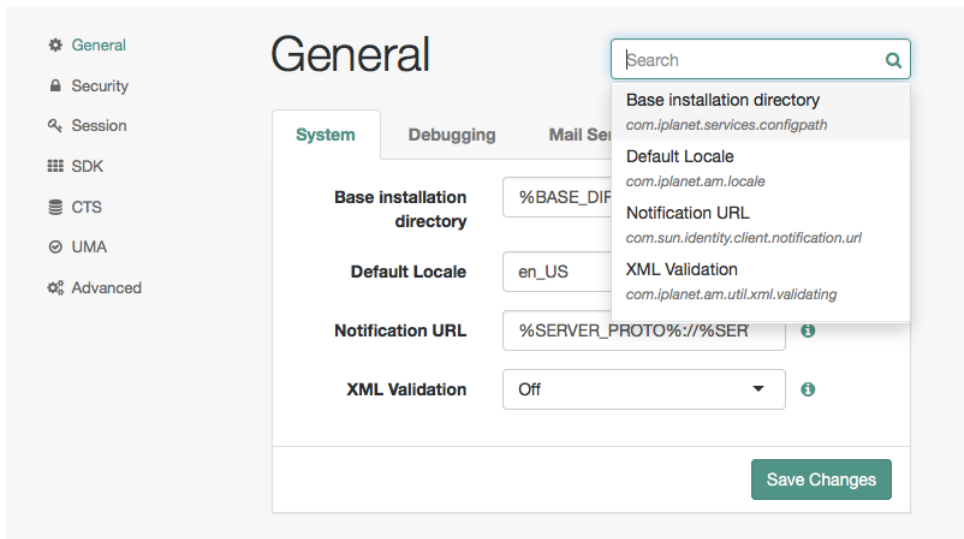
- Device Cookie
- Time Since Last Login
- Profile Attribute
- Geo Location
- Request Header

Save Changes

### 1.1.2. The AM Console Search Feature

Use the search box to find any configuration attribute on the section you are in. It can autocomplete the word you are typing, or you can click on the box and display the list of available attributes for you.

Figure 1.5. The AM Console Search Feature



## 1.2. Command-Line Tools

The script tools in the following list have **.bat** versions for use on Microsoft Windows.

You can install the following command-line tools:

### **agentadmin**

This tool lets you manage policy agent installations.

Unpack this tool as part of policy agent installation.

### **ampassword**

This tool lets you change Administrator passwords, and display encrypted password values.

Install this from the **SSOAdminTools-14.0.0.zip**.

### **amverifyarchive**

This tool checks log archives for tampering.

Install this from **SSOAdminTools-14.0.0.zip**.

### **openam-distribution-configurator-14.0.0.jar**

This executable **.jar** file lets you perform a silent installation of an OpenAM server with a configuration file. For example, the **java -jar configurator.jar -f config.file** command couples the

`configurator.jar` archive with the `config.file`. The `sampleconfiguration` file provided with the tool is set up with the format for the `config.file`, and it must be adapted for your environment.

Install this from `SSOConfiguratorTools-14.0.0.zip`.

## ssoadm

This tool provides a rich command-line interface for the configuration of core services.

Install this from `SSOAdminTools-14.0.0.zip`.

To translate settings applied in the AM console to service attributes for use with **ssoadm**, log in to the AM console as `amadmin` and access the services page, in a URL, such as `http://openam.example.com:8080/openam/services.jsp`.

The commands access the OpenAM configuration over HTTP (or HTTPS). When using the administration commands in a site configuration, the commands access the configuration through the front end load balancer.

Sometimes a command cannot access the load balancer because:

- Network routing restrictions prevent the tool from accessing the load balancer.
- For testing purposes, the load balancer uses a self-signed certificate for HTTPS, and the tool does not have a way of trusting the self-signed certificate.
- The load balancer is temporarily unavailable.

In such cases you can work around the problem by adding an option for each node, such as the following to the **java** command in the tool's script.

Node 1:

```
-D"com.iplanet.am.naming.map.site.to.server=https://lb.example.com:443/openam=  
http://server1.example.com:8080/openam"
```

Node 2:

```
-D"com.iplanet.am.naming.map.site.to.server=https://lb.example.com:443/openam=  
http://server2.example.com:8080/openam"
```

In the above example the load balancer is on the `lb` host, `https://lb.example.com:443/openam` is the site name, and the OpenAM servers in the site are on `server1` and `server2`.

The **ssoadm** command will only use the latest value in the map, so if you have a mapping like:

```
-D"com.iplanet.am.naming.map.site.to.server=https://lb.example.com:443/openam=  
http://server1.example.com:8080/openam, https://lb.example.com:443/openam=  
http://server2.example.com:8080/openam"
```

The **ssoadm** command will always talk to:

```
http://server2.example.com:8080/openam
```

## Chapter 2

# Setting Up Realms

This chapter explains what realms are and how to configure and customize them.

## 2.1. Introducing Realms

OpenAM *realms*, are used to group configuration and identities together. For example, you might have one realm for OpenAM administrators and agents, and another realm for users. In this two-realm setup, the OpenAM administrator can log in to the administrative realm to manage the services, but cannot authenticate as OpenAM administrator to the realm that protects web sites with HR and financial information.

OpenAM associates a realm with at least one identity repository and authentication chain. OpenAM also associates the realm with authorization applications and their policies, and with privileges for administrators. Each realm can have its own configuration for the services it provides.

When you first configure OpenAM, OpenAM sets up the default Top Level Realm, sometimes referred to as the `/` realm or root realm. The Top Level Realm contains OpenAM configuration data and allows authentication using the identity repository that you choose during initial configuration. The Top Level Realm might hold the overall configuration for Example.com, for instance.

You create new realms to subdivide authentication and authorization, and to delegate management of subrealms. For example, your organization might require separate realms for payroll, human resources, and IT management domains and their applications.

By default, a new realm inherits configuration from its parent's configuration. The default identity repository is the one you choose when you deploy and configure OpenAM. The default authentication mechanism corresponds to that identity repository as well. You can, however, constrain authentication to rely on different data stores, and set policy for agents to define authorization in the realm.

### Tip

Keep administration of access management services separate from management of the services themselves:

- Create realms for your organization(s) and separate administrative users from end users. You must then either:
  - Use the `realm=realm-name` query string parameter when redirecting users to OpenAM, which gives you a way to isolate the URLs used by an application:

- Create fully-qualified domain name DNS aliases, and use them to control access to the realms.

## 2.2. Implementing Realms Using the AM Console

You create and configure realms through the AM console, starting from the Realms page.

### Note

AM requires cookies for all configured realms when using DNS aliases. For example, if you install AM in the domain, `openam.example.net` and have realms, `identity.example.org` and `security.example.com` then you must configure cookie domains for `.example.net`, `.example.org`, and `.example.com`. You can set up the cookie domains for each realm by following the procedure in Procedure 2.2, "To Configure DNS Aliases for Accessing a Realm".

This section has the following procedures:

- Procedure 2.1, "To Create a New Realm"
- Procedure 2.2, "To Configure DNS Aliases for Accessing a Realm"

### Procedure 2.1. To Create a New Realm

You can create a new realm through the AM console as described below, or by using the **ssoadm create-realm** command:

1. Log in to the AM console as OpenAM Administrator, `amadmin`.
2. On the Realms page, click New Realm. The New Realm page appears. Complete the form to configure the realm.
3. In the Name field, enter a descriptive name for the realm.

### Note

Realm names must not match any of the following:

- Existing realm names.
- Existing realm aliases.
- Names of OpenAM REST endpoints.

For example `users`, `groups`, `realms`, `policies` or `applications`.

4. The Active button is enabled by default.

**Warning**

If you configure the realm to be inactive, then users cannot use it to authenticate or be granted access to protected resources.

5. In the Parent field, enter the parent of your realm.

Default: the top level realm (/).

6. In the Realm Aliases field, enter a simple text alias to represent the realm.
7. In the DNS Aliases field, enter fully-qualified domain names (FQDN) that can be used to represent the realm.

A DNS alias is not related to the CNAME record used in DNS database zones. In other words, the option shown in the AM console does not conform to the definition of DNS aliases described in RFC 2219.

**Tip**

Entering a DNS alias in the AM console also applies required changes to the advanced server property `com.sun.identity.server.fqdnMap`.

For more information, see Procedure 2.2, "To Configure DNS Aliases for Accessing a Realm".

8. To enable stateless sessions for the realm, toggle the Use Stateless Sessions switch. For more information on sessions, see Section 1.8, "About Sessions" in the *Authentication and Single Sign-On Guide*.
9. Click Create to save your configuration.

### *Procedure 2.2. To Configure DNS Aliases for Accessing a Realm*

You can configure realms to be associated with specific fully-qualified domain names (FQDN).

For example, consider a deployment with the following characteristics:

- The FQDN for OpenAM and the top level realm is `openam.example.com`.
- OpenAM also services `realm1.example.com`, and `realm2.example.com`. In other words, OpenAM receives all HTTP(S) connections for these host names. Perhaps they share an IP address, or OpenAM listens on all interfaces.

Without applying DNS aliases to the relevant realm, when a user visits `http://realm1.example.com:8080/openam`, OpenAM redirects that user to the top level realm, `http://openam.example.com:8080/openam`. If the authenticating user is present only in `realm1`, then authentication fails even with correct credentials.



If no DNS alias is configured for a realm, `realm1` users must visit URLs such as `http://openam.example.com:8080/openam/XUI/?realm=/realm1#login`. This format of URL reveals the top level realm, and exposes extra information about the service.

Configure DNS aliases for realms to prevent redirection and having to expose the top level realm domain by performing the following steps.

#### Note

Realm aliases must be unique within an OpenAM instance, and cannot contain the characters `"`, `#`, `$`, `%`, `&`, `+`, `,`, `/`, `:`, `;`, `<`, `=`, `>`, `?`, `@`, `\`, or spaces.

1. Add the domains that OpenAM services to the list of domains that created cookies will be applicable to, as follows:
  - a. Log in to the AM console as an OpenAM administrator, for example, `amadmin`.
  - b. Navigate to Configure > Global Services, select the System tab, and then click Platform.
  - c. In Cookie Domains, enter the domains that OpenAM will service.  
  
For example, if you installed OpenAM at `openam.example.net`, and intend to have realms associated with FQDNs `realm1.example.org` and `realm2.example.com`, then the Cookie Domains list would include `example.net`, `example.org`, and `example.com`.
2. Set the FQDN for each realm as follows:
  - a. Navigate to Realms > *Realm Name*, and then click Properties.
  - b. In DNS Aliases, enter one or more FQDN values for the realm.
  - c. Save your changes.
3. Adding DNS aliases by using the AM console also adds FQDN mappings to the OpenAM server.

To verify these have been created perform the following steps:

- a. Navigate to Deployment > Servers > *Server Name* > Advanced.
- b. For each FQDN DNS alias configured, verify the existence of a property named `com.sun.identity.server.fqdnMap[Realm FQDN]` with a property value of *Realm FQDN*.

For example, the property may be called `com.sun.identity.server.fqdnMap[realm1.example.com]` with a value of `realm1.example.com`.

If the property does not exist or needs to be changed, manually create the property for each FQDN DNS alias.

- c. Save your changes.

The new realm aliases take effect immediately, it is not necessary to restart OpenAM. You can now use a URL such as `http://realm1.example.com:8080/openam` to access `realm1`, rather than `http://openam.example.com:8080/openam/XUI/?realm=/realm1#login`.

## 2.3. Implementing Realms using the REST API

This section shows how to use the OpenAM RESTful interfaces for identity and realm management.

In this section, long URLs are wrapped to fit the printed page, as some of the output is formatted for easier reading.

Before making a REST API call to manage an identity, make sure that you have:

- Authenticated successfully to OpenAM as a user with sufficient privileges to make the REST API call
- Obtained the session token returned after successful authentication

When making the REST API call, pass the session token in the HTTP header. For more information about the OpenAM session token and its use in REST API calls, see Section A.6, "Using the Session Token After Authentication". For general information about the REST API, see Appendix A, "About the REST API".

### 2.3.1. Identity Management

This section shows how to create, read, update, delete, and list identities using the RESTful APIs.

#### Important

OpenAM is not primarily an identity data store, nor is it provisioning software. For storing identity data, consider OpenDJ. For provisioning, consider OpenIDM. Both of these products provide REST APIs as well.

OpenAM has the `/json/agents`, `/json/groups`, and `/json/users` JSON-based APIs for managing identities. These APIs follow the ForgeRock common REST pattern for creating, reading, updating, deleting, and querying resources.

Examples in this section do not repeat the authentication shown in Section A.5, "Authentication and Logout". For browser-based clients, you can rely on OpenAM cookies rather than construct the header in your application. Managing agent profiles, groups, and users with these APIs requires authentication. The examples shown in this section were performed with the token ID gained after authenticating as an OpenAM administrator, for example `amAdmin`.

Although the examples here show user management, you can use `/json/agents` and `/json/groups` in similar fashion. See Section 2.3.2, "Realm Management" for examples related to realms.

The following sections cover this JSON-based API:

- Section 2.3.1.1, "Creating Identities"
- Section 2.3.1.2, "Reading Identities"
- Section 2.3.1.3, "Updating Identities"
- Section 2.3.1.4, "Deleting Identities"
- Section 2.3.1.5, "Listing Identities"
- Section 2.3.1.7, "Changing Passwords"

### 2.3.1.1. Creating Identities

OpenAM lets administrators create a user profile by making an HTTP POST of the JSON representation of the profile to `/json/subrealm/users/?_action=create`. To add a user to the Top Level Realm, you do not need to specify the realm.

The following example shows an administrator creating a new user. The only required fields are `username` and `userpassword`. If no other name is provided, the entry you make for `username` defaults to both the user id and the user's last name:

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
--data \
'{
  "username": "bjensen",
  "userpassword": "secret12",
  "mail": "bjensen@example.com"
}' \
https://openam.example.com:8443/openam/json/realms/root/users/?_action=create
{
  "_id": "bjensen",
  "_rev": "1092918444",
  "username": "bjensen",
  "realm": "/",
  "uid": [
    "bjensen"
  ],
  "mail": [
    "bjensen@example.com"
  ],
  "universalid": [
    "id=bjensen,ou=user,dc=openam,dc=forgerock,dc=org"
  ],
  "objectClass": [
    "iplanet-am-managed-person",
    "inetuser",
    "sunFederationManagerDataStore",
```

```

"sunFMSAML2NameIdentifier",
"devicePrintProfilesContainer",
"inetorgperson",
"sunIdentityServerLibertyPPService",
"iPlanetPreferences",
"pushDeviceProfilesContainer",
"iplanet-am-user-service",
"forgerock-am-dashboard-service",
"organizationalperson",
"top",
"kbaInfoContainer",
"sunAMAuthAccountLockout",
"person",
"oathDeviceProfilesContainer",
"iplanet-am-auth-configuration-service"
],
"inetUserStatus": [
  "Active"
],
"dn": [
  "uid=bjensen,ou=people,dc=openam,dc=forgerock,dc=org"
],
"sn": [
  "bjensen"
],
"cn": [
  "bjensen"
],
"createTimestamp": [
  "20170110103112Z"
]
}

```

Alternatively, administrators can create user profiles with specific user IDs by doing an HTTP PUT of the JSON representation of the changes to `/json/users/user-id`, as shown in the following example:

```

$ curl \
  --request PUT \
  --header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
  --header "Content-Type: application/json" \
  --header "If-None-Match: *" \
  --data \
  '{
    "username": "janedoe",
    "userpassword": "secret12",
    "mail": "janedoe@example.com"
  }' \
  https://openam.example.com:8443/openam/json/realms/root/users/janedoe
{
  "_id": "janedoe",
  "_rev": "-3822252",
  "username": "janedoe",
  "realm": "/",
  "uid": [
    "janedoe"
  ],
  "mail": [
    "janedoe@example.com"
  ],
}

```

```
"universalid": [
  "id=janedoe,ou=user,dc=openam,dc=forgerock,dc=org"
],
"objectClass": [
  "iplanet-am-managed-person",
  "inetuser",
  "sunFederationManagerDataStore",
  "sunFMSAML2NameIdentifier",
  "devicePrintProfilesContainer",
  "inetorgperson",
  "sunIdentityServerLibertyPPService",
  "iPlanetPreferences",
  "pushDeviceProfilesContainer",
  "iplanet-am-user-service",
  "forgerock-am-dashboard-service",
  "organizationalperson",
  "top",
  "kbaInfoContainer",
  "sunAMAuthAccountLockout",
  "person",
  "oathDeviceProfilesContainer",
  "iplanet-am-auth-configuration-service"
],
"dn": [
  "uid=janedoe,ou=people,dc=openam,dc=forgerock,dc=org"
],
"inetUserStatus": [
  "Active"
],
"sn": [
  "janedoe"
],
"cn": [
  "janedoe"
],
"createTimestamp": [
  "20170110103319Z"
]
}
```

As shown in the examples, OpenAM returns the JSON representation of the profile on successful creation. On failure, OpenAM returns a JSON representation of the error including the HTTP status code. For example, version 2.0 of the CREST [/json/users](#), [/json/groups](#), and [/json/agents](#) endpoints return 403 if the user making the request is not authorized to do so.

The same HTTP POST and PUT mechanisms also work for other objects such as policy agent profiles and groups:

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
--data \
'{
  "username": "myAgent",
  "com.sun.identity.agents.config.fqdn.default":
    ["www.example.com"],
}
```

```
"com.sun.identity.agents.config.repository.location":
  ["centralized"],
"agenttype":["WebAgent"],
"serverurl":["https://openam.example.com:8443/openam/"],
"agenturl":["http://www.example.com:80/"],
"userpassword":["password"],
"com.sun.identity.agents.config.login.url":
  ["[0]=https://openam.example.com:8443/openam/XUI/?realm=#login"],
"com.sun.identity.agents.config.logout.url":
  ["[0]=https://openam.example.com:8443/openam/XUI/?realm=#logout"],
"sunidentityserverdevicestatus":["Active"]
}' \
https://openam.example.com:8443/openam/json/realms/root/agents/?_action=create
{
  "username": "myAgent",
  "realm": "/",
  "com.sun.identity.agents.config.fqdn.default": [
    "www.example.com"
  ],
  "com.sun.identity.agents.config.repository.location": [
    "centralized"
  ],
  "AgentType": [
    "WebAgent"
  ],
  "userpassword": [
    "{SHA-1}W6ph5Mm5Pz8GgiULbPgZG37mj9g="
  ],
  "com.sun.identity.agents.config.login.url": [
    "[0]=https://openam.example.com:8443/openam/XUI/?realm=#login"
  ],
  "com.sun.identity.agents.config.logout.url":
    ["[0]=https://openam.example.com:8443/openam/XUI/?realm=#logout"
  ],
  "sunIdentityServerDeviceStatus": [
    "Active"
  ]
}
```

## Note

The command output above has been truncated to be more readable. When you create a policy agent profile, OpenAM returns the full profile in JSON format.

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
--data '{
  "username": "newGroup",
  "uniquemember": ["uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"]
}' \
https://openam.example.com:8443/openam/json/realms/root/groups?_action=create
{
  "username": "newGroup",
  "realm": "/",
```

```

"uniqueMember": [
  "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
],
"cn": [
  "newGroup"
],
"dn": [
  "cn=newGroup,ou=groups,dc=openam,dc=forgerock,dc=org"
],
"objectclass": [
  "groupofuniquenames",
  "top"
],
"universalid": [
  "id=newGroup,ou=group,dc=openam,dc=forgerock,dc=org"
]
}

$ curl \
--request PUT \
--header "If-None-Match: *" \
--header "iPlanetDirectoryPro: AQIC5w...2NzEz*" \
--header "Content-Type: application/json" \
--data '{
  "username": "anotherGroup",
  "uniqueMember": ["uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"]
}' \
https://openam.example.com:8443/openam/json/realms/root/groups/anotherGroup
{
  "username": "anotherGroup",
  "realm": "/",
  "uniqueMember": [
    "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
  ],
  "cn": [
    "anotherGroup"
  ],
  "dn": [
    "cn=anotherGroup,ou=groups,dc=openam,dc=forgerock,dc=org"
  ],
  "objectclass": [
    "groupofuniquenames",
    "top"
  ],
  "universalid": [
    "id=anotherGroup,ou=group,dc=openam,dc=forgerock,dc=org"
  ]
}

```

### 2.3.1.2. Reading Identities

OpenAM lets users and administrators read profiles by requesting an HTTP GET on [/json/subrealm/users/user-id](#). This allows users and administrators to verify user data, status, and directory. If users or administrators see missing or incorrect information, they can write down the correct information and add it using [Section 2.3.1.3, "Updating Identities"](#). To read a profile on the Top Level Realm, you do not need to specify the realm.

Users can review the data associated with their own accounts, and administrators can also read other user's profiles.

#### Note

If an administrator user is reading their own profile, an additional `roles` element, with a value of `ui-admin` is returned in the JSON response. The XUI verifies this element to grant or deny access to the OpenAM Console.

The following example shows an administrator accessing user data belonging to `demo`:

```
$ curl \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
https://openam.example.com:8443/openam/json/realms/root/users/demo
{
  "id": "demo",
  "_rev": "-320505756",
  "username": "demo",
  "realm": "/",
  "uid": [
    "demo"
  ],
  "universalid": [
    "id=demo,ou=user,dc=openam,dc=forgerock,dc=org"
  ],
  "objectClass": [
    "iplanet-am-managed-person",
    "inetuser",
    "sunFederationManagerDataStore",
    "sunFMSAML2NameIdentifier",
    "devicePrintProfilesContainer",
    "inetorgperson",
    "sunIdentityServerLibertyPPService",
    "iPlanetPreferences",
    "pushDeviceProfilesContainer",
    "iplanet-am-user-service",
    "forgerock-am-dashboard-service",
    "organizationalperson",
    "top",
    "kbaInfoContainer",
    "sunAMAuthAccountLockout",
    "person",
    "oathDeviceProfilesContainer",
    "iplanet-am-auth-configuration-service"
  ],
  "dn": [
    "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
  ],
  "inetUserStatus": [
    "Active"
  ],
  "sn": [
    "demo"
  ],
  "cn": [
    "demo"
  ],
}
```



```
"createTimestamp": [
  "20170105101638Z"
],
"modifyTimestamp": [
  "20170110102632Z"
]
}
```

Use the `_fields` query string parameter to restrict the list of attributes returned. This parameter takes a comma-separated list of JSON object fields to include in the result:

```
$ curl \
--header "iPlanetDirectoryPro: AQIC5w...2NzEz*" \
https://openam.example.com:8443/openam/json/realms/root/users/demo?_fields=username,uid
{"username":"demo","uid":["demo"]}
```

As shown in the examples, OpenAM returns the JSON representation of the profile on success. On failure, OpenAM returns a JSON representation of the error including the HTTP status code.

Using HTTP GET to read also works for other objects such as agent profiles and groups:

```
$ curl \
--header "iPlanetDirectoryPro: AQIC5w...2NzEz*" \
https://openam.example.com:8443/openam/json/realms/root/agents/myAgent
{
  "username": "myAgent",
  "realm": "/",
  "com.sun.identity.agents.config.fqdn.default": [
    "www.example.com"
  ],
  "com.sun.identity.agents.config.repository.location": [
    "centralized"
  ],
  "AgentType": [
    "WebAgent"
  ],
  "userpassword": [
    "{SHA-1}W6ph5Mm5Pz8GgiULbPgZG37mj9g="
  ],
  "com.sun.identity.agents.config.login.url": [
    "[0]=https://openam.example.com:8443/openam/XUI/?realm=#login"
  ],
  "com.sun.identity.agents.config.logout.url": [
    "[0]=https://openam.example.com:8443/openam/XUI/?realm=#logout"
  ],
  "sunIdentityServerDeviceStatus": [
    "Active"
  ]
}
```

## Note

The command output above has been truncated to be more readable. When you read a policy agent profile, OpenAM returns the full profile in JSON format.

The `_prettyPrint` query string parameter can make the resulting JSON easier to read when you are viewing the resulting JSON directly:

```
$ curl \
--header "iPlanetDirectoryPro: AQIC5w...2NzEz*" \
https://openam.example.com:8443/openam/json/realms/root/groups/newGroup?_prettyPrint=true
{
  "username": "newGroup",
  "realm": "dc=openam,dc=forgerock,dc=org",
  "uniquemember": [
    "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
  ],
  "cn": [
    "newGroup"
  ],
  "dn": [
    "cn=newGroup,ou=groups,dc=openam,dc=forgerock,dc=org"
  ],
  "objectclass": [
    "groupofuniquenames",
    "top"
  ],
  "universalid": [
    "id=newGroup,ou=group,dc=openam,dc=forgerock,dc=org"
  ]
}
```

### 2.3.1.3. Updating Identities

OpenAM lets users update their own profiles, and lets administrators update other users' profiles. To update an identity do an HTTP PUT of the JSON representation of the changes to `/json/subrealm/users/user-id`. To update a profile on the Top Level Realm, you do not need to specify the realm.

The following example shows how users can update their own profiles:

```
$ curl \
--request PUT \
--header "iPlanetDirectoryPro: AQIC5...Y3MTAx*" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: protocol=1.0,resource=2.0" \
--data '{"mail": "demo@example.com"}' \
https://openam.example.com:8443/openam/json/realms/root/users/demo
{
  "username": "demo",
  "realm": "/",
  "uid": [
    "demo"
  ],
}
```

```

"mail": [
  "demo@example.com"
],
"universalid": [
  "id=demo,ou=user,dc=openam,dc=forgerock,dc=org"
],
"objectClass": [
  "iplanet-am-managed-person",
  "inetuser",
  "sunFederationManagerDataStore",
  "sunFMSAML2NameIdentifier",
  "devicePrintProfilesContainer",
  "inetorgperson",
  "sunIdentityServerLibertyPPService",
  "iPlanetPreferences",
  "pushDeviceProfilesContainer",
  "iplanet-am-user-service",
  "forgerock-am-dashboard-service",
  "organizationalperson",
  "top",
  "kbaInfoContainer",
  "sunAMAuthAccountLockout",
  "person",
  "oathDeviceProfilesContainer",
  "iplanet-am-auth-configuration-service"
],
"dn": [
  "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
],
"inetUserStatus": [
  "Active"
],
"sn": [
  "demo"
],
"cn": [
  "demo"
],
"createTimestamp": [
  "20170105101638Z"
],
"modifyTimestamp": [
  "20170110105038Z"
],
"roles": [
  "ui-self-service-user"
]
}

```

As shown in the example, OpenAM returns the JSON representation of the profile on success. On failure, OpenAM returns a JSON representation of the error including the HTTP status code.

You can use HTTP PUT to update other objects as well, such as policy agent profiles and groups.

The following example updates a web policy agent profile:

```
$ curl \
```

```
--request PUT \
--header "iPlanetDirectoryPro: AQIC5...Y3MTax*" \
--header "Content-Type: application/json" \
--data '{
  "sunIdentityServerDeviceStatus" : [ "Inactive" ]
}' \
https://openam.example.com:8443/openam/json/realms/root/agents/myAgent?_prettyPrint=true
{
  "username": "myAgent",
  "realm": "/",
  "com.sun.identity.agents.config.fqdn.default": [
    "www.example.com"
  ],
  "com.sun.identity.agents.config.repository.location": [
    "centralized"
  ],
  "AgentType": [
    "WebAgent"
  ],
  "userpassword": [
    "{SHA-1}W6ph5Mm5Pz8GgiULbPgZG37mj9g="
  ],
  "com.sun.identity.agents.config.login.url": [
    "[0]=https://openam.example.com:8443/openam/XUI/?realm=/#login"
  ],
  "com.sun.identity.agents.config.logout.url": [
    "[0]=https://openam.example.com:8443/openam/XUI/?realm=/#logout"
  ],
  "sunIdentityServerDeviceStatus": [
    "Inactive"
  ]
}
```

### Note

The command output above has been truncated to be more readable. When you update a policy agent profile, OpenAM returns the full profile in JSON format.

Notice in the following example that updates `newGroup` the object class value is not included in the JSON sent to OpenAM:

```
$ curl \
--request PUT \
--header "iPlanetDirectoryPro: AQIC5...Y3MTax*" \
--header "Content-Type: application/json" \
--data '{
  "uniquemember": [
    "uid=newUser,ou=people,dc=openam,dc=forgerock,dc=org",
    "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
  ]
}' \
https://openam.example.com:8443/openam/json/realms/root/groups/newGroup
{
  "name": "newGroup",
```

```

"realm": "/",
"uniqueMember": [
  "uid=newUser,ou=people,dc=openam,dc=forgerock,dc=org",
  "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
],
"cn": [
  "newGroup"
],
"dn": [
  "cn=newGroup,ou=groups,dc=openam,dc=forgerock,dc=org"
],
"objectclass": [
  "groupofuniquenames",
  "top"
],
"universalid": [
  "id=newGroup,ou=group,dc=openam,dc=forgerock,dc=org"
]
}

```

### 2.3.1.4. Deleting Identities

OpenAM lets administrators delete a user profile by making an HTTP DELETE call to `/json/subrealm/users/user-id`. To delete a user from the Top Level Realm, you do not need to specify the realm.

The following example removes a user from the top level realm. Only administrators should delete users. The user id is the only field required to delete a user:

```

$ curl \
  --request DELETE \
  --header "iPlanetDirectoryPro: AQIC5w...2NzEz*" \
  https://openam.example.com:8443/openam/json/realms/root/users/bjensen
{"success":"true"}

```

On success, OpenAM returns a JSON object indicating success. On failure, OpenAM returns a JSON representation of the error including the [HTTP status code](#).

You can use this same logic for other resources such as performing an HTTP DELETE of an agent profile or of a group:

```

$ curl \
  --request DELETE \
  --header "iPlanetDirectoryPro: AQIC5w...2NzEz*" \
  https://openam.example.com:8443/openam/json/realms/root/agents/my0Auth2ClientAgent
{"success":"true"}

```

```

$ curl \
  --request DELETE \
  --header "iPlanetDirectoryPro: AQIC5w...2NzEz*" \
  https://openam.example.com:8443/openam/json/realms/root/groups/newGroup
{"success":"true"}

```

## Note

Deleting a user does not automatically remove any of the user's sessions. If you are using stateful sessions, you can remove a user's sessions by checking for any sessions for the user and then removing them using the console's Sessions page. If you are using stateless sessions, you cannot remove users' sessions; you must wait for the sessions to expire.

### 2.3.1.5. Listing Identities

OpenAM lets administrators list identities by making an HTTP GET call to `/json/subrealm/users/?_queryId=*`. To query the Top Level Realm, you do not need to specify the realm:

```
$ curl \
--header "iPlanetDirectoryPro: AQIC5w...2NzEz*" \
"https://openam.example.com:8443/openam/json/realms/root/users?_queryId=*"
{
  "result": [
    {
      "username": "amAdmin",
      "realm": "dc=openam,dc=forgerock,dc=org",
      "sunIdentityMSISDNNumber": [],
      "mail": [],
      "sn": [
        "amAdmin"
      ],
      "givenName": [
        "amAdmin"
      ],
      "universalid": [
        "id=amAdmin,ou=user,dc=openam,dc=forgerock,dc=org"
      ],
      "cn": [
        "amAdmin"
      ],
      "iplanet-am-user-success-url": [],
      "telephoneNumber": [],
      "roles": [
        "ui-global-admin",
        "ui-realm-admin"
      ],
      "iplanet-am-user-failure-url": [],
      "inetuserstatus": [
        "Active"
      ],
      "postalAddress": [],
      "dn": [
        "uid=amAdmin,ou=people,dc=openam,dc=forgerock,dc=org"
      ],
      "employeeNumber": [],
      "iplanet-am-user-alias-list": []
    },
    {
      "username": "demo",
      "realm": "dc=openam,dc=forgerock,dc=org",
      "uid": [
```

```

    "demo"
  ],
  "createTimestamp": [
    "20160108155628Z"
  ],
  "inetUserStatus": [
    "Active"
  ],
  "mail": [
    "demo.user@example.com"
  ],
  "sn": [
    "demo"
  ],
  "cn": [
    "demo"
  ],
  "objectClass": [
    "devicePrintProfilesContainer",
    "person",
    "sunIdentityServerLibertyPPService",
    "sunFederationManagerDataStore",
    "inetorgperson",
    "oathDeviceProfilesContainer",
    "iPlanetPreferences",
    "iplanet-am-auth-configuration-service",
    "sunFMSAML2NameIdentifier",
    "organizationalperson",
    "inetuser",
    "kbaInfoContainer",
    "forgerock-am-dashboard-service",
    "iplanet-am-managed-person",
    "iplanet-am-user-service",
    "sunAMAuthAccountLockout",
    "top"
  ],
  "kbaInfo": [
    {
      "questionId": "2",
      "answer": {
        "$crypto": {
          "value": {
            "algorithm": "SHA-256",
            "data": "VXGtsnjJMC...MQJ/goU5hkfF"
          },
          "type": "salted-hash"
        }
      }
    },
    {
      "questionId": "1",
      "answer": {
        "$crypto": {
          "value": {
            "algorithm": "SHA-256",
            "data": "cfYYzi9U...rVfFl0TdW0iX"
          },
          "type": "salted-hash"
        }
      }
    }
  ]
}

```

```

    }
  },
  "dn": [
    "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
  ],
  "universalid": [
    "id=demo,ou=user,dc=openam,dc=forgerock,dc=org"
  ],
  "modifyTimestamp": [
    "20160113010610Z"
  ]
}
],
"resultCount": 2,
"pagedResultsCookie": null,
"totalPagedResultsPolicy": "NONE",
"totalPagedResults": -1,
"remainingPagedResults": -1
}

```

The `users` endpoint also supports the `_queryFilter` parameter to alter the returned results. For more information, see [Section A.2.12, "Query"](#).

The `_queryId=*` parameter also works for other types of objects, such as agent profiles and groups:

```

$ curl \
--header "iPlanetDirectoryPro: AQIC5w...2NzEz*" \
"https://openam.example.com:8443/openam/json/realms/root/agents?_queryId=*"
{
  "result" : [ "wsp", "wsc", "agentAuth", "SecurityTokenService" ],
  "resultCount" : 4,
  "pagedResultsCookie" : null,
  "remainingPagedResults" : -1
}

```

```

$ curl \
--header "iPlanetDirectoryPro: AQIC5w...2NzEz*" \
"https://openam.example.com:8443/openam/json/realms/root/groups?_queryId=*"
{
  "result" : [ "newGroup", "anotherGroup" ],
  "resultCount" : 2,
  "pagedResultsCookie" : null,
  "remainingPagedResults" : -1
}

```

As the result lists include all objects, this capability to list identity names is mainly useful in testing.

As shown in the examples, OpenAM returns the JSON representation of the resource list if successful. On failure, OpenAM returns a JSON representation of the error including the HTTP status code.



### 2.3.1.6. Retrieving Identities Using the Session Cookie

If you only have access to the `iPlanetDirectoryPro` session cookie, you can retrieve the user ID by performing an HTTP POST operation on the `/json/users` endpoint using the `idFromSession` action:

```
$ curl \
--verbose \
--request POST \
--header "Content-Type: application/json" \
--header "iPlanetDirectoryPro: AQIC5wM2LY4Sfcz...c50Dk4MjYzMzA2MQ..*" \
http://openam.example.com:8080/openam/json/realms/root/users?_action=idFromSession

{
  "id": "demo",
  "realm": "/",
  "dn": "id=demo,ou=user,dc=openam,dc=forgerock,dc=org",
  "successURL": "/openam/console",
  "fullLoginURL": null
}
```

### 2.3.1.7. Changing Passwords

Users other than the top-level administrator can change their own passwords with an HTTP POST to `/json/subrealm/users/username?_action=changePassword` including the new password as the value of `userpassword` in the request data.

#### Note

Changing the top-level administrator's password requires a more complex procedure. See Section 8.2, "Changing the amadmin User's Password" for more information.

Users must provide the current password, which is set in the request as the value of the `currentpassword`.

For cases where users have forgotten their password, see Section 3.3, "Retrieving Forgotten Usernames" in the *User Self Service Guide* instead.

The following example shows a successful request to change the `demo` user's password to `password`:

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "iPlanetDirectoryPro: AQIC5w...NTcy*" \
--data '{
  "currentpassword": "changeit",
  "userpassword": "password"
}' \
https://openam.example.com:8443/openam/json/realms/root/users/demo?_action=changePassword
{}

```

On success, the response is an empty JSON object `{}` as shown in the example.

On failure, OpenAM returns a JSON representation of the error including the HTTP status code. See also Section A.2.13, "HTTP Status Codes" for more information.

Administrators can change non-administrative users' passwords with an HTTP PUT to `/json/subrealm/users/username` including the new password as the value of `userpassword` in the request data.

Unlike users, administrators do not provide users' current passwords when changing passwords.

The following example shows a successful request by an administrator to change the `demo` user's password to `cangetin`:

```
$ curl \
--request PUT \
--header "iPlanetDirectoryPro: AQIC5w...NTcy*" \
--header "Content-Type: application/json" \
--data '{
  "userpassword": "cangetin"
}' \
https://openam.example.com:8443/openam/json/realms/root/users/demo
{
  "_id": "demo",
  "_rev": "-1942782480",
  "username": "demo",
  "realm": "/",
  "uid": [
    "demo"
  ],
  "mail": [
    "demo@example.com"
  ],
  "universalid": [
    "id=demo,ou=user,dc=openam,dc=forgerock,dc=org"
  ],
  "objectClass": [
    "iplanet-am-managed-person",
    "inetuser",
    "sunFederationManagerDataStore",
    "sunFMSAML2NameIdentifier",
    "devicePrintProfilesContainer",
    "inetorgperson",
    "sunIdentityServerLibertyPPService",
    "iPlanetPreferences",
    "pushDeviceProfilesContainer",
    "iplanet-am-user-service",
    "forgerock-am-dashboard-service",
    "organizationalperson",
    "top",
    "kbaInfoContainer",
    "sunAMAuthAccountLockout",
    "person",
    "oathDeviceProfilesContainer",
    "iplanet-am-auth-configuration-service"
  ],
  "dn": [
    "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
  ],
  "inetUserStatus": [
```

```

    "Active"
  ],
  "sn": [
    "demo"
  ],
  "cn": [
    "demo"
  ],
  "modifyTimestamp": [
    "20170110105705Z"
  ],
  "createTimestamp": [
    "20170105101638Z"
  ]
}

```

As shown in the example, OpenAM returns the JSON representation of the profile on success. On failure, OpenAM returns a JSON representation of the error including the HTTP status code. See also Section A.2.13, "HTTP Status Codes" for more information.

## 2.3.2. Realm Management

This section shows how to create, read, update, and delete realms using the `/json/global-config/realms` endpoint.

### Tip

You can use the OpenAM API Explorer for detailed information about this endpoint and to test it against your deployed OpenAM instance.

In the AM console, click the Help icon, and then navigate to API Explorer > /global-config > /realms.

The following sections cover managing realms with the JSON-based RESTful API:

- Section 2.3.2.1, "Required Properties for Managing Realms"
- Section 2.3.2.2, "Creating Realms"
- Section 2.3.2.3, "Listing Realms"
- Section 2.3.2.4, "Reading Realms"
- Section 2.3.2.5, "Updating Realms"
- Section 2.3.2.6, "Deleting Realms"

### 2.3.2.1. Required Properties for Managing Realms

The following table shows the required properties when managing realms using the REST API:

*Table 2.1. Realm Properties for JSON-based API*

Realm Property	Purpose
<code>name</code>	The name of the realm.  Do not use the names of OpenAM REST endpoints as the name of a realm. Names that should not be used include <code>users</code> , <code>groups</code> , <code>realms</code> , <code>policies</code> , and <code>applications</code> .
<code>active</code>	Whether the realm is to be active, or not.  Specify either <code>true</code> or <code>false</code> .
<code>parentPath</code>	The path of the parent realm.
<code>aliases</code>	An array of any applicable aliases associated with the realm. Be aware that an alias can only be used once. Entering an alias used by another realm will remove the alias from that realm and you will lose configuration.

The following shows an example JSON payload when managing a realm:

```
{
  "name": "mySubRealm",
  "active": true,
  "parentPath": "/",
  "aliases": [ "payroll.example.com" ]
}
```

### 2.3.2.2. Creating Realms

OpenAM lets administrators create a realm by performing an HTTP POST of the JSON representation of the realm to `/json/global-config/realms`.

The following example creates a new, active subrealm in the top level realm, named `mySubRealm`:

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
--data '{
  "name": "mySubRealm",
  "active": true,
  "parentPath": "/",
  "aliases": [ "payroll.example.com" ]
}' \
https://openam.example.com:8443/openam/json/global-config/realms
{
  "_id": "L2Fub3RoZXJSZWFSbQ",
  "_rev": "-1054013208",
  "parentPath": "/",
  "active": true,
  "name": "mySubRealm",
  "aliases": [ "payroll.example.com" ]
}
```

OpenAM returns a 201 HTTP status code and a JSON representation of the realm on success. The value returned in the `_id` field is used in subsequent REST calls relating to the realm. On failure, OpenAM returns a JSON representation of the error including the HTTP status code. For more information, see [Section A.2.13, "HTTP Status Codes"](#).

### 2.3.2.3. Listing Realms

To list and query realms, perform an HTTP GET on the `/json/global-config/realms` endpoint, and set the `_queryFilter` query string parameter to `true` as in the following example, which lists all available realms:

```
$ curl \
--header "iPlanetDirectoryPro: AqIC5..." \
https://openam.example.com:8443/openam/json/global-config/realms?_queryFilter=true
{
  "result": [
    {
      "_id": "Lw",
      "_rev": "252584985",
      "parentPath": null,
      "active": true,
      "name": "/",
      "aliases": [
        "openam.example.com",
        "openam"
      ]
    },
    {
      "_id": "L215U3ViUmVhbG0",
      "_rev": "949061198",
      "parentPath": "/",
      "active": true,
      "name": "mySubRealm",
      "aliases": [
        "payroll.example.com"
      ]
    }
  ],
  "resultCount": 2,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

For more information about using the `_queryString` parameter in REST calls, see [Section A.2.12, "Query"](#).

### 2.3.2.4. Reading Realms

To read a realm's details, perform an HTTP GET on the `/json/global-config/realms/realm-id` endpoint, where `realm-id` is the value of `_id` for the realm.

The following example shows an administrator receiving information about a realm called `mySubRealm`, which has an `_id` value of `L215U3ViUmVhbG0`:

```
$ curl \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
https://openam.example.com:8443/openam/json/global-config/realms/L215U3ViUmVhbG0
{
  "_id": "L215U3ViUmVhbG0",
  "_rev": "949061698",
  "parentPath": "/",
  "active": true,
  "name": "mySubRealm",
  "aliases": [
    "payroll.example.com"
  ]
}
```

OpenAM returns a 200 HTTP status code and a JSON representation of the realm on success. On failure, OpenAM returns a JSON representation of the error including the HTTP status code. For more information, see [Section A.2.13, "HTTP Status Codes"](#).

### 2.3.2.5. Updating Realms

To update a realm's aliases or to toggle between active and inactive, perform an HTTP PUT on the `/json/global-config/realms/realm-id` endpoint, where `realm-id` is the value of `_id` for the realm.

The request should include an updated JSON representation of the complete realm. Note that you cannot alter the `name` or `parent` properties of a realm.

The following example shows how to update a realm called `mySubRealm`, which has an `_id` value of `L215U3ViUmVhbG0`. The example changes the realm status to inactive:

```
$ curl \
--request PUT \
--header "iplanetDirectoryPro: AQIC5...Y3MTAx*" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: protocol=1.0" \
--data '{
  "parentPath": "/",
  "active": false,
  "name": "mySubRealm",
  "aliases": [
    "payroll.example.com"
  ]
}' \
https://openam.example.com:8443/openam/json/global-config/realms/L215U3ViUmVhbG0
{
  "_id": "L215U3ViUmVhbG0",
  "_rev": "94906213",
  "parentPath": "/",
  "active": false,
  "name": "mySubRealm",
  "aliases": [
    "payroll.example.com"
  ]
}
```

OpenAM returns a 200 HTTP status code and a JSON representation of the realm on success. On failure, OpenAM returns a JSON representation of the error including the HTTP status code. For more information, see [Section A.2.13, "HTTP Status Codes"](#).

### 2.3.2.6. Deleting Realms

To delete a realm, perform an HTTP DELETE on the `/json/global-config/realms/realm-id` endpoint, where *realm-id* is the value of `_id` for the realm.

The following example shows how to delete a realm called `mySubRealm`, which has an `_id` value of `L215U3ViUmVhbG0`.

#### Caution

Make sure that you do not have any information you need within a realm before deleting it. Once a realm is deleted, the only way to restore it is to return to a backed up deployment of OpenAM.

```
$ curl \
--request DELETE \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
https://openam.example.com:8443/openam/json/global-config/realms/L215U3ViUmVhbG0
{
  "_id": "L215U3ViUmVhbG0",
  "_rev": "949061708",
  "parentPath": "/",
  "active": false,
  "name": "mySubRealm",
  "aliases": [
    "payroll.example.com"
  ]
}
```

OpenAM returns a 200 HTTP status code and a JSON representation of the deleted realm on success. On failure, OpenAM returns a JSON representation of the error including the HTTP status code. For more information, see [Section A.2.13, "HTTP Status Codes"](#).

## 2.4. Customizing Realms

This section shows how to delegate realm administration privileges to users or groups of users and how to configure a policy agent to be directed to a realm and application when requesting policy decisions:

- [Section 2.4.1, "Delegating Realm Administration Privileges"](#)
- [Section 2.4.2, "Working With Realms and Policy Agents"](#)

## 2.4.1. Delegating Realm Administration Privileges

You assign administration privileges to groups of users.

You can grant privileges through the AM console, see [Procedure 2.3, "To Delegate Privileges using the AM Console"](#), or by using the **ssoadm add-privileges** command, see ["ssoadm add-privileges" in the Reference](#).

### *Procedure 2.3. To Delegate Privileges using the AM Console*

1. On the Realms page, click the realm for which you want to delegate administration to view the realm configuration.

Delegating administration privileges in the top level realm allows members of the group full administration access to the OpenAM instance. Administration privileges in any other realm allows the group to administrate only in that realm, and any child realms.

2. On the Privileges tab, click the name of the group to which you intend to grant access.
3. Select the administrative privileges to delegate for the realm:
  - To grant users in the group access to the administration console for the realm, select Read and write access to all realm and policy properties.

In AM 5, administrators can use the AM administration console as follows:

- Delegated administrators with the **RealmAdmin** privilege can access full administration console functionality within the realms they can administer.
- Administrators with lesser privileges, such as the **PolicyAdmin** privilege, can not access the OpenAM administration console.
- Both the top level administrator (such as **amadmin**) and delegated administrators in the Top Level Realm with the **RealmAdmin** privilege have access to full console functionality in all realms and can access OpenAM's global configuration.
- To grant users in the group access to REST endpoints, select them from the list.

For information about the available OpenAM privileges, see [Section 10.2, "Realm Privileges Configuration Reference"](#).

4. Save your work.

To enable delegated subrealm administrators to invalidate sessions, you must add an attribute to their entry in the data store, as described in the following procedure:

### *Procedure 2.4. To Enable Delegated Subrealm Administrators to Invalidate Sessions*

1. Create an LDIF file that modifies the distinguished name entry of the subrealm administrator, adds the **iplanet-am-session-destroy-sessions** attribute, and sets its value to the subrealm's DN.



In the following example, the delegated administrator is named `subRealmAdmin` and the subrealm is called `mySubRealm`:

```
dn: uid=subrealmadmin,ou=people,dc=openam,dc=forgerock,dc=org
changetype: modify
add: objectClass
objectClass: iplanet-am-session-service
-
add: iplanet-am-session-destroy-sessions
iplanet-am-session-destroy-sessions: o=mysubrealm,ou=services,dc=openam,dc=forgerock,dc=org
```

#### Note

All values in the LDIF must be in lowercase, even if the subrealm or administrator name is not.

- Run the **ldapmodify** command included with OpenDJ to apply the LDIF file to the user data store. For example:

```
$ ./ldapmodify -h opendj.example.com -p 1389 -D "cn=Directory Manager" \
-w myBindPassword -f /path/to/ldif.file

# Processing MODIFY request for uid=subrealmadmin,ou=people,dc=openam,dc=forgerock,dc=org
# MODIFY operation successful for DN uid=subrealmadmin,ou=people,dc=openam,dc=forgerock,dc=org
```

The delegated realm administrator will now be able to invalidate sessions created in the subrealm.

## 2.4.2. Working With Realms and Policy Agents

You can configure a policy agent to be directed to a realm and application when requesting policy decisions, or to log users into a different realm than the policy agent's realm:

- Procedure 2.5, "To Specify the Realm and Application for Policy Decisions"
- Procedure 2.6, "To Configure a Web or J2EE Agent for Log In to a Realm"

### *Procedure 2.5. To Specify the Realm and Application for Policy Decisions*

By default, policy agents request policy decisions in the Top Level Realm (/) from the default policy set, `iPlanetAMWebAgentService`. When the realm and policy set differ for your policy agent, you can specify the realm and policy set in the policy agent profile. OpenAM then directs requests from the policy agent to the specified realm and policy set, so this is backwards compatible with existing policy agents.

- In the AM console, browse to Realms > *Realm Name* > Applications > Agents > *Web or J2EE* > *Agent Name* > OpenAM Services > Policy Client Service.
- Set the Realm and Policy Set.

Note that Policy Sets are labelled as "Application" in some parts of the user interface.

For example, if the realm is `/hr` and the policy set is `myHRApp`:

- Realm: `/hr`
- Application: `myHRApp`

3. Save your work.

### *Procedure 2.6. To Configure a Web or J2EE Agent for Log In to a Realm*

You might choose to configure your agent in one realm, yet have your real users authenticate through another realm. In this case, you want your policy agents to redirect users to authenticate to their realm, rather than the agent realm:

1. In the AM console, navigate to Realms > *Realm Name* > Applications > Agents > *Web or J2EE* > *Agent Name* > OpenAM Services.
2. Add login and logout URLs, including the top level realms and the subrealm in the query string.

For example, if your *Realm Name* is `hr`, and you access OpenAM at `http://openam.example.com:8080/openam`:

- Login URL: `http://openam.example.com:8080/openam/XUI/?realm=/hr#login/`
- Logout URL: `http://openam.example.com:8080/openam/XUI/?realm=/hr#logout/`

3. Save your work.

## Chapter 3

# Setting Up Identity Data Stores

This chapter shows what is an identity data store and how to configure and customize them.

## 3.1. Introducing Identity Data Stores

An identity data store, or an identity repository, is a persistent repository of user data, for example, a directory server, an LDAP, a database, and others. At install time you can configure OpenAM to create an embedded OpenDJ server that works as an identity data store among other uses, but you can also consider to configure an external identity data store at install time or afterwards.

OpenAM uses other types of data stores, like the configuration data store, the UMA data store, and the Core Token Service (CTS) data store, but those are not being discussed in this chapter.

## 3.2. Implementing Identity Data Stores

When you first set up a realm, the new real inherits the data store from the parent realm. For example, in an installation where the Top Level Realm has the embedded OpenDJ server as the identity data store, any new realm created would have the embedded OpenDJ server as the identity data store by default.

Yet, if your administrators are in one realm and your users in another, your new child realm might retrieve users from a different data store.

To see and modify the embedded OpenDJ identity data store server configuration, navigate to Realms > *Realm Name* > Data Store > embedded.

Before configuring an OpenAM data store as an external identity data store, make sure that you have prepared the external identity data store for OpenAM. For more information, see [Section 1.4.1, "Preparing an External Identity Repository"](#) in the *Installation Guide*.

To configure an external identity data store, see [Procedure 3.1, "To Configure an Identity Data Store"](#).

### *Procedure 3.1. To Configure an Identity Data Store*

1. In the AM console, browse to Realms > *Realm Name* > Data Stores.

2. Click New in the Data Stores table to create a data store profile, and to provide the information needed to connect to the data store.

3. In the first screen, name the data store and select the type of data store.

Most data stores are directory services, though the Database Repository lets you connect to an SQL database through JDBC.

4. In the second screen, provide information on how to connect to your data store, and then click Finish to save your work.

See the following sections for hints depending on the type of data store.

- Section 10.1.1, "Active Directory Configuration Properties"
- Section 10.1.2, "Active Directory Application Mode Configuration Properties"
- Section 10.1.3, "Database Repository (Early Access) Configuration Properties"
- Section 10.1.4, "Generic LDAPv3 Configuration Properties"
- Section 10.1.5, "OpenDJ Configuration Properties"
- Section 10.1.6, "Sun/Oracle DSEE Configuration Properties"
- Section 10.1.7, "Tivoli Directory Server Configuration Properties"

5. You must index several directory attributes as a post-configuration step if you configured the data store as follows:

- You configured the data store to access an external identity repository.
- You used the "Load schema when finished" option.

For more information about indexing external identity repository attributes, see Procedure 1.10, "To Index External Identity Repository Attributes" in the *Installation Guide*.

6. Click the Subjects tab, and make sure the connection to your new data store is working, by searching for a known identity.

By default the Subjects list only retrieves 100 entries from the data store. Narrow your search if you do not see the identity you are looking for.

7. If you no longer need the connection to the inherited data store *in this realm*, then you can delete its entry in the Data Stores table.

Also, once you change the data store for a realm, you might opt to change the authentication module configuration to use your realm data store, rather than the inherited settings. See Section 2.2, "Configuring Authentication Modules" in the *Authentication and Single Sign-On Guide*.

## 3.3. Customizing Identity Data Stores

Follow this section to create custom attributes to store additional information in your identity data stores, or to create identity repository plugins to customize how OpenAM maps users and groups to a realm if your deployment require different functionality than the already built-in OpenAM:

- Section 3.3.1, "Customizing Profile Attributes".
- Section 3.3.2, "Customizing Identity Data Storage".

### 3.3.1. Customizing Profile Attributes

You can extend user profiles by adding custom attributes. This section demonstrates how to add a custom attribute to a user profile when storing user profiles in the embedded LDAP directory.

Adding a custom attribute involves both updating the `iPlanetAMUserService`, and also updating the identity repository schema to hold the new attribute. Furthermore, to allow users to update the attribute in their own profiles, you must also update the OpenAM policy configuration stored in the configuration directory.

#### Important

Custom profile attributes do not appear in the user profile when users log in to OpenAM using the XUI.

This section includes the following procedures.

- Procedure 3.2, "To Update the AMUser Service For the New Attribute"
- Procedure 3.3, "To Update the Identity Repository For the New Attribute"
- Procedure 3.4, "To Allow Users To Update the New Attribute"

#### *Procedure 3.2. To Update the AMUser Service For the New Attribute*

Follow the steps below to create a custom attribute in OpenAM.

1. Create a backup copy of the configuration file for the `iPlanetAMUserService`.

```
$ cp ~/openam/config/xml/amUser.xml ~/openam/config/xml/amUser.xml.orig
```

2. Edit the file to add your attribute as one of the list of `<User>` attributes.

```
<AttributeSchema name="customAttribute"
  type="single"
  syntax="string"
  any="display"
  i18nKey="Custom Attribute">
</AttributeSchema>
```

Here, the name refers to the attribute type name used in LDAP. The `i18nKey` holds either the reference, or in this case the content, of the text that appears in the user interface.

3. Delete `iPlanetAMUserService`, and then create it from your updated configuration file.

```
$ cd /path/to/tools/openam/bin/
$ ssoadm \
  delete-svc \
    --adminid amadmin \
    --password-file /tmp/pwd.txt \
    --servicename iPlanetAMUserService

Service was deleted.
$ ssoadm \
  create-svc \
    --adminid amadmin \
    --password-file /tmp/pwd.txt \
    --xmlfile $HOME/openam/config/xml/amUser.xml

Service was added.
```

### Procedure 3.3. To Update the Identity Repository For the New Attribute

Follow the steps below to update the identity repository LDAP schema for the custom attribute, and then update OpenAM to use the custom attribute and object class.

If you are adding an existing attribute that is already allowed on user profile entries, you can skip this procedure.

#### Tip

If you are using OpenDJ as the identity repository, you can update the schema through OpenDJ Control Panel > Schema > Manage Schema, as described in the OpenDJ documentation.

1. Prepare the attribute type object class definitions in LDIF format.

```
$ cat custom-attr.ldif
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( temp-custom-attr-oid NAME 'customAttribute' EQUALITY case
  IgnoreMatch ORDERING caseIgnoreOrderingMatch SUBSTR caseIgnoreSubstrings
  Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 USAGE
  userApplications )
-
add: objectClasses
objectClasses: ( temp-custom-oc-oid NAME 'customObjectclass' SUP top AUXILIARY
  MAY customAttribute )
```

2. Add the schema definitions to the directory. The following example assumes that you are using OpenDJ 4.0 and later:

```
$ /path/to/openssl/bin/ldapmodify \
--port 1389 \
--hostname openam.example.com \
--bindDN "cn=Directory Manager" \
--bindPassword password \
custom-attr.ldif
Processing MODIFY request for cn=schema
MODIFY operation successful for DN cn=schema
```

3. In the AM console, browse to Realms > *Realm Name* > Data Stores > *Data Store Name*.
4. Add the object class, here `customObjectclass`, to the LDAP User Object Class list.
5. Add the attribute type, here `customAttribute`, to the LDAP User Attributes list.
6. Save your work.

### Procedure 3.4. To Allow Users To Update the New Attribute

Follow these steps to make the new attribute editable by users. The steps imply use of the embedded configuration directory. If you use a different directory server to store the configuration, then adapt them for your tools.

1. Login to the control panel for the embedded configuration directory.

```
$ ./openam/opens/b/bin/control-panel &
```


Connect using bind DN `cn=Directory Manager` and the the password for `amadmin`.

2. Select Manage Entries to open the LDAP browser.
3. Search with **LDAP Filter:** set to `ou=SelfWriteAttributes`, and then expand the tree views to see the two entries found.
4. In the entry under `iPlanetAMPolicyService`, edit the `sunKeyValue` attribute to add your custom attribute to the list of self-writable attributes, as in `<Value>customAttribute</Value>`.
5. In the entry under `sunEntitlementIndexes`, edit the `sunKeyValue` attribute by adding your custom attribute to the `attributes` JSON array.
6. Restart OpenAM or the web container where it runs. The following example applies to Tomcat.

```
$ /path/to/tomcat/bin/shutdown.sh
$ /path/to/tomcat/bin/startup.sh
```

7. Login to the AM console as a user to check that a user can save a value for your new, custom attribute.

Figure 3.1. User Custom Attribute

 **Information**  
Profile was updated.

**Babs Jensen**

Custom Attribute:	<input type="text" value="This is a test."/>
First Name:	<input type="text" value="Barbara"/>
* Last Name:	<input type="text" value="Jensen"/>
* Full Name:	<input type="text" value="Babs Jensen"/>

### 3.3.2. Customizing Identity Data Storage

OpenAM maps user and group identities into a realm using data stores. An OpenAM data store relies on a Java identity repository (IdRepo) plugin to implement interaction with the identity repository where the users and groups are stored.

#### 3.3.2.1. About the Identity Repository Plugin

This section describes how to create a custom identity repository plugin. OpenAM includes built-in support for LDAP and JDBC identity repositories. For most deployments, you therefore do not need to create your own custom identity repository plugin. Only create custom identity repository plugins for deployments with particular requirements not met by built-in OpenAM functionality.

##### Tip

Before creating your own identity repository plugin, start by reading the OpenAM source code for the `FilesRepo` or `DatabaseRepo` plugins under `com.sun.identity.idm.plugins`.

##### 3.3.2.1.1. IdRepo Inheritance

Your identity repository plugin class must extend the `com.sun.identity.idm.IdRepo` abstract class, and must include a constructor method that takes no arguments.

##### 3.3.2.1.2. IdRepo Lifecycle

When OpenAM instantiates your IdRepo plugin, it calls the `initialize()` method.

```
public void initialize(Map configParams)
```



The `configParams` are service configuration parameters for the realm where the IdRepo plugin is configured. The `configParams` normally serve to set up communication with the underlying identity data store. OpenAM calls the `initialize()` method once, and considers the identity repository ready for use.

If you encounter errors or exceptions during initialization, catch and store them in your plugin for use later when OpenAM calls other plugin methods.

After initialization, OpenAM calls the `addListener()` and `removeListener()` methods to register listeners that inform OpenAM client code of changes to identities managed by your IdRepo.

```
public int addListener(SSOToken token, IdRepoListener listener)
public void removeListener()
```

You must handle listener registration in your IdRepo plugin, and also return events to OpenAM through the `IdRepoListener`.

When stopping, OpenAM calls your IdRepo plugin `shutdown()` method.

```
public void shutdown()
```

You are not required to implement `shutdown()` unless your IdRepo plugin has shut down work of its own to do, such as close connections to the underlying identity data store.

### 3.3.2.1.3. IdRepo Plugin Capabilities

Your IdRepo plugin provides OpenAM with a generic means to manage subjects—including users and groups but also special types such as roles, realms, and agents— and to create, read, update, delete, and search subjects. In order for OpenAM to determine your plugin's capabilities, it calls the methods described in this section.

```
public Set getSupportedTypes()
```

The `getSupportedTypes()` method returns a set of `IdType` objects, such as `IdType.USER` and `IdType.GROUP`. You can either hard-code the supported types into your plugin, or make them configurable through the IdRepo service.

```
public Set getSupportedOperations(IdType type)
```

The `getSupportedOperations()` method returns a set of `IdOperation` objects, such as `IdOperation.CREATE` and `IdOperation.EDIT`. You can also either hard-code these, or make them configurable.

```
public boolean supportsAuthentication()
```

The `supportsAuthentication()` method returns true if your plugin supports the `authenticate()` method.

### 3.3.2.2. Identity Repository Plugin Implementation

Your IdRepo plugin implements operational methods depending on what you support. These methods perform the operations in your data store.

## Create

OpenAM calls `create()` to provision a new identity in the repository, where `name` is the new identity's name, and `attrMap` holds the attributes names and values.

```
public String create(SSOToken token, IdType type, String name, Map attrMap)
    throws IdRepoException, SSOException
```

## Read

OpenAM calls the following methods to retrieve subjects in the identity repository, and to check account activity. If your data store does not support binary attributes, return an empty `Map` for `getBinaryAttributes()`.

```
public boolean isExists(
    SSOToken token,
    IdType type,
    String name
) throws IdRepoException, SSOException

public boolean isActive(
    SSOToken token,
    IdType type,
    String name
) throws IdRepoException, SSOException

public Map getAttributes(
    SSOToken token,
    IdType type,
    String name
) throws IdRepoException, SSOException

public Map getAttributes(
    SSOToken token,
    IdType type,
    String name,
    Set attrNames
) throws IdRepoException, SSOException

public Map getBinaryAttributes(
    SSOToken token,
    IdType type,
    String name,
    Set attrNames
) throws IdRepoException, SSOException

public RepoSearchResults search(
    SSOToken token,
    IdType type,
    String pattern,
    Map avPairs,
    boolean recursive,
    int maxResults,
    int maxTime,
    Set returnAttrs
) throws IdRepoException, SSOException

public RepoSearchResults search(
    SSOToken token,
```

```

    IdType type,
    String pattern,
    int maxTime,
    int maxResults,
    Set returnAttrs,
    boolean returnAllAttrs,
    int filterOp,
    Map avPairs,
    boolean recursive
) throws IdRepoException, SS0Exception

```

## Edit

OpenAM calls the following methods to update a subject in the identity repository.

```

public void setAttributes(
    SS0Token token,
    IdType type,
    String name,
    Map attributes,
    boolean isAdd
) throws IdRepoException, SS0Exception

public void setBinaryAttributes(
    SS0Token token,
    IdType type,
    String name,
    Map attributes,
    boolean isAdd
) throws IdRepoException, SS0Exception

public void removeAttributes(
    SS0Token token,
    IdType type,
    String name,
    Set attrNames
) throws IdRepoException, SS0Exception

public void modifyMemberShip(
    SS0Token token,
    IdType type,
    String name,
    Set members,
    IdType membersType,
    int operation
) throws IdRepoException, SS0Exception

public void setActiveStatus(
    SS0Token token,
    IdType type,
    String name,
    boolean active
)

```

## Authenticate

OpenAM calls `authenticate()` with the credentials from the `DataStore` authentication module.

```
public boolean authenticate(Callback[] credentials)
    throws IdRepoException, AuthLoginException
```

## Delete

The `delete()` method removes the subject from the identity repository. The `name` specifies the subject.

```
public void delete(SSOToken token, IdType type, String name)
    throws IdRepoException, SSOException
```

## Service

The `IdOperation.SERVICE` operation is rarely used in recent OpenAM deployments.

### 3.3.2.3. Identity Repository Plugin Deployment

When you build your IdRepo plugin, include `openam-core-14.0.0.jar` in the classpath. This file is found under `WEB-INF/lib/` where OpenAM is deployed.

You can either package your plugin as a .jar, and then add it to `WEB-INF/lib/`, or add the classes under `WEB-INF/classes/`.

To register your plugin with OpenAM, you add a `SubSchema` to the `sunIdentityRepositoryService` using the `ssoadm` command. First, you create the `SubSchema` document having the following structure.

```
<SubSchema i18nKey="x4000" inheritance="multiple" maintainPriority="no"
    name="CustomRepo" supportsApplicableOrganization="no" validate="yes">
  <AttributeSchema cosQualifier="default" isSearchable="no"
    name="RequiredValueValidator" syntax="string"
    type="validator" >
    <DefaultValues>
      <Value>com.sun.identity.sm.RequiredValueValidator</Value>
    </DefaultValues>
  </AttributeSchema>
  <AttributeSchema any="required" cosQualifier="default"
    i18nKey="x4001" isSearchable="no"
    name="sunIdRepoClass" syntax="string"
    type="single" validator="RequiredValueValidator" >
    <DefaultValues>
      <Value>org.test.CustomRepo</Value>
    </DefaultValues>
  </AttributeSchema>
  <AttributeSchema cosQualifier="default" i18nKey="x4002" isSearchable="no"
    name="sunIdRepoAttributeMapping" syntax="string" type="list">
    <DefaultValues>
      <Value></Value>
    </DefaultValues>
  </AttributeSchema>
</SubSchema>
```

Also include the `AttributeSchema` required to configure your IdRepo plugin.

Notice the `i18nKey` attributes on `SubSchema` elements. The `i18nKey` attribute values correspond to properties in the `amIdRepoService.properties` file under `WEB-INF/classes/` where OpenAM is deployed. The

AM console displays the label for the configuration user interface that it retrieves from the value of the `i18nKey` property in the `amIdRepoService.properties` file.

To make changes to the properties, first extract `amIdRepoService.properties` and if necessary the localized versions of this file from `openam-core-14.0.0.jar` to `WEB-INF/classes/` where OpenAM is deployed. For example, if OpenAM is deployed under `/path/to/tomcat/webapps/openam`, then you could run the following commands.

```
$ cd /path/to/tomcat/webapps/openam/WEB-INF/classes/
$ jar -xvf ../lib/openam-core-14.0.0.jar amIdRepoService.properties
inflated: amIdRepoService.properties
```

Register your plugin using the **ssoadm** command after copy the files into place.

```
$ ssoadm \
  add-sub-schema \
  --adminid amadmin \
  --password-file /tmp/pwd.txt \
  --servicename sunIdentityRepositoryService \
  --schematype Organization \
  --filename customIdRepo.xml
```

Log in to the AM console as administrator, then browse to `Realms > Realm Name > Data Stores`. In the Data Stores table, click `New...` to create a Data Store corresponding to your custom IdRepo plugin. In the first screen of the wizard, name the Data Store and select the type corresponding to your plugin. In the second screen of the wizard, add the configuration for your plugin.

After creating the Data Store, create a new subject in the realm to check that your plugin works as expected. You can do this under `Realms > Realm Name > Subjects`.

If your plugin supports authentication, then users should now be able to authenticate using the `DataStore` module for the realm, by using a URL similar to the following:

```
http://openam.example.com:8080/openam/XUI/?realm=/myrealm#login&module=DataStore
```

## Chapter 4

# Setting Up Policy Agent Profiles

You install policy agents in web servers and web application containers to enforce access policies OpenAM applies to protected web sites and web applications. Policy agents depend on OpenAM for all authentication and authorization decisions. Their primary responsibility consists of enforcing what OpenAM decides in a way that is unobtrusive to the user. In organizations with many servers, you might well install many policy agents.

Policy agents can have local configurations where they are installed. Typically, you store all policy agent configuration information in the OpenAM configuration store, defining policy agent profiles for each, and then you let the policy agents access their profiles through OpenAM. In this way, you manage all agent configuration changes centrally. This chapter describes how to set up policy agent profiles in OpenAM for centralized configuration.

## 4.1. OpenIG or Policy Agent?

OpenAM supports both [OpenIG](#) and also a variety of policy agents. OpenIG and the policy agents can both enforce policy, redirecting users to authenticate when necessary, and controlling access to protected resources. OpenIG runs as a self-contained reverse proxy located between the users and the protected applications. Policy agents are installed into the servers where applications run, intercepting requests in that context.

Use OpenIG to protect access to applications not suited for a policy agent. Not all web servers and Java EE applications have policy agents. Not all operating systems work with policy agents.

Policy agents have the advantage of sitting within your existing server infrastructure. Once you have agents installed into the servers with web applications or sites to protect, then you can manage their configurations centrally from OpenAM.

For organizations with both servers on which you can install policy agents and also applications that you must protect without touching the server, you can use policy agents on the former and OpenIG for the latter.

## 4.2. Types of Agent

You can configure a number of different types of agents.

Each agent type requires an *agent profile* in OpenAM. The agent profile contains essential configuration for agent operation, such as a password to authenticate the agent, and the URL the

agent resides at. For agents that support it, the agent profile can store all agent configuration centrally, rather than locally on the agent server.

Web and J2EE policy agents are the most common, requiring the least integration effort. The available agent types are:

### **Web**

You install web agents in web servers to protect web sites.

### **J2EE**

You install J2EE agents in web application containers to protect web applications.

## **2.2 Agents**

Version 2.2 web and Java EE policy agents hold their configuration locally, connecting to OpenAM with a username/password combination. This agent type is provided for backwards compatibility.

### **OAuth 2.0/OpenID Connect Client**

Register OAuth 2.0 and OpenID Connect clients using this type of profile.

### **Agent Authenticator**

The agent authenticator can read agent profiles by connecting to OpenAM with a user name, password combination, but unlike the agent profile administrator, cannot change agent configuration.

### **SOAP STS Agent**

Secure requests from a SOAP STS deployment to OpenAM using this type of agent profile.

## **4.3. Creating Agent Profiles**

This section concerns creating agent profiles, and creating groups that let agents inherit settings when you have many agents with nearly the same profile settings.

### *Procedure 4.1. To Create an Agent Profile*

To create a new Web or Java EE policy agent profile, you need to create a name and password for the agent. You also need the URLs to OpenAM and the application to protect:

1. Login to the AM console as an administrative user.
2. On the Realms menu of the AM console, select the realm in which the agent profile is to be managed.
3. Click the Agents link, click the tab page for the kind of agent profile you want to create, and then click the New button in the Agent table.

4. In the Name field, enter a name for the agent profile.
5. In the Password and Re-Enter Password fields, enter a password for the new agent profile.
6. Click **Local** or **Centralized** (Default) to determine where the agent properties are stored. If you select **Local**, the properties are stored on the server on which the agent is running. If you select **Centralized**, the properties are stored on the OpenAM server.
7. In the Server URL field, enter the URL to OpenAM. For example, `http://openam.example.com:8080/openam`.
8. In the Agent URL field, enter the primary URL of the web or application server protected by the policy agent. Note for web agents, an example URL would look like: `http://www.example.com:80`. For Java EE policy agents, an example URL must include the **agentapp** context: `http://www.example.com:8080/agentapp`.

### New Agent

\* Indicates required field

\* Name:

\* Password:

\* Re-Enter Password:

Configuration: ☐ Local ☒ Centralized

Where agent properties are stored. Local is the server on which the agent is running. Centralized is the OpenAM Server

\* Server URL:

protocol://host:port/deploymentUri e.g. http://opensso.sample.com:58080/opensso

\* Agent URL:

protocol://host:port e.g. http://agent1.sample.com:1234

9. Click Create. After creating the agent profile, you can click the link to the new profile to adjust and export the configuration.

### Procedure 4.2. To Create an Agent Profile Group and Inherit Settings

Agent profile groups let you set up multiple agents to inherit settings from the group. To create a new agent profile group, you need a name and the URL to the OpenAM server in which you store the profile:

1. Login to the AM console as an administrative user.
2. On the Realms menu of the AM console, Select the realm in which you manage agents.
3. Click the Agents link, click the tab page for the kind of agent group you want to create, and then in the Group table, click New.

After creating the group profile, you can click the link to the new group profile to fine-tune or export the configuration.



4. Inherit group settings by selecting your agent profile, and then selecting the group name in the Group drop-down list near the top of the profile page.

You can then adjust inheritance by clicking Inheritance Settings on the agent profile page.

### Procedure 4.3. To Create an Agent Profile Using the Command Line

You can create a policy agent profile in OpenAM using the **ssoadm** command-line tool. You do so by specifying the agent properties either as a list of attributes, or by using an agent properties file as shown below. Export an existing policy agent configuration before you start to see what properties you want to set when creating the agent profile.

The following procedure demonstrates creating a policy agent profile using the **ssoadm** command:

1. Make sure the **ssoadm** command is installed. See Section 2.3.1, "Setting up Administration Tools" in the *Installation Guide*.
2. Determine the list of properties to set in the agent profile.

The following properties file shows a minimal configuration for a policy agent profile:

```
$ cat myAgent.properties
com.ipplanet.am.server.port=8443
com.sun.identity.agents.config.agenturi.prefix=http://www.example.com:80/amagent
com.sun.identity.agents.config.cdsso.cdcervlet.url[0]= \
https://openam.example.com:8443/openam/cdcervlet
com.sun.identity.agents.config.fqdn.default=www.example.com
com.sun.identity.agents.config.login.url[0]= \
http://openam.example.com:8443/openam/XUI/#login
com.sun.identity.agents.config.logout.url[0]= \
http://openam.example.com:8443/openam/XUI/#logout
com.sun.identity.agents.config.remote.logfile=amAgent_www_example_com_80.log
com.sun.identity.agents.config.repository.location=centralized
com.sun.identity.client.notification.url= \
http://www.example.com:80/UpdateAgentCacheServlet?shortcircuit=false
sunIdentityServerDeviceKeyValue[0]=agentRootURL=http://www.example.com:80/
sunIdentityServerDeviceStatus=Active
userpassword=password
```

3. Create a password file, for example `$HOME/.pwd.txt`. The file should only contain the password string, on a single line.

The password file must be read-only for the user who creates the policy agent profile, and must not be accessible to other users:

```
$ chmod 400 $HOME/.pwd.txt
```

4. Create the profile in OpenAM:

```
$ ssoadm create-agent \
--realm / \
--agentname myAgent \
--agenttype J2EEAgent \
--adminid amadmin \
--password-file $HOME/.pwd.txt \
--datafile myAgent.properties

Agent configuration was created.
```

At this point you can view the profile in the AM console under Realms > *Realm Name* > Agents to make sure the configuration is what you expect.

## 4.4. Delegating Agent Profile Creation

If you want to create policy agent profiles when installing policy agents, then you need the credentials of an OpenAM user who can read and write agent profiles.

You can use the OpenAM administrator account when creating policy agent profiles. If you delegate policy agent installation, then you might not want to share OpenAM administrator credentials with everyone who installs policy agents.

### *Procedure 4.4. To Create Agent Administrators for a Realm*

Follow these steps to create *agent administrator* users for a realm:

1. In the AM console, browse to Realms > *Realm Name* > Subjects.
2. Under Group click New... and create a group for agent administrators.
3. Switch to the Privileges tab for the realm, and click the name of the group you created.
4. Select Read and write access to all configured agents, and then Save your work.
5. Return to the Subjects tab, and under User create as many agent administrator users as needed.
6. For each agent administrator user, edit the user profile.

Under the Group tab of the user profile, add the user to agent profile administrator group, and then Save your work.

7. Provide each system administrator who installs policy agents with their agent administrator credentials.

When installing the policy agent with the `--custom-install` option, the system administrator can choose the option to create the profile during installation, and then provide the agent administrator user name and the path to a read-only file containing the agent administrator

password. For silent installs, you can add the `--acceptLicense` option to auto-accept the software license agreement.

## 4.5. Configuring Web Policy Agent Properties

When you create a web policy agent profile and install the agent, you can choose to store the agent configuration centrally and configure the agent through the AM console. Alternatively, you can choose to store the agent configuration locally and configure the agent by changing values in the properties file. For information on the properties used in a centralized configuration, and the corresponding properties for use in a local configuration file where applicable, see *Configuring Web Policy Agent Properties* in the *OpenAM Web Policy Agent Guide*.

## 4.6. Configuring Java EE Policy Agents

When you create a Java EE policy agent profile and install the agent, you can choose to store the agent configuration centrally and configure the agent through the AM console. Alternatively, you can store the agent configuration locally and configure the agent by changing values in the properties file. For information on the properties used in a centralized configuration, and the corresponding properties for use in a local configuration file where applicable, see *Creating Agent Profiles* in the *OpenAM Java EE Policy Agent Guide*.

## 4.7. Configuring Version 2.2 Policy Agents

This section covers version 2.2 policy agent properties. Version 2.2 agents store their configurations locally with a username-password combination used to connect to OpenAM.

### Warning

ForgeRock no longer supports 2.2 policy agents. Documentation exists only for legacy systems. Do not use 2.2 policy agents for new deployments.

After creating the agent profile, you access agent properties in the AM console under Realms > *Realm Name* > Applications > Agents > 2.2 Agents > *Agent Name*. Properties include:

### Password

Specifies the password the agent uses to connect to OpenAM.

### Status

Specifies whether the agent profile is active, and so can be used.

### Description

Specifies a short description for the agent.

### Agent Key Value(s)

Additional key-value pairs that OpenAM uses to receive agent requests concerning credential assertions.

OpenAM currently supports one property, `agentRootURL=protocol://host:port/` where the key is case-sensitive.

## 4.8. OAuth 2.0 and OpenID Connect 1.0 Client Settings

To register an OAuth 2.0 client with OpenAM as the OAuth 2.0 authorization server, or register an OpenID Connect 1.0 client through The AM console, then create an OAuth 2.0 Client agent profile. After creating the agent profile, you can further configure the properties in the AM console by navigating to *Realms > Realm Name > Applications > OAuth 2.0 > Client Name*.

### *OAuth 2.0 and OpenID Connect 1.0 Client Configuration Fields*

The following configuration fields are for OAuth 2.0 and OpenID Connect 1.0:

#### **Group**

Set this field if you have configured an OAuth 2.0 Client agent group.

#### **Status**

Specify whether the client profile is active for use or inactive.

#### **Client password**

Specify the client password as described by RFC 6749 in the section, [Client Password](#).

#### **Client type**

Specify the client type.

*Confidential* clients can maintain the confidentiality of their credentials, such as a web application running on a server where its credentials are protected. *Public* clients run the risk of exposing their passwords to a host or user agent, such as a JavaScript client running in a browser.

#### **Redirection URIs**

Specify client redirection endpoint URIs as described by RFC 6749 in the section, [Redirection Endpoint](#). OpenAM's OAuth 2.0 authorization service redirects the resource owner's user-agent back to this endpoint during the authorization code grant process. If your client has more than one redirection URI, then it must specify the redirection URI to use in the authorization request. The redirection URI must NOT contain a fragment (#).

Redirection URIs are required for OpenID Connect 1.0 clients.

## Scopes

Specify scopes that are to be presented to the resource owner when the resource owner is asked to authorize client access to protected resources.

Scopes can be entered as simple strings, such as `read`, `email`, `profile`, or `openid`, or as a pipe-separated string in the format: `scope|locale|localized description`. For example, `read|en|Permission to view email messages`.

*Locale* strings have the format: `language_country_variant`. For example, `en`, `en_GB`, or `en_US_WIN`. If the `locale` and pipe is omitted, the *localized description* is displayed to all users having undefined locales. If the *localized description* is omitted, nothing is displayed to all users. For example, a scope of `read|` would allow the client to use the `read` scope but would not display it to the user when requested.

## Claim(s)

Specify one or more claim name translations that will override those specified for the authentication session. Claims are values that are presented to the user to inform them what data is being made available to the client.

Claims can be entered as simple strings, such as `name`, `email`, `profile`, or `sub`, or as a pipe-separated string in the format: `scope|locale|localized description`. For example, `name|en|Full name of user`.

*Locale* strings have the format: `language_country_variant`. For example, `en`, `en_GB`, or `en_US_WIN`. If the `locale` and pipe is omitted, the *localized description* is displayed to all users having undefined locales. If the *localized description* is omitted, nothing is displayed to all users. For example, a claim of `name|` would allow the client to use the `name` claim but would not display it to the user when requested.

If a value is not given, the value is computed from the OAuth2 provider.

## Display name

Specify a client name to display to the resource owner when the resource owner is asked to authorize client access to protected resources. Valid formats include `name` or `locale|localized name`.

The Display name can be entered as a single string or as a pipe-separated string for locale and localized name, for example, `en|My Example Company`.

*Locale* strings have the format: `language_country_variant`. For example, `en`, `en_GB`, or `en_US_WIN`. If the `locale` is omitted, the name is displayed to all users having undefined locales.

## Display description

Specify a client description to display to the resource owner when the resource owner is asked to authorize client access to protected resources. Valid formats include `description` or `locale|localized description`.

The Display description can be entered as a single string or as a pipe-separated string for locale and localized name, for example, `en|The company intranet is requesting the following access permission`.

*Locale* strings have the format: *language\_country\_variant*. For example, `en`, `en_GB`, or `en_US_WIN`. If the `locale` is omitted, the name is displayed to all users having undefined locales.

## Default Scope(s)

Specify scopes in `scope` or `scope|locale|localized description` format. These scopes are set automatically when tokens are issued.

Default scopes can be entered as simple strings, such as `read`, `email`, `profile`, or `openid`, or as a pipe-separated string in the format: `scope|locale|localized description`. For example, `read|en|Permission to view email messages`.

*Locale* strings have the format: *language\_country\_variant*. For example, `en`, `en_GB`, or `en_US_WIN`. If the `locale` and pipe is omitted, the *localized description* is displayed to all users having undefined locales. If the *localized description* is omitted, nothing is displayed to all users. For example, a scope of `read|` would allow the client to use the `read` scope but would not display it to the user when requested.

## Response Types

Specify the response type that the client uses. The response type value specifies the flow that determine how the ID token and access token are returned to the client. For more information, see [OAuth 2.0 Multiple Response Type Encoding Practices](#).

By default, the following response types are available:

- `code`. Specifies that the client application requests an authorization code grant.
- `token`. Specifies that the client application requests an implicit grant type and requests a token from the API.
- `id_token`. Specifies that the client application requests an ID token.
- `code token`. Specifies that the client application requests an access token, access token type, and an authorization code.
- `token id_token`. Specifies that the client application requests an access token, access token type, and an ID token.
- `code id_token`. Specifies that the client application requests an authorization code and an ID token.
- `code token id_token`. Specifies that the client application requests an authorization code, access token, access token type, and an ID token.

## Contacts

Specify the email addresses of users who administer the client.

## Token Endpoint Authentication Method

Specify the authentication method with which a client authenticates to OpenAM (as an authorization server) at the token endpoint. The authentication method applies to OIDC requests

with scope `openid`. For more information, see the OpenID Connect Core 1.0 incorporating errata set 1.

- `client_secret_basic`. Clients authenticate with OpenAM (as an authorization server) using the HTTP Basic authentication scheme after receiving a `client_secret` value.
- `client_secret_post`. Clients authenticate with OpenAM (as an authorization server) by including the client credentials in the request body after receiving a `client_secret` value.
- `private_key_jwt`. Clients sign a JSON web token (JWT) with a registered public key.

## Json Web Key URI

Specify the URI that contains the client's public keys in JSON web key format.

## Json Web Key

Raw JSON web key value containing the client's public keys.

## Sector Identifier URI

Specify the host component of this URI, which is used in the computation of pairwise subject identifiers.

## Subject Type

Specify the subject identifier type, which is a locally unique identifier that will be consumed by the client. Select one of two options:

- `public`. Provides the same `sub` (subject) value to all clients.
- `pairwise`. Provides a different `sub` (subject) value to each client.

## ID Token Signing Algorithm

Specify the signing algorithm that the ID token must be signed with.

## Enable ID Token Encryption

Enable ID token encryption using the specified ID token encryption algorithm.

## ID Token Encryption Algorithm

Specify the algorithm that the ID token must be encrypted with.

Default value: `RSA1_5` (RSAES-PKCS1-V1\_5).

## ID Token Encryption Method

Specify the method that the ID token must be encrypted with.

Default value: `A128CBC-HS256`.

## Client ID Token Public Encryption Key

Specify the Base64-encoded public key for encrypting ID tokens.

## Post Logout Redirect URIs

Specify the URI to which to redirect the user-agent after the client logout process.

## Access Token

Specify the `registration_access_token` value that you provide when registering the client, and then subsequently when reading or updating the client profile.

## Client Session URI

Specify the relying party (client) URI to which the OpenID Connect Provider sends session changed notification messages using the HTML 5 `postMessage` API.

## Client Name

Specify a human-readable name for the client.

## Client JWT Bearer Public Key Certificate

Specify the public key certificate of the client's key pair that is used to sign JWTs issued by the client and used for client authentication or to request access tokens.

This is the base64-encoded X509 certificate containing the public key in PEM format, as in the following example.

```
-----BEGIN CERTIFICATE-----
MIIDETCCAfmAwIBAgIEU8SXLjANBgkqhkiG9w0BAQsFADA5MRswGQYDVQQKEExJvcGVuYW0uZXhh
bXBsZS5jb20xGjAYBgNVBAMTEWp3dC1iZWZyZXItY2xpZW50MB4XDTE0MTAyNzExNTY1NloXDTI0
MTAyNDExNTY1Nlo0TEBMBkGA1UEChMsY21aYHVSvELsWyMa7DjLD+mnjaF8cPRRMkhYZFXDJo/AVc
jyblyT3ntqL+2Js3D7TmS6BSjkkxZwsJHyhJIYEOUwWloc0kizgSm15MwBMcbnksQVN5VWi0e4y4JMbi30t6k38LM62K
KtaSPp6jvnnWLLTmL9uiqLwz54AM6hU3NLCI3J6Rfh8waBIPAEjmHZNqu0L2uGgWumzubYDFJbomL
SQq058RuKVaSVMwDbmEntMYWIKQL2xTt5XAbwEQEgJ/zskwpA2aQt1HE6de3Uym0h0NhRiu4rk3
A1EnEVbxrvy4Ik+wXg7LZVsCawEAAAMhMB8wHQYDVR00BBYEFiU7ejuZTg5tJsh1XyRopG0MBcs
MA0GCSqGSIb3DQEBChUAA4IBAQBm+/tYYVIS6LvPl3mfE8V7x+VPXqj/uK6UecAbfmRTRPk1ph+
jjiI6nmLX9ncomYALWL/JFiSxVsZt3/412f0qjakFVS0PmK1vEPxdlav1drnVA33icy1w0RRRu5/
qA6mwDYPAZSbm5cDVvCR7Lt6vqJ+D0V8GABFwUw9IaX6ajTqkWhldY77usvNeTD0Xc4R70qSBrNA
SNCAulJogWyzhbFlmE9Ne28j4Rvpbz/EZn0oc/cHTJ6Lryzsvf4uD01m3M3kM/MUYXc1Zv3rqBj
TeGSgcqEAd6XLGXy1M/yIeouUTI0F1bk1rNlqjvd57Xb4CEq17tVbGBm0hkECM8
-----END CERTIFICATE-----
```

You can generate a key pair and export the certificate by using the Java **keytool** command.

```
$ keytool \
  -genkeypair \
  -keysize 2048 \
  -alias self-signed \
  -keyalg rsa \
  -dname "CN=jwt-bearer-client,0=openam.example.com" \
  -keystore keystore.jks \
  -keypass changeit \
  -storepass changeit \
  -validity 3650 \
  -v
Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA)
```



```
with a validity of 3,650 days
for: CN=jwt-bearer-client, O=openam.example.com
[Storing keystore.jks]

$ keytool \
  -list \
  -alias self-signed \
  -rfc \
  -keystore keystore.jks \
  -keypass changeit \
  -storepass changeit
Alias name: self-signed
Creation date: Oct 27, 2014
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
-----BEGIN CERTIFICATE-----
MIIDETCCAfmGAWIBAgIEU8SXLjANBgkqhkiG9w0BAQsFADA5MRswGQYDVQQKEExJvcGVuYW0uZXhh
bXBsZS5jb20xGjAYBgNVBAMTEWp3dC1iZWZyZXItY2xpZW50MB4XDTE0MTAyNzExNTY1NloXDTE0
MTAyNDExNTY1Nlo0TEBMBkGA1UEChMsY3B1bmFtLmV4YW1wbGUuY29tMR0wGAYDVQQDExFqd3Qt
YmVhcmV4LWNaWVudDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAlD4ZZ/DIGEBR4QC
2uz0GYF0CULAPanxX21aYHSvELsWyMa7DJLD+mnjaF8cPRRMkhYZFXDJo/AVcjyblYt3ntqL+2Js
3D7TmS6BSjKxZwSJHyhJIYEoUwwLoc0kizgSm15MwBMcbnksQVN5VWioe4y4JMb130t6k38lM62K
KtaSPP6jvnW1LTmL9uiqLWz54AM6hU3NLCI3J6Rfh8waBIPAEjmHZNqu0L2uGgWumzubYDFJbomL
SQq058RuKVaSVMwDbmEntMYWIKQL2xTt5XAbwEQEgJ/zskwpA2aQt1HE6de3Uym0h0NhRiu4rk3
AIEEnEVbxrvy4Ik+wXg7LZVsCAwEAAMhMB8wHQYDVR00BBYEFiU7ejuZTg5tJsh1XyRopG0MBcs
MA0GCSqGSIb3DQEBwUAA4IBAQBm+/tYYVIS6LvPL3mfE8V7x+VPXqj/uK6UecAbfmRTrPk1ph+
jjI6nmLX9ncomYALWL/JfISxcVsZt3/412f0qjakFVS0PmK1vEPxDlav1drnVA33icylwORRRu5/
qA6mwDYPAZSbm5cDvCR7Lt6VqJ+D0V8GABFxUw9IaX6ajTqkWhldY77usvNeTD0Xc4R70qSBrnA
SNcaULJogWyzhbFlmE9Ne28j4RVPbz/EZn0oc/cHTJ6Lryzsisvf4uD01m3M3kM/MUyXc1Zv3rqBj
TeGSgcqEAd6XLGXy1+M/
yIeouUTi0F1bk1rNlqJvd57Xb4CEq17tVbGBm0hkECM8
-----END CERTIFICATE-----
```

## Default Max Age

Specify the maximum time in seconds that a user can be authenticated. If the user last authenticated earlier than this value, then the user must be authenticated again. If specified, the request parameter `max_age` overrides this setting.

Minimum value: 1.

Default: 600

## Default Max Age Enabled

Enable the default max age feature.

## Public key selector

Select the public key for this client, which comes from either the `JWks_URI`, manual JWKS, or X.509 field.

## Authorization Code Lifetime (seconds)

Specify the time in seconds for an authorization code to be valid. If this field is set to zero, the authorization code lifetime of the OAuth2 provider is used.

Default: 6000

### Refresh Token Lifetime (seconds)

Specify the time in seconds for a refresh token to be valid. If this field is set to zero, the refresh token lifetime of the OAuth2 provider is used. If the field is set to **-1**, the token will never expire.

Default: 6000

### Access Token Lifetime (seconds)

Specify the time in seconds for an access token to be valid. If this field is set to zero, the access token lifetime of the OAuth2 provider is used.

Default: 6000

### OpenID Connect JWT Token Lifetime (seconds)

Specify the time in seconds for a JWT to be valid. If this field is set to zero, the JWT token lifetime of the OAuth2 provider is used.

Default: 6000

### Implied Consent

Enable the implied consent feature. When enabled, the resource owner will not be asked for consent during authorization flows. The OAuth2 Provider must also be configured to allow clients to skip consent.

### JWKS URI content cache timeout in ms

Specify the maximum amount of time, in milliseconds, that the content of the JWKS URI can be cached before being refreshed. This avoids fetching the JWKS URI content for every token encryption.

Default: **3600000**

### JWKS URI content cache miss cache time

Specify the minimum amount of time, in milliseconds, that the content of the JWKS URI is cached. This avoids fetching the JWKS URI content for every token signature verification, for example if the key ID (**kid**) is not in the JWKS content already cached.

Default: **60000**

### User info signed response algorithm

Specify the JSON Web Signature (JWS) algorithm for signing UserInfo Responses. If specified, the response will be JSON Web Token (JWT) serialized, and signed using JWS.

The default, if omitted, is for the UserInfo Response to return the claims as a UTF-8-encoded JSON object using the **application/json** content type.

### User info encrypted response algorithm

Specify the JSON Web Encryption (JWE) algorithm for encrypting UserInfo Responses.

If both signing and encryption are requested, the response will be signed then encrypted, with the result being a nested JWT.

The default, if omitted, is that no encryption is performed.

### User info encrypted response encryption algorithm

Specify the JWE encryption method for encrypting UserInfo Responses. If specified, you must also specify an encryption algorithm in the *User info encrypted response algorithm* property.

AM supports the following encryption methods:

- **A128GCM**, **A192GCM**, and **A256GCM** - AES in Galois Counter Mode (GCM) authenticated encryption mode.
- **A128CBC-HS256**, **A192CBC-HS384**, and **A256CBC-HS512** - AES encryption in CBC mode, with HMAC-SHA-2 for integrity.

Default: **A128CBC-HS256**

### User info response format

Specify the output format from the UserInfo endpoint.

The supported output formats are as follows:

- User info JSON response format.
- User info encrypted JWT response format.
- User info signed JWT response format.
- User info signed then encrypted response format.

For more information on the output format of the UserInfo Response, see *Successful UserInfo Response* in the *OpenID Connect Core 1.0 incorporating errata set 1* specification.

Default: User info JSON response format.

### Token Endpoint Authentication Signing Algorithm

Specify the JWS algorithm that must be used for signing JWTs used to authenticate the client at the Token Endpoint.

JWTs that are *not* signed with the selected algorithm in token requests from the client using the **private\_key\_jwt** or **client\_secret\_jwt** authentication methods will be rejected.

Default: **RS256**

### OAuth 2.0 Mix-Up Mitigation enabled

Enable OAuth 2.0 mix-up mitigation on the authorization server side.

Enable this setting only if this OAuth 2.0 client supports the OAuth 2.0 Mix-Up Mitigation draft, otherwise AM will fail to validate access token requests received from this client.

## 4.9. Configuring Agent Authenticators

An *agent authenticator* has read-only access to multiple agent profiles defined in the same realm, typically allowing an agent to read web service agent profiles.

After creating the agent profile, you access agent properties in the AM console under Realms > *Realm Name* > Applications > Agents > Agent Authenticator > *Agent Name*.

### Password

Specifies the password the agent uses to connect to OpenAM.

### Status

Specifies whether the agent profile is active, and so can be used.

### Agent Profiles allowed to Read

Specifies which agent profiles in the realm the agent authenticator can read.

### Agent Root URL for CDSSO

Specifies the list of agent root URLs for CDSSO. The valid value is in the format *protocol://hostname:port/* where *protocol* represents the protocol used, such as *http* or *https*, *hostname* represents the host name of the system where the agent resides, and *port* represents the port number on which the agent is installed. The slash following the port number is required.

If your agent system also has virtual host names, add URLs with the virtual host names to this list as well. OpenAM checks that *goto* URLs match one of the agent root URLs for CDSSO.

## 4.10. Configuring SOAP STS Agents

A SOAP STS deployment accesses OpenAM using a SOAP STS agent.

After creating the agent profile, you access agent properties in the AM console under Realms > *Realm Name* > Applications > Agents > SOAP STS Agent > *Agent Name*. For information on the available properties, see Section 2.1.2.1, "Creating a SOAP STS Agent" in the *Security Token Service Guide*.

## Chapter 5

# Setting Up Keys and Keystores

This chapter shows how to work with keys and keystores, which are used to protect network communication and to sign and encrypt information.

## 5.1. Introducing Keystores

A keystore is a file that serves as a repository of certificates and keys used in SSL/TSL encryption. OpenAM uses them for two purposes: To store keys and certificates used for federation, token signatures, and others, and to store the passwords required during OpenAM's startup process.

Most features that require storing key aliases for signing or encryption use OpenAM's keystore, which is configured by navigating to **Configure > Server Defaults > Security > Key Store**. However, some features may require or support different configurations:

**Features that only use OpenAM's keystore configuration files:**

- **SAML v1.x**

Requires one key pair alias for signing XML files. For more information, see [Section 2.2, "Preparing To Secure SAML v1.x Communications"](#) in the *SAML v1.x Guide*.

- **User self-service**

Requires a JCEKS keystore with a key pair alias for encryption and a key alias for signing. For more information, see [Section 2.1, "Configuring the Signing and Encryption Key Aliases"](#) in the *User Self Service Guide*.

- **Persistent Cookie module**

Requires a key pair alias for encryption. For more information, see [Section 2.2.20, "Persistent Cookie Module"](#) in the *Authentication and Single Sign-On Guide*.

- **Stateless session cookies**

Require an RSA key pair alias for encryption and an RSA/ECDSA key pair alias for signing. For more information, see [Section 6.1.3, "Configuring Stateless Session Cookie Security"](#) in the *Authentication and Single Sign-On Guide*.

- **Java Fedlets**

Require a key pair to sign and verify XML assertions and to encrypt and decrypt SAML assertions. For more information, see Section 3.2.2, "Configuring Java Fedlet Properties" in the *SAML v2.0 Guide*.

- **OAuth2 and OIDC providers**

Require configuring RSA or ECDSA key aliases for signing tokens. For more information, see Section 2.6, "Configuring Digital Signatures" in the *OAuth 2.0 Guide*.

- **Amster**

Requires a `sms.transport.key` key alias to export and import encrypted passwords. For more information, see the *Amster Command-line Interface Guide*.

### Features that support different keystore configurations:

- **ForgeRock Authenticator (OATH) and ForgeRock Authenticator (PUSH) modules**

Support configuring a different keystore to encrypt device profiles. They also support different keystore types that are not available to other features. For more information, see Chapter 4, "Implementing Multi-Factor Authentication" in the *Authentication and Single Sign-On Guide*.

- **SAML v2.0 providers**

Require key pairs in OpenAM's keystore to configure signing or encryption. However, they support setting up a specific password for the key pair instead of using OpenAM's keystore password file. For more information, see Section 5.2, "SAML v2.0 Configuration Properties" in the *SAML v2.0 Guide*.

- **OpenAM's startup process**

Requires a JCEKS keystore to store the password of the directory manager user and the special `dsameuser` user. For more information about OpenAM's startup process, see Section 2.4, "Starting Servers" in the *Installation Guide*.

The startup process supports configuring a different JCEKS keystore, and its configuration is stored in the `/path/to/openam/boot.json` file. This file and the entries in the keystore for the directory manager and `dsameuser` users are created during installation and upgrade processes. For more information about possible keystore configurations after an upgrade, see Section 5.1.1, "Keystore Configuration After Upgrade".

### Features that require different keystore configurations:

- **Security Token Service**

Requires configuring a JKS keystore for encrypting SAML v2.0 and OpenID Connect tokens. It does not require files to store the keystore password or the key aliases' passwords. For more information, see Section 1.4, "About the STS" in the *Security Token Service Guide*.

## • CSV audit logging handler

Requires configuring a JKS keystore file called `Logger.jks` for tamper-proofing. It does not require a file to store the keystore password; the password is configured in the AM console. For more information, see Section 6.2.2.2, "Configuring CSV Audit Event Handlers".

OpenAM supports two keystore types: JCEKS, configured by default in new installations, and JKS. During installation, OpenAM deploys a keystore of each type with several self-signed key aliases for demo and test purposes only. For production deployments, you should generate your own key aliases and configure OpenAM to use them.

For a comparison between the default configuration of the JCEKS and the JKS keystores in OpenAM, see the following table:

*Table 5.1. JCEKS and JKS Keystore Comparison*

	JCEKS	JKS
<b>Used by default in OpenAM?</b>	Yes <sup>a</sup>	No
<b>In which path is it?</b>	<code>/path/to/openam/openam/keystore.jceks</code>	<code>/path/to/openam/openam/keystore.jks</code>
<b>Where is its password stored?</b> <sup>b</sup>	<code>/path/to/openam/openam/.storepass</code>	<code>/path/to/openam/openam/.storepass</code>
<b>Which test aliases does it contain?</b>	<code>es256test</code> <sup>c</sup> <code>es384test</code> <sup>c</sup> <code>es512test</code> <sup>c</sup> <code>selfserviceentest</code> <sup>d</sup> <code>selfservicesigntest</code> <sup>e</sup> <code>test</code> <sup>d</sup>	<code>test</code> <sup>d</sup>
<b>Which password strings does it contain?</b>	<code>configstorepwd</code> <sup>f</sup> <code>dsamesuserpwd</code> <sup>g</sup>	None
<b>Where is the private key password file?</b> <sup>h</sup>	<code>/path/to/openam/openam/.keypass</code>	<code>/path/to/openam/openam/.keypass</code>

<sup>a</sup> New OpenAM installations use the JCEKS keystore as the default keystore. For more information about upgraded configurations, see Section 5.1.1, "Keystore Configuration After Upgrade".

<sup>b</sup> The password of the JCEKS and JKS keystores is a random-generated string stored in cleartext. This string is also the password of the `configstorepwd` and `dsamesuserpwd` password strings.

<sup>c</sup> ECDSA key.

<sup>d</sup> Asymmetric RSA key.

<sup>e</sup> Symmetric secret signing key.

<sup>f</sup> The value of the `configstorepwd` is a string. It is the password of the configuration store, which is accessed at OpenAM's startup time.

<sup>g</sup> The value of the `dsamesuserpwd` is a string. It is the password of an special user which is accessed at OpenAM's startup time.

<sup>h</sup> The password for all the test key aliases in the JCEKS and JKS keystores is `changeit`, stored in cleartext.

### 5.1.1. Keystore Configuration After Upgrade

After an upgrade to AM 5, keystores are deployed and configured depending on the circumstances at the time of the upgrade:

- **Upgrading from OpenAM 13 or earlier**

A `keystore.jceks` file is deployed with the configuration described in Table 5.1, "JCEKS and JKS Keystore Comparison". This keystore is used by OpenAM's startup process. OpenAM's keystore is still JKS, and contains the key aliases it had before the upgrade.

- **Upgrading from OpenAM 13.5**

- If OpenAM 13.5 had a JCEKS keystore configured in Configure > Server Defaults > Security > Key Store at the time of the upgrade, the password strings described in Table 5.1, "JCEKS and JKS Keystore Comparison" are appended to the existing JCEKS keystore.
- If OpenAM 13.5 had a JKS keystore configured in Configure > Server Defaults > Security > Key Store at the time of the upgrade, the behavior is the same as noted in *Upgrading from OpenAM 13 or earlier*.

#### Important

The upgrade does not convert keys from an old keystore to a new one. For example, if you upgrade from OpenAM 13 or earlier and you want to use the user self-service features, you need to configure a JCEKS keystore with suitable keys for user self-service and any other that might be configured in the JKS keystore.

## 5.2. Configuring Keystores

OpenAM provides a JCEKS keystore by default in new installations. If you upgraded from OpenAM 13 or earlier versions, OpenAM uses a JKS keystore by default unless you reconfigured OpenAM to use a JCEKS keystore.

When modifying the keystore in your setup, consider the following points:

- Key aliases are not migrated from one keystore to another when changing the keystore configuration in OpenAM. You must prepare the new keystore as required before configuring it. For more information about creating a new keystore and new keys, see the Section 5.3, "Configuring Key Aliases".
- In a multi-server environment, every server has its own keystore files. Make sure keystores, key aliases and certificates are maintained on every server.
- You must restart OpenAM if you make any changes to the keystore. Changes are, for example, adding or removing keys or changing key or keystore passwords.
- The keystore used for the OpenAM's startup process must contain the `configstorepwd` and the `dsameuserpwd` password strings. Failure to do so will render OpenAM unbootable. For more



information about configuring keystores for OpenAM's startup process, see Section 2.4, "Starting Servers" in the *Installation Guide*.

### *Procedure 5.1. To Configure Keystore Properties*

To configure OpenAM to use a JCEKS or a JKS keystore, or to modify OpenAM keystore configuration, perform the following steps:

1. Determine whether you want to configure the keystore for all your servers or configure the keystore for a single server:
  - To configure the keystore for all your servers, navigate to Configure > Server Defaults > Security > Key Store.
  - To configure the keystore on a single server, navigate to Deployment > Servers > *Server Name* > Security > Key Store.

For more information about inherited properties, see Section 2.3.1, "Configuring Servers" in the *Reference*.

2. Enter the keystore file name and path in the Keystore File field.
3. Set the Keystore Type to **JKS** or **JCEKS**.
4. In the Keystore Password File field, enter the location of the keystore password file.
5. In the Private Key Password File field, enter the location of the private key password file.
6. In the Certificate Alias field, enter the alias of the private key to sign SAML v1.x XML files. If you do not require SAML v1.x functionality, you can leave the default **test** alias.
7. Save your changes.

Figure 5.1. Security Key Store Tab

Security

Search

Encryption Validation Cookie **Key Store** Certificate Revocation List Caching

**Keystore File** %BASE\_DIR%/%SERVER\_URI%/keystore.jceks ⓘ

**Keystore Type** JCEKS ⓘ

**Keystore Password File** %BASE\_DIR%/%SERVER\_URI%/storepass ⓘ

**Private Key Password File** %BASE\_DIR%/%SERVER\_URI%/keypass ⓘ

**Certificate Alias** test ⓘ

Save Changes

8. (Optional) If you are using the same keystore file for the startup process, verify that the values configured in the Keystore File, Keystore Type, Keystore Password File, and Private Key Password File fields are configured in the `/path/to/openam/boot.json` file.

Note that a configuration of `%BASE_DIR%/%SERVER_URI%/keystore.jceks` in the AM console corresponds to the path `/path/to/openam/openam/keystore.jceks` in the `boot.json` file. For more information on how to configure the `boot.json` file, see Section 2.4, "Starting Servers" in the *Installation Guide*.

9. Ensure that password files and keystores are maintained on every server in your environment. Every OpenAM server has its own keystores and password files.
10. Restart the OpenAM server or servers affected by the configuration changes.

### 5.2.1. Changing the Keystore Password

Decrypting and viewing the contents of a keystore requires a password. This password is specified by the user at the time the keystore is created, but you might need to update the password at a later time.

The password contained in the `/path/to/openam/openam/.storepass` protects the keystore and the `configstorepwd` and `dsameuserpwd` password string aliases.

## Procedure 5.2. To Change the Keystore Password

1. Change directories to the keystore location, for example `/path/to/openam/openam`.
2. Back up the `/path/to/openam/openam/keystore.jceks` and `/path/to/openam/openam/.storepass` files.
3. Use the **keytool** command to change the password:

```
$ keytool -storepasswd -storetype JCEKS -keystore keystore.jceks
Enter keystore password:Enter the password in the .storepass file.
New keystore password:
Re-enter new keystore password:
```

4. Replace the old password in the `.storepass` file with the new one:

```
$ echo newpassword > .storepass
```

5. Ensure the `.storepass` file has read-only permissions for its owner:

```
$ chmod 400 .storepass
```

6. Use the **keytool** command to change the password of the `configstorepwd` and `dsameuserpwd` password string aliases, for example:

```
$ keytool -keypasswd -storetype JCEKS -keystore keystore.jceks -alias configstorepwd
Enter keystore password:
New key password for <configstorepwd> Enter the password in the .storepass file.
Re-enter new key password for <configstorepwd>
```

7. If you also need to change the key aliases' password, see Procedure 5.5, "To Change Key Aliases Passwords".
8. (Optional) If you are using this keystore in OpenAM's startup file, ensure the path to the `.storepass` file is configured in the `keyStorePasswordFile` JSON property.
9. Ensure that password files and keystores are maintained on every server in your environment. Every OpenAM server has its own keystores and password files.
10. Restart the OpenAM server or servers affected by the configuration changes.

## 5.3. Configuring Key Aliases

OpenAM deploys a JCEKS and a JKS keystore during installation that include several test key aliases ready to use for demo purposes. For more information about which keys are provided, see Table 5.1, "JCEKS and JKS Keystore Comparison".

When deleting or adding key aliases in your setup, consider the following points:

- There may be more than one key alias in the keystore. For example, you may have one key alias for SAML 2.0 configuration, two more key aliases for the user self-service features, and others.

- The `.keypass` file contains a password in cleartext that unlocks the key aliases contained in the OpenAM keystore. This file is shared among most of the features in OpenAM, but some can configure their own passwords. For a list of the features and their configurations, see [Section 5.1, "Introducing Keystores"](#).
- In a multi-server environment, every server has its own keystore files. Make sure keystores, key aliases and certificates are maintained on every server.
- You must restart OpenAM if you make any changes to the keystore. For example, adding or removing keys or changing key or keystore passwords.

The following table contains recommendations on which algorithm to use for some OpenAM features:

*Table 5.2. Recommended Algorithms to Create Key Aliases for Features*

Usage	Recommended Algorithm
User self-service encryption key	RSA with SHA-256, minimum 2048-bit
User self-service signing secret	HMAC with SHA-256
SAML v1.x	RSA with SHA-256, minimum 2048-bit
SAML v2.0	RSA with SHA-256, minimum 2048-bit
Persistent Cookie Encryption	RSA with SHA-256, minimum 2048-bit
Stateless Sessions	See <a href="#">Section 6.1.3, "Configuring Stateless Session Cookie Security"</a> in the <i>Authentication and Single Sign-On Guide</i>
ForgeRock Authenticator (OATH) and ForgeRock Authenticator (PUSH) modules	See <a href="#">Chapter 4, "Implementing Multi-Factor Authentication"</a> in the <i>Authentication and Single Sign-On Guide</i> .
OAuth2 and OIDC Providers	See <a href="#">Section 2.6, "Configuring Digital Signatures"</a> in the <i>OAuth 2.0 Guide</i>

### 5.3.1. Changing the Signing Key

By default, OpenAM uses the `test` key alias as follows:

- To sign persistent cookies, configured in [Realms > Realm Name > Authentication > Settings > Security > Persistent Cookie Encryption Certificate Alias](#).
- To sign SAML v1.x XML files, configured in [Configure > Server Defaults > Security > Key Store > Certificate Alias](#).
- To sign and encrypt stateless sessions, configured in [Configure > Global Configuration > Session](#).

Do not use the `test` key alias for production environments.

### Procedure 5.3. To Change Default test Signing Key

Perform the following steps to change the **test** key alias, configured by default, to another key:

1. Change directories to the keystore location, for example `/path/to/openam/openam`.
2. Back up the `/path/to/openam/openam/keystore.jceks`, `/path/to/openam/openam/.storepass`, and `/path/to/openam/openam/.keypass` files.
3. Acquire a new key from your certificate authority, or generate new self-signed keys. You can:
  - Generate a new key (self-signed or not) and add it to the existing keystore configured in OpenAM.
  - Generate a new key and a new keystore.
  - Import a key to a keystore.

When you create or import a new key, the **keytool** command adds the new alias to the specified keystore if it exists, or creates a new keystore if it does not exist.

This step uses self-signed keys as an example and creates a new keystore file, `keystore.jceks`, in a temporary location with a new asymmetric RSA key alias called **newkey**. If you want to create ECDSA keys, see Section 6.1.3.3, "Configuring Elliptic Curve Digital Signature Algorithms" in the *Authentication and Single Sign-On Guide*.

#### Important

If you decide to create a new keystore to store key aliases for OpenAM's features, ensure that the original `keystore.jceks`, `.storepass`, and `.keypass` files remain on your server and that OpenAM's startup process is configured to use these files. Failure to do so will render OpenAM unbootable.

The passwords entered in this step are used in the next step. Keep track of them:

```
$ cd /tmp
$ keytool \
  -genkeypair \
  -alias newkey \
  -keyalg RSA \
  -keysize 2048 \
  -validity 730 \
  -storetype JCEKS \
  -keystore keystore.jceks
Enter keystore password:
Reenter new password:
What is your first and last name?
[Unknown]: openam.example.com
What is the name of your organizational unit?
[Unknown]: Eng
What is the name of your organization?
[Unknown]: ForgeRock.com
What is the name of your City or Locality?
[Unknown]: Grenoble
What is the name of your State or Province?
[Unknown]: Isere
What is the two-letter country code for this unit?
[Unknown]: FR
Is CN=openam.example.com, OU=Eng, O=ForgeRock.com, L=Grenoble, ST=Isere,
C=FR correct?
[no]: yes

Enter key password for <newkey>
(RETURN if same as keystore password):
Reenter new password:
```

4. Create two files, each containing only the password in cleartext:
  - `.storepass` contains the cleartext keystore password.
  - `.keypass` contains the cleartext password for the key aliases that reside in the keystore.
5. Replace OpenAM's keystore and password files with the ones that you just created. For example:
 

```
$ cp keystore.jceks .storepass .keypass /path/to/openam/openam/
```
6. Make sure the password files have read-only permission for their owner. For example:
 

```
$ chmod 400 .storepass
$ chmod 400 .keypass
```
7. If you have changed the path or the name of any of the keystore files, perform the following steps:
  - a. Log in to the AM console as an administrative user, for example, `amadmin`.
  - b. Follow the steps in [Procedure 5.1, "To Configure Keystore Properties"](#) to change the keystore configuration in the AM console.
8. If you have an authentication chain configured with the Persistent Cookie module, perform the following steps:

- a. Log in to the AM console as an administrative user, for example, `amadmin`.
  - b. Navigate to Realms > *Realm Name* > Authentication > Settings > Security.
  - c. Change the value in the Persistent Cookie Encryption Certificate Alias field from `test` to `newkey`.
  - d. Save your changes.
9. Ensure that password files and keystores are maintained on every server in your environment. Every OpenAM server has its own keystores and password files.
  10. Restart the OpenAM servers affected by the configuration changes to use the new keystore and encrypted password files.
  11. Replace the `test` key alias as needed, for example:
    - a. If you have configured a SAML v2.0 identity provider, navigate to Realms > *Realm Name* > Applications > SAML > Entity Providers > *Provider Name* > Assertion Content > Signing and Encryption, and then edit the signing key certificate alias.
    - b. Navigate to Configure > Server Defaults > Security > Key Store and replace the `test` key alias in the Certificate Alias property for SAML v1.x usage.
  12. (Optional) Self-signed keys are not automatically recognized by other entities. If you created new self-signed key aliases, you must share them as described in Procedure 4.2, "To Share Self-Signed Certificates" in the *Installation Guide*.
  13. (Optional) Share updated metadata with other entities in your circle of trust as described in Section 2.3, "Configuring Identity Providers, Service Providers, and Circles of Trust" in the *SAML v2.0 Guide*.

### 5.3.2. Changing User Self-Service Key Aliases

OpenAM requires a JCEKS keystore to configure user self-service features. The JCEKS keystore deployed with OpenAM has two keys already configured (one for signing, one for encrypting) that can be used for testing or evaluation purposes. You should replace these keys in your production environment.

#### *Procedure 5.4. To Change Default User Self-Service Key Aliases*

User self-service requires a key pair for encryption and a signing secret key to be available before configuring any of its features. OpenAM provides the demo `selfserviceentest` key alias for encrypting, and the demo `selfservicesigntest` secret key alias for signing.

Follow the steps in this procedure to create new key aliases for the user self-service features without creating a new keystore. If you need to create a new keystore and replace the default `test` key alias as

well, see Procedure 5.3, "To Change Default test Signing Key" for an example before continuing with this procedure.

Perform the following steps:

1. Back up the `/path/to/openam/openam/keystore.jceks` file.
2. Acquire a new key from your certificate authority, or generate new self-signed keys. The password of the new keys for the user self-service features must match the passwords of those keys already present in the keystore and configured in the `/path/to/openam/openam/.keypass` file.

This example generates a self-signed key for encryption and a new signing secret key into an existing keystore, but you could also import CA-provided keys to the keystore.

- a. Create the new self-signed encryption key alias:

```
$ cd /path/to/openam/openam
$ keytool \
  -genkeypair \
  -alias newenckey \
  -keyalg RSA \
  -keysize 2048 \
  -validity 730 \
  -storetype JCEKS \
  -keystore keystore.jceks
Enter keystore password:
What is your first and last name?
[Unknown]: openam.example.com
What is the name of your organizational unit?
[Unknown]: Eng
What is the name of your organization?
[Unknown]: ForgeRock.com
What is the name of your City or Locality?
[Unknown]: Grenoble
What is the name of your State or Province?
[Unknown]: Isere
What is the two-letter country code for this unit?
[Unknown]: FR
Is CN=openam.example.com, OU=Eng, O=ForgeRock.com, L=Grenoble, ST=Isere,
C=FR correct?
[no]: yes

Enter key password for <newenckey>
(RETURN if same as keystore password): Enter the password in the .keypass file.
Re-enter new password:
```

- b. Create the new signing secret key alias:



```
$ cd /path/to/openam/openam
$ keytool \
  -genseckey \
  -alias newsigkey \
  -keyalg HmacSHA256 \
  -keysize 256 \
  -storetype JCEKS \
  -keystore keystore.jceks
Enter keystore password:
Enter key password for <newsigkey>
(RETURN if same as keystore password): Enter the password in the .keypass file.
Re-enter new password:
```

3. Ensure that password files and keystores are maintained on every server in your environment. Every OpenAM server has its own keystores and password files.
4. Restart the OpenAM servers affected by the configuration changes.
5. Configure user self-service to use the new keys. For more information, see Procedure 2.1, "To Configure Self Service Key Aliases" in the *User Self Service Guide*.

### 5.3.3. Changing the Key Aliases' Passwords

Decrypting a key alias in a keystore requires a password. This password is initially specified when you generate the key, or when you import the key into a keystore, but you might need to update the password at a later time.

#### *Procedure 5.5. To Change Key Aliases Passwords*

1. Change directories to the keystore location, for example `/path/to/openam/openam`.
2. Back up the `/path/to/openam/openam/keystore.jceks` and `/path/to/openam/openam/.keypass` files.
3. List all the keys and password strings contained in the keystore:

```
$ keytool -list -storetype JCEKS -keystore keystore.jceks

Enter keystore password:

Keystore type: JCEKS
Keystore provider: SunJCE

Your keystore contains 5 entries

configstorepwd, 10-Oct-2016, SecretKeyEntry,
dsameuserpwd, 10-Oct-2016, SecretKeyEntry,
selfserviceentest, 18-Mar-2016, PrivateKeyEntry,
Certificate fingerprint (SHA1): 72:3F:17:D9:8F:DF:53:E0:03:01:68:69:AB:5F:8C:F8:53:E4:F3:E8
test, 18-Mar-2016, PrivateKeyEntry,
Certificate fingerprint (SHA1): 84:C3:0D:2C:B3:D6:C9:41:B4:FC:B3:9D:0B:C6:8C:E3:F8:65:D9:B7
selfservicesigntest, 18-Mar-2016, SecretKeyEntry,
```

4. Most of the key aliases in OpenAM's keystore must have the same password. You need to change the passwords of all the key aliases in the keystore file to match the content of the `.keypass` file. For more information about features that may not use this password file, see Section 5.1, "Introducing Keystores".

Use the **keytool** command to change the password of each of the key aliases, for example:

```
$ keytool -keypasswd -storetype JCEKS -keystore keystore.jceks -alias test
Enter keystore password:
New key password for <test>
Re-enter new key password for <test>
```

#### Important

The `configstorepwd` and `dsameuserpwd` password string aliases do not use the `.keypass` file to store their password. If you need to change their password, see Section 5.2.1, "Changing the Keystore Password".

5. Replace the old password in the `.keypass` file with the new one:
 

```
$ echo newpassword > .keypass
```
6. Make sure the `.keypass` file has read-only permissions for its owner:
 

```
$ chmod 400 .keypass
```
7. If you also need to change the keystore password, see Procedure 5.2, "To Change the Keystore Password".
8. If you are using this password file in OpenAM's startup file, ensure the path to the `.keypass` file is configured in the `keyPasswordFile` JSON property.
9. (Optional) The name and path of the `.keypass` file is configured in the AM console for several features. If you have changed the name or the path to this file, you may need to adjust the configuration. For information about different configurations, see Section 5.1, "Introducing Keystores".
10. Ensure that password files and keystores are maintained on every server in your environment. Every OpenAM server has its own keystores and password files.
11. Restart the OpenAM servers affected by the configuration changes.

## 5.4. Configuring Password String Aliases

During the startup process, OpenAM needs the following two aliases inside the keystore configured in the `/path/to/openam/openam/boot.json` file:

### **configstorepwd**

Stores the password of the OpenAM configuration store. The password that protects this string is contained in the `/path/to/openam/openam/.storepass` file.

To update this password, navigate to Deployment > Servers > Server Name > Directory Configuration and modify the configuration store bind password. Every time you change the bind alias, OpenAM modifies the content of the key alias in the `keystore.jceks` file.

If you need to change the password manually or recreate the key alias, see Procedure 5.6, "To Recreate the configstorepwd Password String Alias".

### dsameuserpwd

Stores the password of a special user required at OpenAM startup time. The password that protects this string is contained in the `/path/to/openam/openam/.storepass` file.

The password string for `dsameuserpwd` cannot be recreated in a new keystore. If you need to create a new keystore to store key aliases for OpenAM features, keep the JCEKS keystore deployed by OpenAM during install or upgrade configured in the `/path/to/openam/openam/boot.json` file, and configure the new keystore by following Procedure 5.1, "To Configure Keystore Properties".

Unlike key aliases, which store certificates or signing keys, these aliases store password-protected strings that are used as passwords by OpenAM's startup process.

- For more information about OpenAM's startup process and the `/path/to/openam/openam/boot.json` file, see Section 2.4, "Starting Servers" in the *Installation Guide*.
- To manually recreate the `configstorepwd` password string alias in an existing keystore, see Procedure 5.6, "To Recreate the configstorepwd Password String Alias".
- To change the password that protect the strings, which is stored in the `.keypass` file and is shared with the key aliases inside the keystore, see Procedure 5.5, "To Change Key Aliases Passwords".

### Procedure 5.6. To Recreate the configstorepwd Password String Alias

OpenAM updates the content of the `configstorepwd` alias when the bind password of the configuration store is modified. Follow this procedure only if you need to change the content of the `configstorepwd` key alias manually:

1. Change directories to the keystore location, for example `/path/to/openam/openam`.
2. Back up the `/path/to/openam/openam/keystore.jceks`.
3. Delete the `configstorepwd` alias using the **keytool** command:

```
$ keytool -delete -storetype JCEKS -keystore keystore.jceks -alias configstorepwd
Enter the keystore password: Enter the password in the .storepass file.
```

4. Use the **keytool** command to create the new `configstorepwd` alias in the keystore:

```
$ keytool -importpassword -storetype JCEKS -keystore keystore.jceks -alias configstorepwd
Enter keystore password: Enter the password in the .storepass file.
Enter the password to be stored: See NOTE below.
Re-enter password:
Enter key password for <configstorepwd>
(RETURN if same as keystore password): Enter the password in the .storepass file.
Re-enter new password:
```

**Note**

The password to store in the `configstorepwd` alias is the password that is configured in Deployment > Servers > *Server Name* > Directory Configuration as the configuration store binding. That means you need to store the OpenDJ `cn=Directory Server` user's password.

- If the OpenAM installation uses the embedded OpenDJ instance as the configuration store, the password to store in the `configstorepwd` alias is the password of the `amadmin` user. For more information, see Procedure 8.6, "To Change the amadmin User's Password: Embedded Configuration Store".
- If the OpenAM installation uses an external configuration store, refer to the configuration store administrator for the password.

## Chapter 6

# Setting Up Audit Logging

## 6.1. Introducing the Audit Logging Service

OpenAM supports a comprehensive Audit Logging Service that captures key auditing events, critical for system security, troubleshooting, and regulatory compliance.

Audit logs gather operational information about events occurring within an OpenAM deployment to track processes and security data, such as authentication mechanisms, system access, user and administrator activity, error messages, and configuration changes.

This chapter describes the common REST-based Audit Logging Service available in AM 5. AM 5 also supports a legacy Logging Service, based on a Java SDK. The legacy Logging Service will be deprecated in a future release of OpenAM.

The Audit Logging Service uses a structured message format that adheres to a consistent log structure common across the ForgeRock Identity Platform. This common structure allows correlation between log messages of the different components of the Platform once the transaction IDs are trusted by enabling the ForgeRock trust transaction header system property.

For more information, see [Configuring the Trust Transaction Header System Property](#).

### Important

OpenDJ 4.0's JSON logger is enabled by default. However, the ForgeRock transaction IDs are not trusted initially. You must set `trust-transaction-ids:true` to correlate log messages in OpenDJ with log messages in OpenAM. For more information, see *To Enable LDAP JSON Access Logs in the OpenDJ Administration Guide*.

### 6.1.1. About the Audit Logging Service

OpenAM writes log messages generated from audit events triggered by its instances, policy agents, the **ssoadm** tool, and connected ForgeRock Identity Platform implementations.

OpenAM's Audit Logging Service provides a versatile and rich feature set as follows:

- **Global and Realm-Based Log Configuration.** You can configure audit logging globally, which ensures that all realms inherit your global log settings. You can also configure audit logging by realm, which allows you to set different log settings for each realm.
- **Audit Event Handlers.** The Audit Logging Service supports a variety of audit event handlers that allow you to write logs to different types of data stores. See Section 6.2.2, "Configuring Audit Event Handlers" for a list of event handlers available in AM 5.

- **Audit Event Buffering.** By default, OpenAM writes each log message separately as they are generated. OpenAM supports message buffering, a type of batch processing, that stores log messages in memory and flushes the buffer after a preconfigured time interval or after a certain number of log messages reaches the configured threshold value.
- **Tamper-Evident Logging.** You can digitally sign your audit logs to ensure no unauthorized tampering of your logs has taken place. To configure this feature, you must deploy a preconfigured logger certificate and store it at `/path/to/openam/openam/Logger.jks`.
- **Log Rotation and Retention Policies.** By default, OpenAM rotates its audit logs when it reaches a specified maximum size. You can also configure a time-based rotation policy, which disables the max-size rotation policy and implements log rotation based on a preconfigured time sequence. You also have the option to disable log rotation and use an external log rotation tool.
- **Blacklisting Sensitive Fields.** The Audit Logging Service supports blacklisting, a type of filtering to hide sensitive values or fields, such as HTTP headers, query parameters, cookies, or the entire field value.
- **Reverse DNS Lookup.** The Audit Logging Service supports a reverse DNS lookup feature for network troubleshooting purposes. Reverse DNS lookup is disabled by default as it enacts a performance hit in operation throughput.

## 6.1.2. Audit Log Topics

OpenAM integrates log messages based on four different audit topics. A *topic* is a category of audit log event that has an associated one-to-one mapping to a schema type. Topics can be broadly categorized as access details, system activity, authentication operations, and configuration changes. The following table shows the basic event topics and associated audit log files for OpenAM's default audit logging configuration, which uses a JSON audit event handler:

Table 6.1. Audit Log Topics

Event Topic	File Name	Description
Access	<code>access.audit.json</code>	Captures who, what, when, and output for every access request.
Activity	<code>activity.audit.json</code>	<p>Captures state changes to objects that have been created, updated, or deleted by end users (that is, non-administrators). For this release, only session changes are captured in the logs.</p> <p>Future releases may also record changes to user trusted devices, UMA policies, OAuth 2.0 tokens and others.</p>
Authentication	<code>authentication.audit.json</code>	Captures when and how a subject is authenticated and related events.
Configuration	<code>config.audit.json</code>	Captures configuration changes to the product with a timestamp and by whom. Note that the

Event Topic	File Name	Description
		<code>userId</code> indicating the subject who made the configuration change is not captured in the <code>config.audit.json</code> but may be tracked using the <code>transactionId</code> in the <code>access.audit.json</code> .

## 6.2. Implementing the Audit Logging Service

When implementing the Audit Logging Service, decide whether you require specific audit systems per realm, or if a global configuration suits your deployment. Next, determine which event handlers suit your needs from those supported by OpenAM. See the following sections for more information:

- To configure the Audit Logging Service, see Section 6.2.1, "Configuring Audit Logging".
- To configure the audit event handlers, see Section 6.2.2, "Configuring Audit Event Handlers".
- To configure the propagation of the transaction ID across the ForgeRock Identity Platform, see Section 6.2.3, "Configuring the Trust Transaction Header System Property".

AM 5 also supports the classic Logging Service, based on Java SDK, that will be deprecated in a future release. For more information, see Section 6.3, "Implementing the Classic Logging Service".

### 6.2.1. Configuring Audit Logging

OpenAM's default audit event handler is the JSON audit event handler, which comes configured and enabled for the global Audit Logging Service. The global configuration is used to control audit logging in realms that do not have the Audit Logging Service added to them. OpenAM also supports configuring an Audit Logging Service on a per-realm basis.

The JSON audit event handler stores its JSON log files under `/path/to/openam/openam/log`.

- To modify the global audit logging configuration, see Procedure 6.1, "To Configure Global Audit Logging".
- To override the global audit logging configuration for a realm, see Procedure 6.2, "To Configure Audit Logging in a Realm".

#### *Procedure 6.1. To Configure Global Audit Logging*

1. Log in to the AM console as an administrator, for example `amadmin`.
2. Navigate to Configure > Global Services > Audit Logging.
3. Configure the following options on the Global Attributes tab:
  - a. Activate Audit logging to start the audit logging feature.

- b. In the Field exclusion policies list, enter any values to exclude from your audit events, or change the default exclusions for items that need to be included. To specify a field or value within a field to be filtered out of the event, start the pointer with the event topic (access, activity, authentication, or configuration) followed by the field name or the path to the value in the field.

This feature allows two types of filtering:

- Filter fields from the event. You may not be interested in capturing HTTP headers, query parameters, or potentially sensitive data like passwords in the access logs. For example, to filter out the `userid` field in an access event, specify the pointer as:

```
/access/userid
```

- Filter specific values from within fields that store key-value pairs as JSON, such as the HTTP headers, query parameters, and cookies. For example, to filter out the content type value in the `http.request.headers` field, specify the pointer as:

```
/access/http/request/headers/content-type
```

- c. Save your changes.
4. On the Secondary Configurations tab, you can edit the configuration of the Global JSON Handler, or you can create new audit event handlers. For more information, see [Section 6.2.2, "Configuring Audit Event Handlers"](#).

## Procedure 6.2. To Configure Audit Logging in a Realm

You can configure the Audit Logging Service for realms, allowing you to configure realm-specific log locations and handler types.

When the Audit Logging Service is added to a realm, it inherits the configuration defined under [Configure > Global Services > Audit Logging > Realm Defaults](#). Properties configured explicitly in the realm-level service override the realm defaults.

To configure the Audit Logging Service in a realm, perform the following steps:

1. Navigate to [Realms > Realm Name > Services](#).
2. Select [Add a Service](#).
3. On the Choose a service type drop-down menu, select [Audit Logging](#).
4. Select [Create](#).

The Audit Logging Service page appears. Configure the Audit Logging Service as follows:

5. Ensure audit logging is [Enabled](#).
6. In the Field exclusion policies list, enter any values to exclude from your audit events, or delete the ones by default that you do not need. To specify a field or value within a field to be filtered



out of the event, start the pointer with the event topic (access, activity, authentication, or configuration) followed by the field name or the path to the value in the field.

This feature allows two types of filtering:

- Filter fields from the event. You may not be interested in capturing HTTP headers, query parameters, or potentially sensitive data like passwords in the access logs. For example, to filter out the `userid` field in an access event, specify the pointer as:

```
/access/userid
```

- Filter specific values from within fields that store key-value pairs as JSON. For example, the HTTP headers, query parameters, and cookies. For example, to filter out the content type value in the `http.request.headers` field, specify the pointer as:

```
/access/http/request/headers/content-type
```

7. Save your changes.
8. On the Secondary Configurations tab, select Add a Secondary Configuration. Choose an event handler from the list.

For more information about supported event handlers and how to configure them, see Section 6.2.2, "Configuring Audit Event Handlers".

## 6.2.2. Configuring Audit Event Handlers

OpenAM supports the following types of audit event handlers:

*Table 6.2. Audit Event Handlers*

Audit Event Handler Type	Publishes to	How to Configure
JSON	JSON files	Section 6.2.2.1, "Configuring JSON Audit Event Handlers"
CSV	CSV files	Section 6.2.2.2, "Configuring CSV Audit Event Handlers"
Syslog	The syslog daemon	Section 6.2.2.3, "Configuring Syslog Audit Event Handlers"
JDBC	A relational database	Section 6.2.2.4, "Implementing JDBC Audit Event Handlers"
Elasticsearch	An Elasticsearch store	Section 6.2.2.5, "Implementing Elasticsearch Audit Event Handlers"
JMS	JMS topics	Section 6.2.2.6, "Implementing JMS Audit Event Handlers"
Splunk	A Splunk server	Section 6.2.2.7, "Implementing Splunk Audit Event Handlers"

This section provides procedures for configuring each type of audit handler.

### 6.2.2.1. Configuring JSON Audit Event Handlers

The following procedure describes how to configure a JSON audit event handler:

#### *Procedure 6.3. To Configure a JSON Audit Event Handler*

1. Log in to the AM console as an administrator, for example `amadmin`.
2. Determine whether to create the event handler in a realm or use the default global event handler, then take one of the following actions:
  - To create the event handler in the global configuration, navigate to `Configure > Global Services > Audit Logging`.
  - To create the event handler in a realm, navigate to `Realms > Realm Name > Services > Audit Logging`.
3. On the Secondary Configurations tab, click Global JSON Handler or the Edit icon on the right if present. If no handler is present, click Add a Secondary Configuration, and select JSON.
4. Under the New JSON configuration page, enter a name for the event handler. For example, `JSON Audit Event Handler`.
5. (Optional) In the Rotation Times field, enter a time duration after midnight to trigger file rotation, in seconds. For example, you can provide a value of `3600` to trigger rotation at 1:00 AM. Negative durations are not supported.
6. Click Create.

After the JSON audit event handler is created, several configuration tabs appear. To configure the event handler, perform the following steps:

7. On the General Handler Configuration tab, enable the event handler and configure the topics for your audit logs:
  - a. Select Enabled to activate the event handler, if disabled.
  - b. Choose the topics for your audit logs. For a description of each topic, see Section 6.1.2, "Audit Log Topics".
  - c. Click Save Changes.
8. On the JSON Configuration tab, configure JSON options:

- a. Override the default location of your logs if necessary, and save your changes. The default value is `%BASE_DIR%/SERVER_URI%/Log/`.

#### Important

Make sure to configure a different log directory for each JSON audit event handler instance. If two instances are writing to the same file, it can interfere with log rotation and tamper-evident logs.

- b. Select ElasticSearch JSON Format Compatible to direct OpenAM to generate JSON formats that are compatible with the ElasticSearch format.
  - c. In the File Rotation Retention Check Interval field, edit the time interval (seconds) to check the time-base file rotation policies.
  - d. Click Save Changes.
9. On the File Rotation tab, configure how files are rotated when they reach a specified file size or time interval:
- a. Select Rotation Enabled to activate file rotation. If file rotation is disabled, OpenAM ignores log rotation and appends to the same file.
  - b. In the Maximum File Size field, enter the maximum size of an audit file before rotation.
  - c. (Optional). In the File Rotation Prefix field, enter an arbitrary string that will be prefixed to every audit log to identify it. This parameter is used when time-based or size-based rotation is enabled.
  - d. In the File Rotation Suffix field, enter a timestamp suffix based on the Java SimpleDateFormat that will be added to every audit log. This parameter is used when time-based or size-based log rotation is enabled. The default value is `-yyyy.MM.dd-kk.mm.ss`.
  - e. In the Rotation Interval field, enter a time interval to trigger audit log file rotation in seconds. A negative or zero value disables this feature.
  - f. (Optional) In the Rotation Times field, enter a time duration after midnight to trigger file rotation, in seconds. For example, you can provide a value of `3600` to trigger rotation at 1:00 AM. Negative durations are not supported.
  - g. Click Save Changes.
10. On the File Retention tab, configure how long log files should be retained in your system:
- a. In the Maximum Number of Historical Files field, enter a number for allowed backup audit files. A value of `-1` indicates an unlimited number of files and disables the pruning of old history files.

- b. In the Maximum Disk Space field, enter the maximum amount of disk space that the audit files can use. A negative or zero value indicates that this policy is disabled.
  - c. In the Minimum Free Space Required field, enter the minimum amount of disk space required to store audit files. A negative or zero value indicates that this policy is disabled.
  - d. Click Save Changes.
11. On the Buffering tab, configure whether log events should be buffered in memory before they are written to the JSON file:
  - a. In the Batch Size field, enter the maximum number of audit log events that can be buffered.
  - b. In the Write interval field, enter the time interval in milliseconds at which buffered events are written to a file.
  - c. Click Save Changes.

#### 6.2.2.2. Configuring CSV Audit Event Handlers

The following procedure describes how to configure a comma-separated values (CSV) audit event handler:

##### *Procedure 6.4. To Configure a CSV Audit Event Handler*

1. Log in to the AM console as an administrator, for example `amadmin`.
  2. Determine whether to create the event handler in a realm or use the default global event handler, then take one of the following actions:
    - To create the event handler in the global configuration, navigate to `Configure > Global Services > Audit Logging`.

Note that the CSV audit event handler is already configured in the global configuration. Select it to change its properties.

    - To create the event handler in a realm, navigate to `Realms > Realm Name > Services > Audit Logging`.
  3. On the Secondary Configurations tab, select Add a Secondary Configuration. Choose CVS from the list.
- The New CVS page appears. Enter the basic configuration for the new handler by performing the following actions:
4. Enter a name for the event handler. For example, `CSV Audit Event Handler`.

5. (Optional) In the Rotation Times field, enter a time duration after midnight to trigger file rotation, in seconds. For example, you can provide a value of `3600` to trigger rotation at 1:00 AM. Negative durations are not supported.
6. Enable or disable the Buffering option.
7. Select Create.

After the CSV audit event handler is created, several configuration tabs appear. To configure the event handler, perform the following steps:

8. On the General Handler Configuration tab, enable the event handler and configure the topics for your audit logs:
  - a. Select Enabled to activate the event handler, if disabled.
  - b. Choose the topics for your audit logs. For a description of each topic, see Section 6.1.2, "Audit Log Topics".
  - c. Save your changes.
9. On the CSV Configuration tab, override the default location of your logs if necessary, and save your changes. The default value is `%BASE_DIR%/%SERVER_URI%/Log/`.

#### Important

Configure a different log directory for each CVS audit event handler instance. If two instances are writing to the same file, it can interfere with log rotation and tamper-evident logs.

10. On the File Rotation tab, configure how files are rotated when they reach a specified file size or time interval:
  - a. Select Rotation Enabled to activate file rotation. If file rotation is disabled, OpenAM ignores log rotation and appends to the same file.
  - b. In the Maximum File Size field, enter the maximum size of an audit file before rotation.
  - c. (Optional). In the File Rotation Prefix field, enter an arbitrary string that will be prefixed to every audit log to identify it. This parameter is used when time-based or size-based rotation is enabled.
  - d. In the File Rotation Suffix field, enter a timestamp suffix based on the Java SimpleDateFormat that will be added to every audit log. This parameter is used when time-based or size-based log rotation is enabled. The default value is `-yyyy.MM.dd-kk.mm.ss`.
  - e. In the Rotation Interval field, enter a time interval to trigger audit log file rotation in seconds. A negative or zero value disables this feature.

- f. (Optional) In the Rotation Times field, enter a time duration after midnight to trigger file rotation, in seconds. For example, you can provide a value of **3600** to trigger rotation at 1:00 AM. Negative durations are not supported.
        - g. Save your changes.
11. On the File Retention tab, configure how long log files should be retained in your system:
  - a. In the Maximum Number of Historical Files field, enter a number for allowed backup audit files. A value of **-1** indicates an unlimited number of files and disables the pruning of old history files.
  - b. In the Maximum Disk Space field, enter the maximum amount of disk space that the audit files can use. A negative or zero value indicates that this policy is disabled.
  - c. In the Minimum Free Space Required field, enter the minimum amount of disk space required to store audit files. A negative or zero value indicates that this policy is disabled.
  - d. Save your changes.
12. On the Buffering tab, configure whether log events should be buffered in memory before they are written to the CSV file:
  - a. Select Buffering Enabled to activate buffering.

When buffering is enabled, all audit events are put into an in-memory buffer (one per handled topic), so that the original thread that generated the event can fulfill the requested operation, rather than wait for I/O to complete. A dedicated thread (one per handled topic) constantly pulls events from the buffer in batches and writes them to the CSV file. If the buffer becomes empty, the dedicated thread goes to sleep until a new item gets added. The default buffer size is **5000** bytes.
  - b. Enable Flush Each Event Immediately to write all buffered events before flushing.

When the dedicated thread accesses the buffer, it copies the contents to an array to reduce contention, and then iterates through the array to write to the CSV file. The bytes written to the file can be buffered again in Java classes and the underlying operating system.

When Flush Each Event Immediately is enabled, OpenAM flushes the bytes after each event is written. If the feature is disabled (default), the Java classes and underlying operation system determine when to flush the bytes.
  - c. Save your changes.
13. On the Tamper Evident Configuration tab, configure whether to detect audit log tampering:
  - a. Select Is Enabled to activate the tamper evident feature for CSV logs.

When tamper evident logging is enabled, OpenAM generates an HMAC digest for each audit log event and inserts it into each audit log entry. The digest detects any addition or modification to an entry.

OpenAM also supports another level of tamper evident security by periodically adding a signature entry to a new line in each CSV file. The entry signs the preceding block of events, so that verification can establish if any of these blocks have been added, removed, or edited by some user.

- b. In the Certificate Store Location field, enter the location of the `Logger.jks` keystore, by default `%BASE_DIR%/%SERVER_URI%/Logger.jks`. You must manually create the keystore and place it in this location. You can use a simple script to create your Java keystore: `create-keystore.sh`.
- c. In the Certificate Store Password field, enter the password of the keystore.
- d. In the Signature Interval field, enter a value in seconds for OpenAM to generate and add a new signature to the audit log entry.
- e. Save your changes.

### 6.2.2.3. Configuring Syslog Audit Event Handlers

OpenAM can publish audit events to a syslog server, which is based on a widely-used logging protocol. You can configure your syslog settings on the AM console.

The following procedure describes how to configure a Syslog audit event handler:

#### *Procedure 6.5. To Configure a Syslog Audit Event Handler*

1. Log in to the AM console as an administrator, for example `amadmin`.
2. Determine whether to create the event handler in a realm or use the default global event handler, then take one of the following actions:
  - To create the event handler in the global configuration, navigate to `Configure > Global Services > Audit Logging`.
  - To create the event handler in a realm, navigate to `Realms > Realm Name > Services > Audit Logging`.
3. On the Secondary Configurations tab, select `Add a Secondary Configuration`. Choose Syslog from the list.

The New Syslog page appears. Enter the basic configuration for the new handler by performing the following actions:

4. Enter a name for the event handler. For example, `Syslog Audit Event Handler`.

5. In the Server hostname field, enter the hostname or IP address of the receiving syslog server.
6. In the Server port field, enter the port of the receiving syslog server.
7. In the Connection timeout field, enter the number of seconds to connect to the syslog server. If the server has not responded in the specified time, a connection timeout occurs.
8. Enable or disable the Buffering option.
9. Select Create.

After the syslog audit event handler is created, several configuration tabs appear. To configure the event handler, perform the following steps:

10. On the General Handler Configuration tab, enable the event handler and configure the topics for your audit logs:
  - a. Select Enabled to activate the event handler, if disabled.
  - b. Choose the topics for your audit logs. For a description of each topic, see [Section 6.1.2, "Audit Log Topics"](#).
  - c. Save your changes.
11. On the Audit Event Handler Factory tab, keep the default class name for the audit event handler.
12. On the Syslog Configuration tab, configure the main syslog event handler properties:
  - a. In the Server hostname field, enter the hostname or IP address of the receiving syslog server.
  - b. In the Server port field, enter the port of the receiving syslog server.
  - c. In the Connection timeout field, enter the number of seconds to connect to the syslog server. If the server has not responded in the specified time, a connection timeout occurs.
  - d. In the Transport Protocol drop-down menu, select TCP or UDP.
  - e. Choose the facility.

A syslog message includes a PRI field that is calculated from the facility and severity values. All topics set the severity to **INFORMATIONAL** but you can choose the facility on the Facility drop-down menu:

*Table 6.3. Syslog Facilities*

Facility	Description
AUTH	Security or authorization messages
AUTHPRIV	Security or authorization messages



Facility	Description
CLOCKD	Clock daemon
CRON	Scheduling daemon
DAEMON	System daemons
FTP	FTP daemon
KERN	Kernel messages
LOCAL0	Local use 0 (local0)
LOCAL1	Local use 1 (local1)
LOCAL2	Local use 2 (local2)
LOCAL3	Local use 3 (local3)
LOCAL4	Local use 4 (local4)
LOCAL5	Local use 5 (local5)
LOCAL6	Local use 6 (local6)
LOCAL7	Local use 7 (local7)
LOGALERT	Log alert
LOGAUDT	Log audit
LPR	Line printer subsystem
MAIL	Mail system
NEWS	Network news subsystem
NTP	Network time protocol
SYSLOG	Internal messages generated by syslogd
USER	User-level messages
UUCP	Unix-to-unix-copy (UUCP) subsystem

f. Save your changes.

13. On the Buffering tab, configure whether you want buffering or not:

a. Select Buffering Enabled to activate it.

When buffering is enabled, all audit events that get generated are formatted as syslog messages and put into a queue. A dedicated thread constantly pulls events from the queue in batches and transmits them to the syslog server. If the queue becomes empty, the dedicated thread goes to sleep until a new item gets added. The default queue size is 5000.

b. Save your changes.

## 6.2.2.4. Implementing JDBC Audit Event Handlers

OpenAM supports audit logging to relational databases using the JDBC audit event handler. You can configure OpenAM to write to Oracle, MySQL, or other databases.

Before configuring the JDBC audit event handler, you must perform several steps to allow OpenAM to log to the database:

### *Procedure 6.6. To Prepare for JDBC Audit Logging*

1. Create tables in the relational database in which you will write the audit logs. The SQL for Oracle and MySQL table creation is in the `audit.sql` file under `/path/to/tomcat/webapps/openam/WEB-INF/template/sql/db-type`.

If you are using a different relational database, tailor the Oracle or MySQL `audit.sql` file to conform to your database's SQL syntax.

2. JDBC audit logging requires a database user with read and write privileges for the audit tables. Do one of the following:
  - Identify an existing database user and grant that user privileges for the audit tables.
  - Create a new database user with read and write privileges for the audit tables.
3. Obtain the JDBC driver from your database vendor. Place the JDBC driver `.zip` or `.jar` file in the container's `WEB-INF/lib` classpath. For example, place the JDBC driver in `/path/to/tomcat/webapps/openam/WEB-INF/lib` if you use Apache Tomcat.

The following procedure describes how to configure a JDBC audit event handler. Perform the following steps after you have created audit log tables in your database and installed the JDBC driver in the OpenAM web container:

### *Procedure 6.7. To Configure a JDBC Audit Event Handler*

1. Log in to the AM console as an administrator, for example `amadmin`.
2. Determine whether to create the event handler in a realm or use the default global event handler, then take one of the following actions:
  - To create the event handler in the global configuration, navigate to `Configure > Global Services > Audit Logging`.
  - To create the event handler in a realm, navigate to `Realms > Realm Name > Services > Audit Logging`.
3. On the Secondary Configurations tab, select Add a Secondary Configuration. Choose JDBC from the list.

The New JDBC page appears. Enter the basic configuration for the new handler by performing the following actions:.

4. Enter a name for the event handler. For example, `JDBC Audit Event Handler`.
5. In the JDBC Database URL field, enter the URL for your database server. For example, `jdbc:oracle:thin@//host.example.com:1521/ORCL`.
6. In the JDBC Drive field, enter the classname of the driver to connect to the database. For example, `oracle.jdbc.driver.OracleDriver` or `com.mysql.jdbc.Driver`.
7. In the Database Username field, enter the username to authenticate to the database server.  
This user must have read and write privileges for the audit tables.
8. In the Database Password field, enter the password used to authenticate to the database server.
9. Enable or disable the Buffering option.
10. Select the Create button.

After the JDBC audit event handler is created, several configuration tabs appear. To configure the event handler, perform the following steps:

11. On the General Handler Configuration tab, enable the handler and configure the topics for your audit logs:
  - a. Select Enabled to activate the event handler, if disabled.
  - b. Select the topics for your audit logs. For a description of each topic, see Section 6.1.2, "Audit Log Topics".
  - c. Save your changes.
12. On the Audit Event Handler Factory tab, enter the fully qualified class name of your custom JDBC audit event handler and save your changes.
13. On the Database Configuration tab, configure the main JDBC event handler properties:
  - a. On the Database Type drop-down menu, select the audit database type. The default value is `oracle`.
  - b. In the JDBC Database URL field, enter the URL for your database server. For example, `jdbc:oracle:thin@//host.example.com:1521/ORCL`.
  - c. In the JDBC Drive field, enter the classname of the driver to connect to the database. For example, `oracle.jdbc.driver.OracleDriver` or `com.mysql.jdbc.Driver`.
  - d. In the Database Username field, enter the username to authenticate to the database server.  
This user must have read and write privileges for the audit tables.
  - e. In the Database Password field, enter the password used to authenticate to the database server.

- f. In the Connection Timeout (seconds) field, enter the maximum wait time before failing the connection.
- g. In the Maximum Connection Idle Timeout (seconds) field, enter the maximum idle time in seconds before the connection is closed.
- h. In the Maximum Connection Time (seconds) field, enter the maximum time in seconds for a connection to stay open.
- i. In the Minimum Idle Connections field, enter the minimum number of idle connections allowed in the connection pool.
- j. In the Maximum Connections field, enter the maximum number of connections in the connection pools.
- k. Save your changes.

14. On the Buffering tab, configure the buffering settings:

- a. Select Buffering Enabled to start audit event buffering.
- b. In the Buffer Size (number of events) field, set the size of the event buffer queue where events should queue up before being written to the database.

If the queue reaches full capacity, the process will block until a write occurs.

- c. In the Write Interval field, set the interval in seconds in which buffered events are written to the database.
- d. In the Writer Threads field, set the number of threads used to write the buffered events.
- e. In the Max Batched Events field, set the maximum number of batched statements the database can support per connection.
- f. Save your changes.

### 6.2.2.5. Implementing Elasticsearch Audit Event Handlers

OpenAM supports audit logging to Elasticsearch. When you store OpenAM's audit logs in an Elasticsearch data store, you can use Kibana to perform data discovery and visualization on your logs.

You can experiment with an Elasticsearch audit handler without enabling any Elasticsearch security features. However, for a more secure deployment, ForgeRock recommends that you use Elasticsearch Shield to require authentication to Elasticsearch. Depending on your network topology, you might also want to configure SSL for Elasticsearch Shield.

Before configuring the Elasticsearch audit event handler, you must configure an Elasticsearch index with OpenAM's audit schema:

### Procedure 6.8. To Prepare for Elasticsearch Audit Logging

1. Review the JSON file containing OpenAM's audit schema. You can find the JSON file for the audit schema at the path `/path/to/tomcat/webapps/openam/WEB-INF/template/elasticsearch/audit.json`.
2. Copy the `audit.json` file to the system where you will create the Elasticsearch index for OpenAM auditing.

In this example, you create an Elasticsearch index by executing an Elasticsearch REST API call using the `curl` command. Copy the `audit.json` file to a location that is accessible to the `curl` command you will run in the next step.

3. Create an Elasticsearch index for OpenAM auditing as follows:

```
$ curl \
  --request POST \
  --header "Content-Type: application/json" \
  --data @audit.json \
  http://elasticsearch.example.com:9200/my_openam_audit_index
```

In this example, note the following:

- `elasticsearch.example.com` is the name of the host on which Elasticsearch runs.
- `9200` is the port number that you use to access Elasticsearch's REST API.
- `my_openam_audit_index` is the name of the Elasticsearch index that you want to create.

The following procedure describes how to configure an Elasticsearch audit event handler. Perform the following steps after you have created an Elasticsearch index for OpenAM audit logging:

### Procedure 6.9. To Configure an Elasticsearch Audit Event Handler

1. If your Elasticsearch deployment uses Elasticsearch Shield configured for SSL, import the CA certificate used to sign Elasticsearch node certificates into the Java keystore on the host that runs OpenAM. For example:

```
$ keytool \
  -import \
  -trustcacerts \
  -alias elasticsearch \
  -file /path/to/cacert.pem \
  -keystore $JAVA_HOME/jre/lib/security/cacerts
```

If you are running an OpenAM site, import the CA certificate on all the servers in your site.

2. Log in to the AM console as an administrator, for example `amadmin`.
3. Determine whether to create the event handler in a realm or use the default global event handler, then take one of the following actions:

- To create the event handler in the global configuration, navigate to **Configure > Global Services > Audit Logging**.
  - To create the event handler in a realm, navigate to **Realms > *Realm Name* > Services > Audit Logging**.
4. On the **Secondary Configurations** tab, select **Add a Secondary Configuration**. Choose **Elasticsearch** from the list.  
  
The **New Elasticsearch** page appears. Enter the basic configuration for the new handler by performing the following actions:
    5. Enter a name for the event handler. For example, **Elasticsearch Audit Event Handler**.
    6. In the **Server Hostname** field, enter the hostname or IP address of the Elasticsearch server to which OpenAM should connect when writing audit events.
    7. In the **Server Port** field, enter the port number to access Elasticsearch's REST API.
    8. In the **Elasticsearch Index** field, specify the name of the index to be used for OpenAM audit logging. The index you specify in this field must be identical to the index you created in [Procedure 6.8, "To Prepare for Elasticsearch Audit Logging"](#).
    9. Specify the name and password of an Elasticsearch user if you have configured Elasticsearch Shield for user authentication:
      - a. In the **Username** field, enter the username to authenticate into the Elasticsearch server.
      - b. In the **Password** field, enter the password of the user that authenticates into the Elasticsearch server.
      - c. Save your changes.

If you are not using Elasticsearch Shield for user authentication, you can leave these fields with their default values.
  10. Enable or disable the **Buffering** option.
  11. Select **Create**.

After the Elasticsearch audit event handler is created, several configuration tabs appear. To configure the event handler, perform the following steps:

12. On the **General Handler Configuration** tab, enable the handler and configure the topics for your audit logs:
  - a. Select **Enabled** to activate the event handler, if disabled.
  - b. Select the topics for your audit logs. For a description of each topic, see [Section 6.1.2, "Audit Log Topics"](#).

- c. Save your changes.
13. On the Audit Event Handler Factory, keep the default class name for the audit event handler.
  14. On the Elasticsearch Configuration tab, configure the main Elasticsearch event handler properties:
    - a. In the Server Hostname field, enter the hostname or IP address of the Elasticsearch server to which OpenAM should connect when writing audit events.
    - b. In the Server Port field, enter the port number to access Elasticsearch's REST API.
    - c. If SSL is enabled in your Elasticsearch deployment, select SSL Enabled.
    - d. In the Elasticsearch Index field, specify the name of the index to be used for OpenAM audit logging. The index you specify in this field must be identical to the index you created in Procedure 6.8, "To Prepare for Elasticsearch Audit Logging".
    - e. Save your changes.
  15. On the Authentication tab, specify the name and password of an Elasticsearch user if you have configured Elasticsearch Shield for user authentication:
    - a. In the Username field, enter the username to authenticate into the Elasticsearch server.
    - b. In the Password field, enter the password of the user that authenticates into the Elasticsearch server.
    - c. Save your changes.

If you are not using Elasticsearch Shield for user authentication, you can leave these fields with their default values.

16. On the Buffering tab, configure whether log events should be buffered in memory before they are written to the Elasticsearch data store:
  - a. Activate Buffering Enabled to start buffering audit messages.

When buffering is enabled, all audit events are put into an in-memory buffer (one per handled topic), so that the original thread that generated the event can fulfill the requested operation, rather than wait for I/O to complete. A dedicated thread (one per handled topic) constantly pulls events from the buffer in batches and writes them to Elasticsearch. If the buffer becomes empty, the dedicated thread goes to sleep until a new item gets added.
  - b. In the Batch Size field, specify the number of audit events that OpenAM pulls from the audit buffer when writing a batch of events to Elasticsearch.
  - c. In the Queue Capacity field, specify the maximum number of audit events that OpenAM can queue in this audit handler's buffer.

If the number of events to queue exceeds the queue capacity, OpenAM raises an exception and the excess audit events are dropped, and therefore not written to Elasticsearch.

- d. In the Write interval (in millis) field, specify how often OpenAM should write buffered events to Elasticsearch.
- e. Save your changes.

### 6.2.2.6. Implementing JMS Audit Event Handlers

OpenAM supports audit logging to a JMS message broker. JMS is a Java API for sending messages between clients using a publish and subscribe model as follows:

- OpenAM audit logging to JMS requires that the JMS message broker supports using JNDI to locate a JMS connection factory. See your JMS message broker documentation to verify that you can make connections to your broker by using JNDI before attempting to implement an OpenAM JMS audit handler.
- OpenAM acts as a JMS publisher client, publishing JMS messages containing audit events to a JMS *topic*.<sup>1</sup>
- A JMS subscriber client, which is not part of the OpenAM software and must be developed and deployed separately from OpenAM, subscribes to the JMS topic to which OpenAM publishes audit events. The client then receives the audit events over JMS and processes them as desired.

Before configuring the JMS audit event handler, you must perform several steps to allow OpenAM to publish audit events as a JMS client:

#### *Procedure 6.10. To Prepare for JMS Audit Logging*

1. Obtain JNDI connection properties that OpenAM requires to connect to your JMS message broker. The specific connection properties vary depending on the broker. See your JMS message broker documentation for details.

For example, connecting to an Apache ActiveMQ message broker requires the following properties:

*Table 6.4. Example Apache ActiveMQ JNDI Connection Properties*

Property Name	Example Value
<code>java.naming.factory.initial</code>	<code>org.apache.activemq.jndi.ActiveMQInitialContextFactory</code>
<code>java.naming.provider.url</code>	<code>tcp://localhost:61616</code>

<sup>1</sup> Note that OpenAM and JMS use the term *topic* differently. An *OpenAM audit topic* is a category of audit log event that has an associated one-to-one mapping to a schema type. A *JMS topic* is a distribution mechanism for publishing messages delivered to multiple subscribers.



Property Name	Example Value
<code>topic.audit</code>	<code>audit</code>

- Obtain the JNDI lookup name of the JMS connection factory for your JMS message broker.

For example, for Apache ActiveMQ, the JNDI lookup name is `ConnectionFactory`.

- Obtain the JMS client `.jar` file from your JMS message broker vendor. Add the `.jar` file to OpenAM's classpath by placing it in the `WEB-INF/lib` directory.

For example, place the JMS client `.jar` file in `/path/to/tomcat/webapps/openam/WEB-INF/lib` if you use Apache Tomcat.

The following procedure describes how to configure a JMS audit event handler.

If your JMS message broker requires an SSL connection, you might need to perform additional, broker-dependent configuration tasks. For example, you might need to import a broker certificate into OpenAM's keystore, or provide additional JNDI context properties.

See your JMS message broker's documentation for specific requirements for making SSL connections to your broker, and implement them as needed in addition to the steps in the following procedure.

Perform the following steps after you have installed the JMS client `.jar` file in the OpenAM web container:

#### *Procedure 6.11. To Configure a JMS Audit Event Handler*

- Log in to the AM console as an administrator, for example `amadmin`.
- Determine whether to create the event handler in a realm or use the default global event handler, then take one of the following actions:
  - To create the event handler in the global configuration, navigate to `Configure > Global Services > Audit Logging`.
  - To create the event handler in a realm, navigate to `Realms > Realm Name > Services > Audit Logging`.

- On the Secondary Configurations tab, select Add a Secondary Configuration. Choose JMS from the list.

The New JMS page appears. Enter the basic configuration for the new handler by performing the following actions:

- Enter a name for the event handler. For example, `JMS Audit Event Handler`.
- Select Create.

After the JMS audit event handler is created, several configuration tabs appear. To configure the event handler, perform the following steps:

6. On the General Handler Configuration tab, enable the handler and configure the topics for your audit logs:
  - a. Select Enabled to activate the event handler, if disabled.
  - b. Select the topics for your audit logs. For a description of each topic, see Section 6.1.2, "Audit Log Topics".
  - c. Save your changes.
7. On the Audit Event Handler Factory tab, keep the default class name for the audit event handler.
8. On the JMS Configuration tab, configure the main JMS event handler properties:
  - a. On the Delivery Mode drop-down menu, specify the JMS delivery mode.

With persistent delivery, the JMS provider ensures that messages are not lost in transit in case of a provider failure by logging messages to storage when they are sent. Therefore, persistent delivery mode guarantees JMS message delivery, while non-persistent mode provides better performance.

The default delivery mode is **NON-PERSISTENT** delivery. Therefore, if your deployment requires delivery of every audit event to JMS subscriber clients, be sure to set the configuration to **PERSISTENT** delivery.

- b. On the Session Mode drop-down menu, leave the default setting, **AUTO**, unless your JMS broker implementation requires otherwise. See your broker documentation for more information.
  - c. Specify properties that OpenAM will use to connect to your JMS message broker as key-value pairs in the JNDI Context Properties field.
 

OpenAM is configured for the **audit** JNDI lookup name and JMS topic, but you can modify or delete this configuration, or add new key-value pairs. To add new key-value pairs, fill the Key and Value fields and select the +add button.
  - d. In the JMS Topic field, specify the name of the JMS topic to which OpenAM will publish messages containing audit events.
 

Subscriber clients that process OpenAM audit events must subscribe to this topic.
  - e. In the JMS Connection Factory Name, specify the JNDI lookup name of the JMS connection factory.
  - f. Save your changes.
9. On the Batch Events tab, configure whether log events should be batched before they are published to the JMS message broker:
  - a. Activate Batch Enabled to start batch publishing of audit events. Audit events will be queued and published to the JMS message broker in batches.

If batch publishing is not enabled, OpenAM publishes audit events to the JMS message broker individually.

- b. In the Capacity field, specify the maximum capacity of the publishing queue. Execution is blocked if the queue size reaches capacity.
- c. In the Max Batched field, specify the maximum number of events to be delivered when OpenAM publishes the events to the JMS message broker.
- d. In the Thread Count field, specify the number of worker threads OpenAM should use to process the batch queue.
- e. Specify the batching timeout configuration as follows:
  - In the Insert Timeout field, specify the amount of time in seconds for queued events to be transmitted to the JMS message broker.
  - In the Polling Timeout field, specify the amount of time in seconds that worker threads wait for new audit events before becoming idle.
  - In the Shutdown Timeout field, specify the amount of time in seconds that worker threads wait for new audit events before shutting down.
- f. Save your changes.

### 6.2.2.7. Implementing Splunk Audit Event Handlers

OpenAM supports sending audit logging data to a Splunk HTTP event collector REST endpoint. This endpoint requires an authentication token created in Splunk and configured in the OpenAM event handler.

Before configuring the Splunk audit event handler in OpenAM, configure the endpoint in Splunk:

#### *Procedure 6.12. To Prepare Splunk*

Perform the following steps in Splunk:

1. Create a source type to match OpenAM audit logs. Consider the following configuration tips:
  - It must be a structured type.
  - It must not have indexed extractions.
  - It must have event breaks on every line.
  - Timestamp extraction must be automatic.
2. Create an HTTP event collector token that uses the source type you just created.

3. Obtain the HTTP event collector token value for the token you just created. For example, `AD565CB5-BB8A-40FD-8A7C-94F549CEDEC`.

This token, which allows OpenAM to log data to Splunk, is required for the OpenAM audit event handler configuration.

4. Obtain the HTTP event collector port. For example, `8088`.
5. Ensure that the HTTP event collector and its tokens are enabled.

The following procedure describes how to configure a Splunk audit event handler in OpenAM. Perform the following steps after you have created a Splunk HTTP event collector token:

### *Procedure 6.13. To Configure a Splunk Audit Event Handler*

1. Log in to the AM console as an administrator, for example `amadmin`.
2. Determine whether to create the event handler in a realm or use the default global event handler, then take one of the following actions:
  - To create the event handler in the global configuration, navigate to `Configure > Global Services > Audit Logging`.
  - To create the event handler in a realm, navigate to `Realms > Realm Name > Services > Audit Logging`.
3. On the Secondary Configurations tab, select `Add a Secondary Configuration`. Choose Splunk from the list.

The New Splunk configuration page appears. Enter the basic configuration for the new handler by performing the following actions:

4. Enter a name for the event handler. For example, `Splunk Audit Event Handler`.
5. In the Server Hostname field, enter the hostname or IP address of the Splunk HTTP event collector to which OpenAM should connect when writing audit events.
6. In the Server Port field, enter the port number to access the Splunk HTTP event collector.
7. In the Authorization Token field, enter the authorization token for the Splunk HTTP event collector REST endpoint.
8. Select `Create`.

After the Splunk audit event handler is created, several configuration tabs appear. To configure the event handler, perform the following steps:

9. On the General Handler Configuration tab, enable the handler and configure the topics for your audit logs:

- a. Select Enabled to activate the event handler, if disabled.
  - b. Select the topics for your audit logs. For a description of each topic, see Section 6.1.2, "Audit Log Topics".
  - c. Save your changes.
10. On the Audit Event Handler Factory tab, keep the default class name for the audit event handler.
11. On the Splunk Configuration tab, configure the main Splunk event handler properties:
  - a. In the Server Hostname field, enter the hostname or IP address of the Splunk server to which OpenAM should connect when writing audit events.
  - b. In the Server Port field, enter the port number to access the Splunk HTTP event collector REST endpoint.
  - c. If SSL is enabled and configured between OpenAM and the Splunk server, select SSL Enabled.
  - d. In the Authorization Token field, enter the authorization token for the Splunk HTTP event collector REST endpoint.
  - e. Save your changes.
12. On the Buffering tab, configure options about how log events should be buffered in memory before they are written to the Splunk data store:
  - a. In the Batch Size field, specify the number of audit events that OpenAM pulls from the audit buffer when writing a batch of events to Splunk.

When buffering is enabled, all audit events are put into an in-memory buffer (one per handled topic), so that the original thread that generated the event can fulfill the requested operation, rather than wait for I/O to complete. A dedicated thread (one per handled topic) constantly pulls events from the buffer in batches and writes them to Splunk. If the buffer becomes empty, the dedicated thread goes to sleep until a new item gets added.
  - b. In the Queue Capacity field, specify the maximum number of audit events that OpenAM can queue in this audit handler's buffer.

If the number of events to queue exceeds the queue capacity, OpenAM raises an exception and the excess audit events are dropped, and therefore not written to Splunk.
  - c. In the Write interval (in milliseconds) field, specify how often OpenAM should write buffered events to Splunk.
  - d. Save your changes.

### 6.2.3. Configuring the Trust Transaction Header System Property

OpenAM supports the propagation of the transaction ID across the ForgeRock platform, such as from OpenDJ or OpenIDM to OpenAM, using the HTTP header `X-ForgeRock-TransactionId`. The `X-ForgeRock-TransactionId` header is automatically set in all outgoing HTTP calls from one ForgeRock product to another. Customers can also set this header themselves from their own applications or scripts calling into the ForgeRock platform.

You can set a new property `org.forgerock.http.TrustTransactionHeader` to `true`, which will trust any incoming `X-ForgeRock-TransactionId` headers. By default, the `org.forgerock.http.TrustTransactionHeader` is set to `false`, so that a malicious actor cannot flood the system with requests using the same transaction ID header to hide their tracks.

#### *Procedure 6.14. To Configure the Trust Transactions Header System Property*

1. Log in to the AM console.
2. Navigate to Configure > Server Defaults > Advanced.
3. In the Add a Name field, enter `org.forgerock.http.TrustTransactionHeader`, and enter `true` in the corresponding Add a Value field.
4. Save your changes.

Your OpenAM instance will now accept incoming `X-ForgeRock-Transactionid` headers, which can then be tracked in the audit logs.

5. Repeat this procedure for all servers requiring this property.

## 6.3. Implementing the Classic Logging Service

### Note

AM 5 supports two Audit Logging Services: the classic Logging Service, which is based on a Java SDK and is available in OpenAM versions prior to AM 5, and a common REST-based Audit Logging Service. The classic Logging Service is deprecated.

To configure OpenAM logging properties, log in to the OpenAM console as OpenAM administrator, and navigate to Configure > Global Services > Logging.

For more information on the available settings, see Section 10.5, "Audit Logging" reference.

### 6.3.1. Audit Logging to Flat Files

By default, OpenAM audit logs are written to files in the configuration directory for the instance, such as `$HOME/openam/log/`.

OpenAM sends messages to different log files, each named after the service logging the message, with two different types log files per service: `.access` and `.error`. Thus, the current log files for the authentication service are named `amAuthentication.access` and `amAuthentication.error`.

For details, see Chapter 6, "Log Files and Messages" in the *Reference*.

### 6.3.2. Audit Logging to a Syslog Server

OpenAM supports sending audit log messages to a syslog server for collation.

You can enable syslog audit logging by using the AM console, or the `ssoadm` command.

#### *Procedure 6.15. Enabling Syslog Audit Logging by Using the AM Console*

1. Log in to the AM console as OpenAM administrator, for example `amadmin`.
2. Navigate to Configure > Global Services > Logging.
3. On the Syslog tab, configure the following settings as appropriate for your syslog server, and save your changes:

- Syslog server host
- Syslog server port
- Syslog server protocol
- Syslog facility
- Syslog connection timeout

For information on these settings, see Section 2.2.11, "Logging" in the *Reference*.

4. On the General tab, set the Logging Type drop-down menu to `Syslog`, and save your changes.

#### *Procedure 6.16. Enabling Syslog Audit Logging by Using the ssoadm Command*

1. Create a text file, for example, `MySyslogServerSettings.txt` containing the settings used when audit logging to a syslog server, as shown below:

```
iplanet-am-logging-syslog-port=514
iplanet-am-logging-syslog-protocol=UDP
iplanet-am-logging-type=Syslog
iplanet-am-logging-syslog-connection-timeout=30
iplanet-am-logging-syslog-host=localhost
iplanet-am-logging-syslog-facility=local5
```

2. Use the following SSOADM command to configure audit logging to a syslog server:

```
$ ssoadm \  
  set-attr-defs \  
    --adminid amadmin \  
    --password-file /tmp/pwd.txt \  
    --servicename iPlanetAMLoggingService \  
    --schematype Global \  
    --datafile MySyslogServerSettings.txt
```

Schema attribute defaults were set.

### 6.3.3. Audit Logging in Policy Agents

By default, OpenAM Policy Agents log to local files in their configuration directories for debugging. The exact location depends on where you installed the agent.

By default, OpenAM policy agents send log messages remotely to OpenAM when you log auditing information about URL access attempts. To configure audit logging for a centrally managed policy agent, login to the OpenAM console as administrator, and navigate to Realms > *Realm Name* > Applications > Agents > *Agent Type* > *Agent Name* > Global, and then scroll down to the Audit section.



## Chapter 7

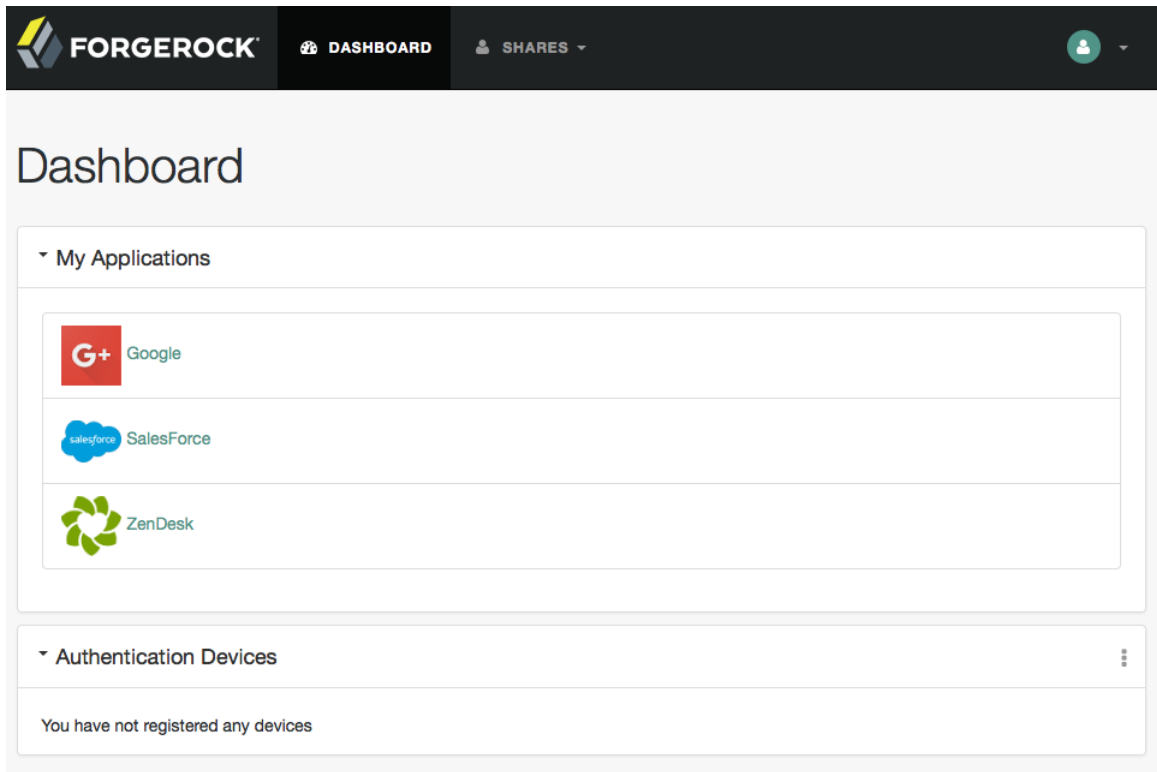
# Setting Up the Dashboard Service

This chapter shows how to configure the Dashboard service.

## 7.1. Introducing the Dashboard Service

The Dashboard Service provides the end user with an interface to access applications secured by OpenAM, both cloud-based applications like Salesforce and internal applications protected by policy agents. The Dashboard Service uses SSO to login to the applications when the user clicks on the application icon. For some apps, like Salesforce, you will want to limit access to only a few users. Other apps, like Google Mail or Drive, you will probably want to make available to all users.

Figure 7.1. User Dashboard Screen



The Dashboard Service is meant to give users a single place to access their applications. Keep in mind that this does not limit user access, only what appears on the user dashboard.

There are three stages to setting up the Dashboard Service:

- Setup the Dashboard Service and add applications.
- Add the service to the realms.
- Assign users applications so that they appear on the users' dashboards. This can be done manually or through a provisioning solution.

User dashboard pages require the XUI, which is enabled by default in OpenAM. To verify that XUI is enabled, log in to the AM console as administrator and navigate to Configure > Authentication, click Core Attributes, and then enable XUI Interface.

Once the Dashboard Service is configured for a user, the user can access their dashboard after logging in through the XUI under `/XUI/?realm=/#dashboard/`.

When making a request to the XUI, specify the realm or realm alias as the value of a `realm` parameter in the query string, or the DNS alias in the domain component of the URL. If you do not use a realm alias, then you must specify the entire hierarchy of the realm, starting at the top-level realm. For example `https://openam.example.com:8443/openam/XUI/?realm=/customers/europe#login/`.

For example, the full URL depending on the deployment might be at `https://openam.example.com:8443/openam/XUI/?realm=/myrealm#dashboard/`.

## 7.2. Implementing the Dashboard Service

Making some applications universally available ensures that all users have the same basic applications. However, some of your applications should be protected from the majority of your users. You will need to single out which users will include the application on their dashboard.

There are three default applications in the Dashboard Service: Google, Salesforce, and ZenDesk.

- To add applications to the Dashboard Service, see Procedure 7.1, "To Add Applications to the Dashboard Service".
- To add the Dashboard Service to a realm, see Procedure 7.2, "To Add the Application Dashboard Service to a Realm".
- To add an application to a user's dashboard, see Procedure 7.3, "To Add an Application to a User's Dashboard".
- To remove an application from a user's dashboard, see Procedure 7.4, "Removing User Access to an Application".

### *Procedure 7.1. To Add Applications to the Dashboard Service*

You can add applications to the Dashboard Service with the following steps. All fields except the dashboard class name and ICF Identifier are required for the application to work properly from the dashboard:

1. Log in to the AM console as OpenAM Administrator, `amadmin`.
2. Navigate to Configure > Global Services, click Dashboard, and then click New to add a new application to the Dashboard Service and to provide the information needed to connect to the app.
3. Provide a unique name for the application.
4. Add a Dashboard Class Name that identifies how the end user will access the app, such as `SAML2ApplicationClass` for a SAML v2.0 application.
5. Add a Dashboard Name for the application.
6. Add a Dashboard Display Name. This name is what the end user will see, such as Google.

7. Add the Dashboard Icon you would like the end user to see for the application. Either use a fully-qualified URL or an appropriate relative URL so that the icon is rendered properly on the user dashboard.
8. Add the Dashboard Login URL to point to the location the end user will go to once they click on the icon.
9. Leave the ICF Identifier blank.
10. Click Add when you are done.

### *Procedure 7.2. To Add the Application Dashboard Service to a Realm*

You must add the Dashboard Service to a realm before it will be available. The following instructions show you how to add an application to a single realm. Before you begin, make sure you have the name of the application as it appears on the Secondary Configuration Instance table under Configure > Global Services > Dashboard:

1. Select Realms > *Realm Name* > Services, and then click Add a Service.
2. Select the Dashboard service, and then click Create.
3. Add or remove the applications you would like to appear on the Dashboard service for the realm.
4. Click Save Changes when you are done.

### *Procedure 7.3. To Add an Application to a User's Dashboard*

Use the following steps to add an application to a user's dashboard:

1. Select Realms > *Realm Name* > Subjects and click the user identifier to edit the user's profile.
2. Under Services, click Dashboard.
3. Add the application beside the user name under the user's Assigned Dashboard list.
4. Click Save.

### *Procedure 7.4. Removing User Access to an Application*

You may need to remove an application from user's dashboard, but you do not want to entirely delete the user. The following steps walk you through removing an application from a user's dashboard:

1. Select Realms > *Realm Name* > Subjects and click the user identifier to edit the user's profile.
2. Under Services, click Dashboard.
3. Delete the application beside the user name under the user's Assigned Dashboard list.

4. Click Save.

## 7.2.1. Displaying Dashboard Applications

OpenAM lets administrators configure online applications to display applications on user Dashboards. You can use the exposed REST API to display information about the online applications.

### **/dashboard/assigned**

This endpoint retrieves the list of applications assigned to the authenticated user.

```
$ curl \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
https://openam.example.com:8443/openam/json/realms/root/dashboard/assigned
{
  "google": {
    "dashboardIcon": [
      "Google.gif"
    ],
    "dashboardName": [
      "Google"
    ],
    "dashboardLogin": [
      "http://www.google.com"
    ],
    "ICFIdentifier": [
      ""
    ],
    "dashboardDisplayName": [
      "Google"
    ],
    "dashboardClassName": [
      "SAML2ApplicationClass"
    ]
  }
}
```

### **/dashboard/available**

This endpoint retrieves the list of applications available in the authenticated user's realm. The example is based on two of the default Dashboard applications: Google and Salesforce.

```
$ curl \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
https://openam.example.com:8443/openam/json/realms/root/dashboard/available
{
  "google": {
    "dashboardIcon": [
      "Google.gif"
    ],
    "dashboardName": [
      "Google"
    ],
    "dashboardLogin": [
      "http://www.google.com"
    ]
  }
}
```

```

    },
    "ICFIdentifier": [
        ""
    ],
    "dashboardDisplayName": [
        "Google"
    ],
    "dashboardClassName": [
        "SAML2ApplicationClass"
    ]
}
"salesforce": {
    "dashboardIcon": [
        "salesforce.gif"
    ],
    "dashboardName": [
        "Salesforce"
    ],
    "dashboardLogin": [
        "http://salesforce.com"
    ],
    "ICFIdentifier": [
        ""
    ],
    "dashboardDisplayName": [
        "Salesforce"
    ],
    "dashboardClassName": [
        "SAML2ApplicationClass"
    ]
}
}
}

```

### **/dashboard/defined**

This endpoint retrieves the list of all applications available defined for the OpenAM Dashboard service. The example is based on the three default Dashboard applications: Google, Salesforce, and Zendesk.

```

$ curl \
--header "iplanetDirectoryPro: AQIC5w..2NzEz*" \
https://openam.example.com:8443/openam/json/realms/root/dashboard/defined
{
  "google": {
    "dashboardIcon": [
      "Google.gif"
    ],
    "dashboardName": [
      "Google"
    ],
    "dashboardLogin": [
      "http://www.google.com"
    ],
    "ICFIdentifier": [
      "idm magic 34"
    ],
    "dashboardDisplayName": [
      "Google"
    ]
  }
}

```

```

    ],
    "dashboardClassName": [
        "SAML2ApplicationClass"
    ]
},
"salesforce": {
    "dashboardIcon": [
        "salesforce.gif"
    ],
    "dashboardName": [
        "SalesForce"
    ],
    "dashboardLogin": [
        "http://www.salesforce.com"
    ],
    "ICFIdentifier": [
        "idm magic 12"
    ],
    "dashboardDisplayName": [
        "Salesforce"
    ],
    "dashboardClassName": [
        "SAML2ApplicationClass"
    ]
},
"zendesk": {
    "dashboardIcon": [
        "ZenDesk.gif"
    ],
    "dashboardName": [
        "ZenDesk"
    ],
    "dashboardLogin": [
        "http://www.ZenDesk.com"
    ],
    "ICFIdentifier": [
        "idm magic 56"
    ],
    "dashboardDisplayName": [
        "ZenDesk"
    ],
    "dashboardClassName": [
        "SAML2ApplicationClass"
    ]
}
}

```

If your application runs in a user-agent such as a browser, you can rely on OpenAM to handle authentication.

## Chapter 8

# Maintaining an Instance

This chapter provides a how-to approach to complete common maintenance tasks.

## 8.1. Backing Up and Restoring Configurations

OpenAM stores configuration data in an LDAP directory server and in files. The directory service replicates configuration data between directory servers, allowing OpenAM to share configuration data across servers in a site. During normal production operations, you rely on directory replication to maintain multiple, current copies of OpenAM service configuration. To recover from the loss of a server or from a serious administrative error, back up directory data and configuration files.

This section shows how to backup and restore OpenAM configuration data by backing up and restoring local configuration files and local (embedded) configuration directory server data. If your deployment uses an external configuration directory server, then refer to the documentation for your external directory server or work with your directory server administrator to back up and restore configuration data stored in the external directory service.

For OpenDJ directory server you can find more information in the chapter on *Backing Up and Restoring Data*.

This section aims to cover the following uses of backup data.

1. Recovery from server failure:
  - Procedure 8.1, "To Back Up All Server Configuration Data"
  - Procedure 8.2, "To Restore All Server Configuration Data"
2. Recovery from serious administrative error:
  - Procedure 8.3, "To Export Only Configuration Data"
  - Procedure 8.4, "To Restore Configuration Data After Serious Error"

### *Procedure 8.1. To Back Up All Server Configuration Data*

This procedure backs up all the configuration data stored with the server. This backup is to be restored when rebuilding a failed server.

Have the following points in mind when using this procedure:



- Use this procedure *only* when OpenAM stores configuration data in the embedded OpenDJ directory server, which means that the embedded OpenDJ directory server files are co-located with other OpenAM configuration files.

If your deployment uses an external configuration directory server, then refer to the documentation for your external directory server or work with your directory server administrator to back up and restore configuration data stored in the external directory service. For OpenDJ directory server you can find more information in the chapter on *Backing Up and Restoring Data*.

- Do not restore configuration data from a backup of a different release of OpenAM. The structure of the configuration data can change from release to release.
- In OpenAM deployments where configuration directory data is replicated, you must take the following points into consideration:
  - Directory replication mechanically applies new changes to ensure that replicated data is the same everywhere. When you restore older backup data, directory replication applies newer changes to the older data.

This includes new changes that the administrator sees as mistakes. To recover from administrative error, you must work around this behavior either by performing a change to be replicated that repairs the error or by restoring all replicas to a state prior to the error.

- When preparing directory server backup and restore operations, also know that data replication purge operations affect the useful lifetime of any data that you back up.

Replication relies on historical data to resolve any conflicts that arise. If directory servers did not eventually purge this historical data, the data would continue to grow until it filled all available space. Directory servers therefore purge older historical data. OpenDJ purges historical data older than 3 days by default.

When the directory server encounters a gap in historical data it cannot correctly complete replication operations. You must make sure, therefore, that any data you restore from backup is not older than the replication purge delay. Otherwise your restoration operation could break replication with the likely result that you must restore all servers from backup, losing any changes that occurred in the meantime.

For more information about purge delay, see the OpenDJ *Administration Guide* section on *To Restore a Replica*.

Follow these steps for each OpenAM server that you want to back up:

1. Stop OpenAM or the container in which it runs.
2. Back up OpenAM configuration files including those of the configuration directory server but skipping log and lock files.

The following example uses the default configuration location. \$HOME is the home directory of the user who runs the web container where OpenAM is deployed, and OpenAM is deployed in Apache Tomcat under `openam`:

```
$ cd $HOME
$ zip \
--recurse-paths \
/path/to/backup-`date -u +i5%F-%m-%S`.zip \
openam .openamcfg/AMConfig_path_to_tomcat_webapps_openam_ \
--exclude openam/openam/debug/* openam/openam/log/* openam/openam/stats* \
openam/opends/logs/* openam/opends/locks/*
...
$ ls /path/to/backup-2014-12-01-12-00.zip
/path/to/backup-2014-12-01-12-00.zip
```

The backup is valid until the end of the purge delay.

3. Start OpenAM or the container in which it runs.

### *Procedure 8.2. To Restore All Server Configuration Data*

This procedure applies when rebuilding a failed server.

Have the following points in mind when using this procedure:

- This procedure restores all the configuration data for a server, where the configuration data has been backed up as described in [Procedure 8.1, "To Back Up All Server Configuration Data"](#).
- Use this procedure *only* when OpenAM stores configuration data in the embedded OpenDJ directory server, which means that the embedded OpenDJ directory server files are co-located with other OpenAM configuration files.

If your deployment uses an external configuration directory server, then refer to the documentation for your external directory server or work with your directory server administrator to back up and restore configuration data stored in the external directory service. For OpenDJ directory server, you can find more information in the chapter on *Backing Up and Restoring Data*.

- Do not restore configuration data from a backup of a different release of OpenAM. The structure of the configuration data can change from release to release.
- If OpenAM also stores CTS data in the embedded OpenDJ directory server, then the restore operation overwrites current CTS data with older data. After you restore from backup, users authenticate again as necessary.
- In OpenAM deployments where configuration directory data is replicated, you must take the following points into consideration:
  - Directory replication mechanically applies new changes to ensure that replicated data is the same everywhere. When you restore older backup data, directory replication applies newer changes to the older data.

This includes new changes that the administrator sees as mistakes. To recover from administrative error, you must work around this behavior either by performing a change to be replicated that repairs the error or by restoring all replicas to a state prior to the error.

- When preparing directory server backup and restore operations, also know that data replication purge operations affect the useful lifetime of any data that you back up.

Replication relies on historical data to resolve any conflicts that arise. If directory servers did not eventually purge this historical data, the data would continue to grow until it filled all available space. Directory servers therefore purge older historical data. OpenDJ purges historical data older than 3 days by default.

When the directory server encounters a gap in historical data it cannot correctly complete replication operations. You must make sure, therefore, that any data you restore from backup is not older than the replication purge delay. Otherwise your restoration operation could break replication with the likely result that you must restore all servers from backup, losing any changes that occurred in the meantime.

For more information about purge delay, see the OpenDJ *Administration Guide* section on *To Restore a Replica*.

Follow these steps for each OpenAM server to restore. If you are restoring OpenAM after a failure, make sure you make a copy of any configuration and log files that you need to investigate the problem before restoring OpenAM from backup:

1. Stop OpenAM or the container in which it runs.
2. Restore files in the configuration directory as necessary, making sure that you restore from a valid backup, one that is newer than the replication purge delay:

```
$ cd $HOME
$ unzip /path/to/backup-2014-12-01-12-00.zip
Archive:  /path/to/backup-2014-12-01-12-00.zip
replace openam/.configParam? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
...
```

3. Start OpenAM or the container in which it runs.

### Procedure 8.3. To Export Only Configuration Data

LDAP Data Interchange Format (LDIF) is a standard, text-based format for storing LDAP directory data. You can use LDIF excerpts to make changes to directory data.

This procedure takes an LDIF backup of OpenAM configuration data only. Use this LDIF data when recovering from a serious configuration error:

1. Make sure that OpenAM's configuration is in correct working order before exporting configuration data.
2. Use the OpenDJ **export-ldif** command to run a task that exports only configuration data, not CTS data.

You can run this command without stopping OpenAM.

Find OpenDJ commands under the file system directory that contains OpenAM configuration files.

The bind password for Directory Manager is the same as the password for the OpenAM global administrator (amadmin):

```
$ $HOME/openam/opens/bin/export-ldif \
--port 4444 \
--hostname openam.example.com \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--backendID userRoot \
--includeBranch dc=openam,dc=forgerock,dc=org \
--excludeBranch ou=tokens,dc=openam,dc=forgerock,dc=org \
--ldifFile /path/to/backup-`date -u +%F-%m-%S`.ldif \
--start 0 \
--trustAll
Export task 20141208113331302 scheduled to start Dec 8, 2014 11:33:31 AM CET
```

When the task completes, the LDIF file is at the expected location:

```
$ ls /path/to/*.ldif
/path/to/backup-2014-12-08-12-1418034808.ldif
```

#### Procedure 8.4. To Restore Configuration Data After Serious Error

A serious configuration error is an error that you cannot easily repair by using OpenAM configuration tools, such as the AM console or the **ssoadm** command.

Use this procedure to recover from a serious configuration error by manually restoring configuration data to an earlier state. This procedure depends on LDIF data that you exported as described in Procedure 8.3, "To Export Only Configuration Data".

1. Read the OpenDJ change log to determine the LDAP changes that caused the configuration problem.

The OpenDJ change log provides an external change log mechanism that allows you to read changes made to directory data for replicated directory servers.

For instructions on reading the change log, see the *OpenDJ Administration Guide* section on *Change Notification For Your Applications*.

2. Based on the data in the change log, determine what changes would reverse the configuration error.

For changes that resulted in one attribute value being replaced by another, you can recover the information from the change log alone.

3. For deleted content not contained in the change log, use the LDIF resulting from Procedure 8.3, "To Export Only Configuration Data" to determine a prior, working state of the configuration entry before the configuration error.

4. Prepare LDIF to modify configuration data in a way that repairs the error by restoring the state of directory entries before the administrative error.
5. Use the OpenDJ **ldapmodify** command to apply the modification.

For instructions on making changes to directory data see the section on *Updating the Directory* in the *OpenDJ Directory Server Developer's Guide*.

## 8.2. Changing the amadmin User's Password

The built-in **amadmin** account cannot be disabled, deleted, or renamed, since it is hard-coded in the source code of several files.

If you want a user to have administration rights in OpenAM other than **amadmin**, delegate realm administration privileges to the new user. For more information about delegating realm administration privileges, see Section 2.4.1, "Delegating Realm Administration Privileges".

In this section you will find procedures to change the password of the top-level administrator **amadmin**, when:

- OpenAM is configured using an external configuration store.

See Procedure 8.5, "To Change the amadmin User's Password: External Configuration Store".

- OpenAM is configured using the embedded OpenDJ instance as the configuration store. It may be configured with an external data store, or with the embedded OpenDJ instance as a data store.

See Procedure 8.6, "To Change the amadmin User's Password: Embedded Configuration Store".

### *Procedure 8.5. To Change the amadmin User's Password: External Configuration Store*

If OpenAM is configured to use an external configuration store, perform the following steps to change the **amadmin** user's password:

1. Log in to the AM console as the administrator, **amadmin**.
2. Navigate to Realms > Top Level Realm > Subjects, and then click **amAdmin**.
3. On the Edit User page, select Edit next to Password.
4. On the Change Password page, enter the new password in the New Password field.
5. Click OK to save your changes.

If your deployment has multiple OpenAM servers, the new password replicates across all servers.

### Procedure 8.6. To Change the amadmin User's Password: Embedded Configuration Store

If OpenAM is configured to use the embedded OpenDJ instance for the configuration store, you must change the passwords of the following two users in the embedded OpenDJ accounts to match the new `amadmin` password:

You must change these two passwords in the embedded OpenDJ instance regardless of whether you use an external or embedded data store.

- The `cn=Directory Manager` user, created during installation.
- The global administrator, created in OpenDJ by OpenAM after a second OpenAM server has been added to the deployment.

Some functionality might not work if the OpenDJ directory manager, OpenAM administrator `amadmin`, and OpenDJ global administrator passwords are not identical. For example, adding new servers to the deployment.

To change the OpenAM `amadmin`, OpenDJ directory manager, and OpenDJ global administrator passwords and the required bindings, perform the following steps:

1. Back up your deployment as described in Section 8.1, "Backing Up and Restoring Configurations".
2. Log in to the AM console as the administrator, `amadmin`.
3. Navigate to Realms > Top Level Realm > Subjects, and then click `amAdmin`.
4. On the Edit User page, select Edit next to Password.
5. On the Change Password page, enter the new password in the New Password field.
6. Click OK to save your changes.

If your deployment has multiple OpenAM servers, the new password replicates across all servers.

7. OpenAM binds to the embedded OpenDJ server using the `cn=Directory Manager` account. Change the `cn=Directory Manager` account's bind password in the OpenAM configuration as follows:
  - a. Change the password for the configuration store binding:
    - i. Navigate to Deployment > Servers > *Server Name* > Directory Configuration.
    - ii. Enter the new bind password, which is the new `amadmin` password, and save your changes.

Make this change for each of your OpenAM servers.

#### Note

Changing the bind password of the configuration store updates the `configstorepwd` alias in the `keystore.jceks` keystore file automatically.

For more information, see Section 5.4, "Configuring Password String Aliases".

- b. (Optional) If you use the embedded OpenDJ instance as a data store, change the following bind passwords:
  - i. Navigate to Realms > *Realm Name* > Data Stores > embedded:
    - Enter the new bind password, which is the new **amadmin** password, and save your changes.  
  
Make this change in every OpenAM realm that uses the embedded OpenDJ as a data store.
  - ii. Navigate to Realms > *Realm Name* > Services > Policy Configuration:
    - Enter the new bind password, which is the new **amadmin** password, and save your changes.  
  
Make this change in every OpenAM realm that uses the embedded OpenDJ as a data store.
  - iii. Navigate to Realms > *Realm Name* > Authentication > Modules, and select LDAP:
    - Enter the new bind password, which is the new **amadmin** password, and save your changes.  
  
Make this change in every OpenAM realm that uses the embedded OpenDJ as a data store.
8. To change the **cn=Directory Manager** and the global administrator passwords in the embedded OpenDJ, see [Resetting Administrator Passwords in the \*OpenDJ Administration Guide\*](#).

## 8.3. Monitoring Services

This section covers how to monitor services to ensure appropriate performance and service availability.

### 8.3.1. Monitoring Interfaces

OpenAM lets you monitor OpenAM over protocol through web pages, Java Management Extensions (JMX), or Simple Network Management Protocol (SNMP). The services are based on JMX.

To configure monitoring services, login to the AM console as OpenAM administrator, and navigate to Configure > System, and then click Monitoring. Alternatively, you can use the **ssoadm set-attr-defs** command:

```
$ ssoadm \
  set-attr-defs \
    --servicename iPlanetAMMonitoringService \
    --schematype Global \
    --adminid amadmin \
    --password-file /tmp/pwd.txt \
    --attributevalues iplanet-am-monitoring-enabled=true
```

Restart OpenAM for the changes to take effect. You must also restart OpenAM if you disable monitoring.

### 8.3.1.1. Web-Based Monitoring

You can configure OpenAM to allow you to access a web based view of OpenAM MBeans on port 8082 where the core server runs, such as <http://openam-ter.example.com:8082/>. Either use the console, or use the **ssoadm** command:

```
$ ssoadm \
  set-attr-defs \
    --servicename iPlanetAMMonitoringService \
    --schematype Global \
    --adminid amadmin \
    --password-file /tmp/pwd.txt \
    --attributevalues iplanet-am-monitoring-http-enabled=true
```

The default authentication file allows you to authenticate over HTTP as user **demo**, password **changeit**. The user name and password are kept in the file specified, with the password encrypted:

```
$ cat openam/openam/openam_mon_auth
demo AQICMBCKLwx6G3vzK3TYRbtTpNYAagVIPNP
```

Or:

```
$ cat openam/openam/opensso_mon_auth
demo AQICvSe+tXEg8TUUT8ekzHb8IRzVSvm1Lc2u
```

You can encrypt a new password using the **ampassword** command. After changing the authentication file, you must restart OpenAM for the changes to take effect.



Figure 8.1. MBeans in a Browser

## MBean View

- MBean Name:**  
SUN\_OPENSSO\_SERVER\_MIB/ssoServerServerTable:ssoServerServerEntry.serverHostName=openam-ter.example.com,ssoServerServerEntry.serverPort=8080
- MBean Java Class:** com.sun.identity.monitoring.SsoServerServerEntryImpl

[Project OpenDMKopendmk-1.0-b02]

Reload Period in seconds:

[Back to Agent View](#)

---

**MBean description:**

Information on the management interface of the MBean

---

**List of MBean attributes:**

Name	Type	Access	Value
<a href="#">ServerHostName</a>	java.lang.String	RO	openam-ter.example.com
<a href="#">ServerId</a>	java.lang.Integer	RO	1
<a href="#">ServerPort</a>	java.lang.Integer	RO	8080
<a href="#">ServerProtocol</a>	java.lang.String	RO	http
<a href="#">ServerStatus</a>	java.lang.Integer	RO	1

### 8.3.1.2. JMX Monitoring

You can configure OpenAM to allow you to listen for Java Management eXtension (JMX) clients, by default on port 9999. Either use the OpenAM console page under Configure > Global Services > System > Monitoring and make sure both Monitoring Status and Monitoring RMI interface status are both set to Enabled, or use the **ssoadm** command:

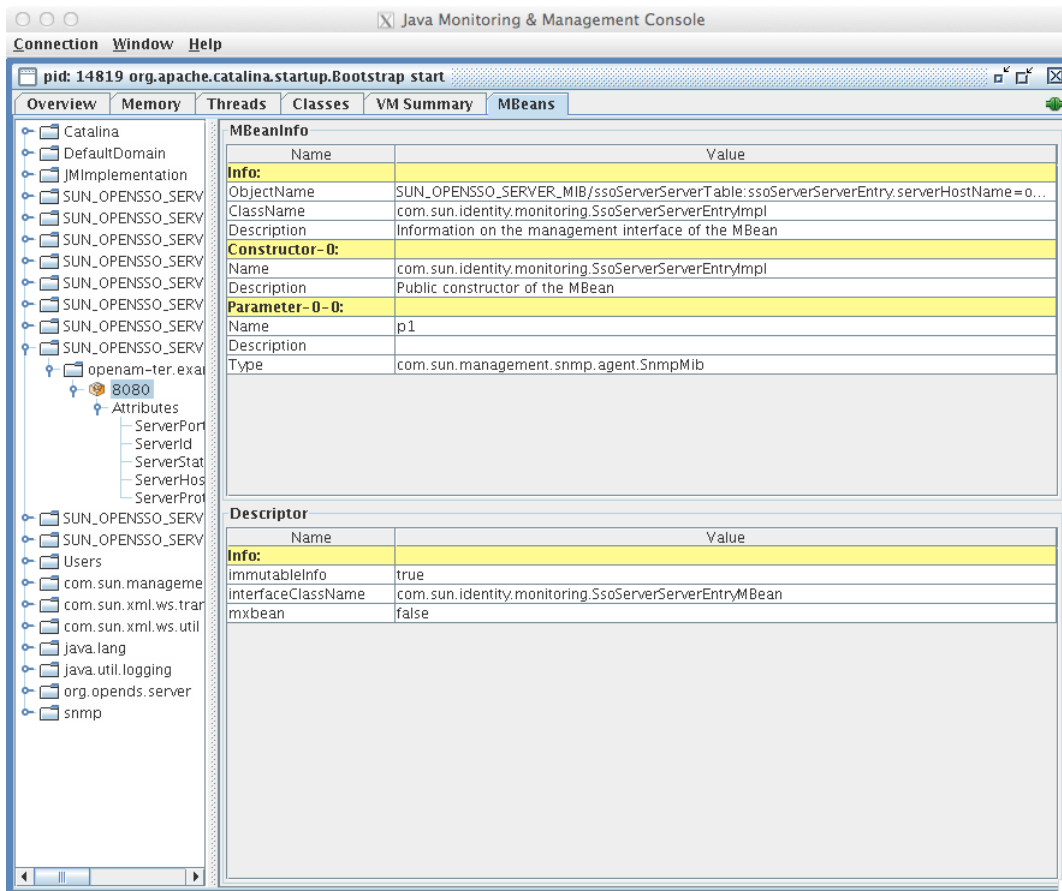
```
$ ssoadm \
  set-attr-defs \
  --servicename iPlanetAMMonitoringService \
  --schematype Global \
  --adminid amadmin \
  --password-file /tmp/pwd.txt \
  --attributevalues iplanet-am-monitoring-enabled=true \
  iplanet-am-monitoring-rmi-enabled=true
```

A number of tools support JMX, including **jvisualvm** and **jconsole**. When you use **jconsole** to browse OpenAM MBeans for example, the default URL for the OpenAM running on the local system is `service:jmx:rmi:///jndi/rmi://localhost:9999/server`.

```
$ jconsole service:jmx:rmi:///jndi/rmi://localhost:9999/server &
```

You can also browse the MBeans by connecting to your web application container, and browsing to the OpenAM MBeans. By default, JMX monitoring for your container is likely to be accessible only locally, using the process ID.

Figure 8.2. JConsole Browsing MBeans



Also see [Monitoring and Management Using JMX](#) for instructions on how to connect remotely, how to use SSL, and so forth.

### Important

JMX has a limitation in that some Operations and CTS tables cannot be properly serialized from OpenAM to JMX. As a result, only a portion of OpenAM's monitoring information is available through JMX. SNMP is a

preferred monitoring option over JMX and exposes all OpenAM tables, especially for CTS, with no serialization limitations.

### 8.3.1.3. SNMP Monitoring

You can configure OpenAM to allow you to listen on port 8085 for SNMP monitoring. Either use the AM console, or use the **ssoadm** command:

```
$ ssoadm \
  set-attr-defs \
    --servicename iPlanetAMMonitoringService \
    --schematype Global \
    --adminid amadmin \
    --password-file /tmp/pwd.txt \
    --attributevalues iplanet-am-monitoring-snmp-enabled=true
```

### 8.3.2. Monitoring CTS Tokens

The Core Token Service (CTS) provides persistent and highly available token storage for a several components within OpenAM, including user sessions, OAuth 2.0, SAML v2.0 and UMA tokens. For information on configuring the Core Token Service, see [Chapter 3, "Implementing the Core Token Service"](#) in the *Installation Guide*.

Depending on system load and usage, the CTS can produce a large quantity of tokens, which can be short lived. This style of usage is significantly different from typical LDAP usage. As such, systems administrators may be interested in monitoring this usage as part of LDAP directory maintenance.

The CTS functions only with one external LDAP service, OpenDJ.

To that end, the current state of CTS tokens on a system can be monitored over SNMP. The current state of different types of CTS tokens are associated with different Object Identifiers (OIDs) in a Management Information Base (MIB).

To enable SNMP, see [Section 8.3.1.3, "SNMP Monitoring"](#)

#### 8.3.2.1. CTS SNMP Monitoring

Once activated, SNMP monitoring works over UDP by default. You may want to install one of many available network monitoring tools. For the purpose of this section, basic SNMP service and monitoring tools have been installed on a GNU/Linux system. The same commands should work on a Mac OS X system.

SNMP depends on labels known as Object Identifiers (OIDs). These are uniquely defined labels, organized in tree format. For OpenAM, they are configured in a Management Information Base file named **FORGEROCK-OPENAM-CTS.mib**.

For detailed information on configured OIDs, see [Section 7.1, "Core Token Service \(CTS\) Object Identifiers"](#) in the *Installation Guide*.

With the OIDs in hand, you can set up an SNMP server to collect the data. You would also need SNMP utility commands with associated OIDs to measure the current state of a component. First, to verify the operation of SNMP on a GNU/Linux system, over port 8085, using SNMP version 2c, run the following command:

```
# snmpstatus -c public -v 2c localhost
```

The output should normally specify communications over UDP. If you get a **timeout** message, the SNMP service may not be running.

You can get the value for a specific OID. For example, the following command would retrieve the cumulative count for CTS create operations, over port 8085:

```
# snmpget -c public -v 2c :8085 enterprises.36733.1.2.3.1.1.1
```

For one view of the tree of OIDs, you can use the **snmpwalk** command. For example, the following command lists all OIDs related to CTS:

```
# snmpwalk -c public -v 2c :8085 enterprises.36733.1.2.3
```

A number of CTS OIDs are listed with a **Counter64** value. As defined in *RFC 2578*, an OID so configured has a maximum value of  $2^{64} - 1$ .

### 8.3.2.2. SNMP Monitoring for Policy Evaluation

You can monitor policy evaluation performance over SNMP. OpenAM records statistics for up to a number of recent policy evaluation requests. (You can configure the number in the AM console under Configuration > System > Monitoring. For details, see the reference section Section 10.7, "Monitoring".

Interface Stability: *Evolving* in the *Release Notes*

As described in Section 8.3.2.1, "CTS SNMP Monitoring", SNMP uses OIDs defined in a Management Information Base (MIB) file that specifies the statistics OpenAM keeps for policy evaluation operations, **FORGEROCK-OPENAM-POLICY.mib**. Adapt the examples in Section 8.3.2.1, "CTS SNMP Monitoring" to read monitoring statistics about policy evaluation on the command line.

When monitoring is active, OpenAM records statistics about both the numbers and rates of policy evaluations performed, and also the times taken to process policy evaluations.

The statistics are all read-only. The base OID for policy evaluation statistics is **enterprises.36733.1.2.2.1**. The following table describes the values that you can read:

*Table 8.1. OIDs Used in SNMP Monitoring For Policy Evaluation*

OID	Description	Syntax
<a href="#">enterprises.36733.1.2.2.1.1.1</a>	Cumulative number of policy evaluations for specific resources (self)	Counter64
<a href="#">enterprises.36733.1.2.2.1.1.2</a>	Average rate of policy evaluations for specific resources (self)	Counter64
<a href="#">enterprises.36733.1.2.2.1.1.3</a>	Minimum rate of policy evaluations for specific resources (self)	Counter64
<a href="#">enterprises.36733.1.2.2.1.1.4</a>	Maximum rate of policy evaluations for specific resources (self)	Counter64
<a href="#">enterprises.36733.1.2.2.1.2.1</a>	Cumulative number of policy evaluations for a tree of resources (subtree)	Counter64
<a href="#">enterprises.36733.1.2.2.1.2.2</a>	Average rate of policy evaluations for a tree of resources (subtree)	Counter64
<a href="#">enterprises.36733.1.2.2.1.2.3</a>	Minimum rate of policy evaluations for a tree of resources (subtree)	Counter64
<a href="#">enterprises.36733.1.2.2.1.2.4</a>	Maximum rate of policy evaluations for a tree of resources (subtree)	Counter64
<a href="#">enterprises.36733.1.2.2.1.2.1.1</a>	Average length of time to evaluate a policy for a specific resource (self)	Counter64
<a href="#">enterprises.36733.1.2.2.1.2.1.2</a>	Slowest evaluation time for a specific resource (self)	SnmpAdminString
<a href="#">enterprises.36733.1.2.2.1.2.2.1</a>	Average length of time to evaluate a policy for a tree of resources (subtree)	Counter64
<a href="#">enterprises.36733.1.2.2.1.2.2.2</a>	Slowest evaluation time for a tree of resources (subtree)	SnmpAdminString
<a href="#">enterprises.36733.1.2.2.1.3.1</a>	Slowest individual policy evaluation time overall	SnmpAdminString

### 8.3.2.3. SNMP Monitoring for Sessions

You can monitor stateful session statistics over SNMP. OpenAM records statistics for up to a configurable number of recent sessions. (You can configure the number in the AM console under Configuration > System > Monitoring. For details, see the system configuration reference section, Section 2.2.12, "Monitoring" in the *Reference*.)

SNMP monitoring is not available for stateless sessions.

Interface Stability: Evolving in the *Release Notes*

As described in Section 8.3.2.1, "CTS SNMP Monitoring", SNMP uses OIDs defined in a Management Information Base (MIB) file that specifies the statistics OpenAM keeps for policy evaluation operations, `FORGEROCK-OPENAM-SESSION.mib`. Adapt the examples in Section 8.3.2.1, "CTS SNMP Monitoring" to read monitoring statistics about sessions on the command line.

When monitoring is active, OpenAM records statistics about both the numbers of internal, remote, and CTS sessions, and also the times taken to process sessions.

The statistics are all read-only. The base OID for session statistics is `enterprises.36733.1.2.1`. Times are expressed in nanoseconds rather than milliseconds, as many operations take less than one millisecond. The following table describes the values that you can read:

*Table 8.2. OIDs Used in SNMP Monitoring For Sessions*

OID	Description	Syntax
<code>enterprises.36733.1.2.1.1.1</code>	Total number of current internal sessions	Counter64
<code>enterprises.36733.1.2.1.1.2</code>	Average time it takes to refresh an internal session	Counter64
<code>enterprises.36733.1.2.1.1.3</code>	Average time it takes to logout an internal session	Counter64
<code>enterprises.36733.1.2.1.1.4</code>	Average time it takes to destroy an internal session	Counter64
<code>enterprises.36733.1.2.1.1.5</code>	Average time it takes to set a property on an internal session	Counter64
<code>enterprises.36733.1.2.1.2.1</code>	Total number of current remote sessions	Counter64
<code>enterprises.36733.1.2.1.2.2</code>	Average time it takes to refresh a remote session	Counter64
<code>enterprises.36733.1.2.1.2.3</code>	Average time it takes to logout a remote session	Counter64
<code>enterprises.36733.1.2.1.2.4</code>	Average time it takes to destroy a remote session	Counter64
<code>enterprises.36733.1.2.1.2.5</code>	Average time it takes to set a property on a remote session	Counter64
<code>enterprises.36733.1.2.1.3.1</code>	Total number of sessions currently in the Core Token Service (CTS)	Counter64
<code>enterprises.36733.1.2.1.3.2</code>	Average time it takes to refresh a CTS session	Counter64
<code>enterprises.36733.1.2.1.3.3</code>	Average time it takes to logout a CTS session	Counter64

OID	Description	Syntax
<code>enterprises.36733.1.2.1.3.4</code>	Average time it takes to destroy a CTS session	<code>Counter64</code>
<code>enterprises.36733.1.2.1.3.5</code>	Average time it takes to set a property on a CTS session	<code>Counter64</code>

## 8.4. Tuning an Instance

This section covers key OpenAM tunings to ensure smoothly performing access and federation management services, and to maximize throughput while minimizing response times.

### Note

The recommendations provided here are guidelines for your testing rather than hard and fast rules for every situation. Said another way, the fact that a given setting is configurable implies that no one setting is right in all circumstances.

The extent to which performance tuning advice applies depends to a large extent on your requirements, on your workload, and on what resources you have available. Test suggestions before rolling them out into production.

The suggestions in this section pertain to OpenAM deployments with the following characteristics:

- The deployment has a dedicated OpenDJ directory server for the Core Token Service. The host running this directory server is a high-end server with a large amount of memory and multiple CPUs.
- The OpenAM server is configured to use stateful sessions.

### 8.4.1. Tuning Server Settings

OpenAM has a number of settings that can be tuned to increase performance.

#### 8.4.1.1. General Settings

The following general points apply:

- Set debug level to `error`.
- Set container-level logging to a low level, such as `error` or `severe`.

#### 8.4.1.2. LDAP Settings

Tune your LDAP data stores, your LDAP authentication modules, and connection pools for CTS and configuration stores.

#### 8.4.1.2.1. Tuning LDAP Data Store Settings

To change LDAP data store settings, navigate to Realms > *Realm Name* > Data Stores > *Data Store Name* in the AM console. Each data store has its own connection pool and therefore each data store needs its own tuning:

*Table 8.3. LDAP Data Store Settings*

Property	Default Value	Suggestions
LDAP Connection Pool Minimum Size	1	The minimum LDAP connection pool size; a good tuning value for this property is 10.  ( <a href="#">sun-idrepo-ldapv3-config-connection_pool_min_size</a> )
LDAP Connection Pool Maximum Size	10	The maximum LDAP connection pool size; a high tuning value for this property is 65, though you might well be able to reduce this for your deployment. Ensure your LDAP server can cope with the maximum number of clients across all the OpenAM servers.  ( <a href="#">sun-idrepo-ldapv3-config-connection_pool_max_size</a> )

#### 8.4.1.2.2. Tuning LDAP Authentication Module Settings

To change connection pool settings for the LDAP authentication module, in the OpenAM console, navigate to Configure > Authentication, and then click Core Attributes.

*Table 8.4. LDAP Authentication Module Setting*

Property	Default Value	Suggestions
Default LDAP Connection Pool Size	1:10	The minimum and maximum LDAP connection pool used by the LDAP authentication module. This should be tuned to 10:65 for production.  ( <a href="#">iplanet-am-auth-ldap-connection-pool-default-size</a> )

#### 8.4.1.2.3. Tuning LDAP CTS and Configuration Store Settings

When tuning LDAP connection pool settings for the Core Token Service (CTS), what you change depends on whether the directory service backing the CTS is the same directory service backing OpenAM configuration.

When the same directory service backs both the CTS and also OpenAM configuration (the default), then the same connection pool is shared for any LDAP operations requested by the CTS or by a service accessing the OpenAM configuration. In this case, one connection is reserved for cleanup of expired CTS tokens. Roughly half of the connections are allocated for CTS operations, to the  
To be precise, the number of connections allocated for CTS operations is equal to the power of two that is nearest to half the maximum number of connections in the pool.



nearest power of two.<sup>1</sup> The remaining connections are allocated to services accessing the OpenAM configuration. For a default configuration, where the maximum number of connections in the pool is ten, one connection is allocated for cleanup of expired CTS tokens, four connections are allocated for other CTS operations, and five connections are allocated for services accessing the configuration. If the Maximum Connection Pool size is 20, one connection is allocated for cleanup of expired CTS tokens, eight connections are allocated for other CTS operations, and 11 connections are allocated for services accessing the configuration. If the pool size is 65, then the numbers are 1, 32, and 32, and so on.

The minimum number of connections is 6.

When the directory service backing the CTS is external (differs from the directory service backing the OpenAM configuration) then the connection pool used to access the directory service for the CTS is separate from the pool used to access the directory service for the OpenAM configuration. One connection is reserved for cleanup of expired CTS tokens. Remaining connections are allocated for CTS operations such that the number of connections allocated is equal to a power of two. In this case, set the maximum number of connections to  $2^{n+1}$ , as in 9, 17, 33, 65, and so forth.

If the same directory service backs both the CTS and also OpenAM configuration, then set the Maximum Connection Pool property size under Deployment > Servers > *Server Name* > Directory Configuration.

If the directory service backing the CTS is external (differs from the directory service backing the OpenAM configuration), then set the Maximum Connection property size under Deployment > Servers > *Server Name* > CTS > CTS Token Store.

In both cases, if you must change the default connection timeouts, set the following advanced properties under Deployment > Servers > *Server Name* > Advanced:

*Table 8.5. CTS Store LDAP Connection Pool Settings*

Property	Default Value	Suggestions
Maximum Connection Pool	10	Find this setting in the AM console under Deployment > Servers > <i>Server Name</i> > Directory Configuration.  When the same directory service backs both the CTS and also OpenAM configuration, consider increasing this to at least 19 to allow 9 connections for the CTS, and 10 connections for access to the OpenAM configuration (including for example looking up policies).
Max Connections	10	Find this setting in the AM console under Deployment > Servers > <i>Server Name</i> > CTS > External Store Configuration.  When the directory service backing the CTS is external and the load on the CTS is high, consider setting this to $2^{n+1}$ , where $n = 4, 5, 6$ , and so on. In other words, try setting this to 17, 33, 65, and so on when testing performance under load.

Property	Default Value	Suggestions
		( <a href="#">org.forgerock.services.cts.store.max.connections</a> )
CTS connection timeout (advanced property)	10 (seconds)	<p>Most CTS requests to the directory server are handled quickly, so the default timeout is fine for most cases.</p> <p>If you choose to vary this setting for performance testing, set the advanced property, <a href="#">org.forgerock.services.datalayer.connection.timeout.cts.async</a>, under Deployment &gt; Servers &gt; <i>Server Name</i> &gt; Advanced.</p> <p>You must restart OpenAM or the container in which it runs for changes to take effect.</p>
Configuration management connection timeout (advanced property)	10 (seconds)	<p>Most configuration management requests to the directory server are handled quickly, so the default timeout is fine for most cases.</p> <p>If you choose to vary this setting for performance testing, set the advanced property, <a href="#">org.forgerock.services.datalayer.connection.timeout</a>, under Deployment &gt; Servers &gt; <i>Server Name</i> &gt; Advanced.</p> <p>You must restart OpenAM or the container in which it runs for changes to take effect.</p>

### 8.4.1.3. Notification Settings

OpenAM has two thread pools used to send notifications to clients. The Service Management Service (SMS) thread pool can be tuned in the AM console under Configure > Server Defaults > SDK > Data Store:

*Table 8.6. SMS Notification Setting*

Property	Default Value	Suggestions
Notification Pool Size	10	<p>This is the size of the thread pool used to send notifications. In production this value should be fine unless lots of clients are registering for SMS notifications.</p> <p>(<a href="#">com.sun.identity.sm.notification.threadpool.size</a>)</p>

The session service has its own thread pool to send notifications to listeners about changes to stateful sessions. This is configured under Configure > Server Defaults > Session > Notification:

*Table 8.7. Session Service Notification Settings*

Property	Default Value	Suggestions
Notification Pool Size	10	This is the size of the thread pool used to send notifications. In production this should be around 25-30.  ( <code>com.iplanet.am.notification.threadpool.size</code> )
Notification Thread Pool Threshold	5000	This is the maximum number of notifications in the queue waiting to be sent. The default value should be fine in the majority of installations.  ( <code>com.iplanet.am.notification.threadpool.threshold</code> )

#### 8.4.1.4. Session Settings

The Session Service has additional properties to tune, which are configured under Configure > Server Defaults > Session > Session Limits. The following suggestion applies to deployments using stateful sessions:

*Table 8.8. Session Settings*

Property	Default Value	Suggestion
Maximum Session Cache Size	5000	Maximum number of OpenAM sessions to cache on the server.  In production, this value can safely be set into the 100,000s. The maximum session cache size is really controlled by the maximum size of the JVM heap which must be tuned appropriately to match the desired session cache size.  ( <code>org.forgerock.openam.session.service.access.persistence.caching.maxsize</code> )

#### 8.4.2. Tuning Java Virtual Machine Settings

This section gives some initial guidance on configuring the JVM for running OpenAM. These settings provide a strong foundation to the JVM before a more detailed garbage collection tuning exercise, or as best practice configuration for production:

*Table 8.9. Heap Size Settings*

JVM Parameters	Suggested Value	Description
<code>-Xms</code> & <code>-Xmx</code>	At least 1 GB (2 GB with embedded OpenDJ), in production environments at least 2 GB to 3 GB. This setting depends on the available physical memory, and on whether a 32- or 64-bit JVM is used.	-
<code>-server</code>	-	Ensures the server JVM is used
<code>-XX:PermSize</code> & <code>-XX:MaxPermSize</code> (JDK 7)	Set both to 256 MB	Controls the size of the permanent generation in the JVM
<code>-XX:MetaspaceSize</code> & <code>-XX:MaxMetaspaceSize</code> (JDK 8)	Set both to 256 MB	Controls the size of the metaspace in the JVM
<code>-Dsun.net.client.defaultReadTimeout</code>	60000	Controls the read timeout in the Java HTTP client implementation  This applies only to the Sun/Oracle HotSpot JVM.
<code>-Dsun.net.client.defaultConnectTimeout</code>	High setting: 30000 (30 seconds)	Controls the connect timeout in the Java HTTP client implementation  When you have hundreds of incoming requests per second, reduce this value to avoid a huge connection queue.  This applies only to the Sun/Oracle HotSpot JVM.

*Table 8.10. Security Settings*

JVM Parameters	Suggested Value	Description
<code>-Dhttps.protocols</code>	<code>TLSv1,TLSv1.1,TLSv1.2</code>	Controls the protocols used for outbound HTTPS connections from OpenAM.  Specify one or more of the following values, separated by commas:

JVM Parameters	Suggested Value	Description
		<ul style="list-style-type: none"> <li>• TLSv1</li> <li>• TLSv1.1</li> <li>• TLSv1.2</li> </ul> <p>This setting applies only to Sun/Oracle Java environments.</p>
<code>-Dorg.forgerock.openam.ldap.secure.protocol.version</code>	-	<p>Controls the protocol OpenAM uses to connect to various external resources.</p> <p>Specify one or more of the following values, separated by commas:</p> <ul style="list-style-type: none"> <li>• TLSv1</li> <li>• TLSv1.1</li> <li>• TLSv1.2</li> </ul>

#### Note

For `-Dhttps.protocols`, specify the protocol version(s) Java clients can use to connect to OpenAM.

For `-Dorg.forgerock.openam.ldap.secure.protocol.version`, see Section 4.4, "Securing Communications" in the *Installation Guide* for a list of external resources to which communication is affected.

Specify a single protocol if OpenAM will only use that protocol when connecting to affected external resources. For example, a value of `TLSv1.2` configures OpenAM to only use the TLSv1.2 protocol to connect.

Specify a comma-separated list with multiple protocols if OpenAM will use the most secure protocol supported by the external resources. For example, a value of `TLSv1,TLSv1.1,TLSv1.2` configures OpenAM to attempt to use the TLSv1.2 protocol to connect to external configuration and user data stores. If a TLSv1.2 connection is not supported, OpenAM attempts to use TLSv1.1 to connect. If TLSv1.1 is not supported, OpenAM uses TLSv1.

*Table 8.11. Garbage Collection Settings*

JVM Parameters	Suggested Value	Description
<code>-verbose:gc</code>	-	Verbose garbage collection reporting
<code>-Xloggc:</code>	<code>\$CATALINA_HOME/logs/gc.log</code>	Location of the verbose garbage collection log file
<code>-XX:+PrintClassHistogram</code>	-	Prints a heap histogram when a SIGTERM signal is received by the JVM
<code>-XX:+PrintGCDetails</code>	-	Prints detailed information about garbage collection
<code>-XX:+PrintGCTimeStamps</code>	-	Prints detailed garbage collection timings

JVM Parameters	Suggested Value	Description
<code>-XX:+HeapDumpOnOutOfMemoryError</code>	-	Out of Memory errors generate a heap dump automatically
<code>-XX:HeapDumpPath</code>	<code>\$CATALINA_HOME/logs/heapdump.hprof</code>	Location of the heap dump
<code>-XX:+UseConcMarkSweepGC</code>	-	Use the concurrent mark sweep garbage collector
<code>-XX:+UseCMSCompactAtFullCollection</code>	-	Aggressive compaction at full collection
<code>-XX:+CMSClassUnloadingEnabled</code>	-	Allow class unloading during CMS sweeps

### 8.4.3. Tuning Caching

OpenAM caches data to avoid having to query user and configuration data stores each time it needs the information. By default, OpenAM makes use of LDAP persistent search to receive notification of changes to cached data. For this reason, caching works best when data are stored in a directory server that supports LDAP persistent search.

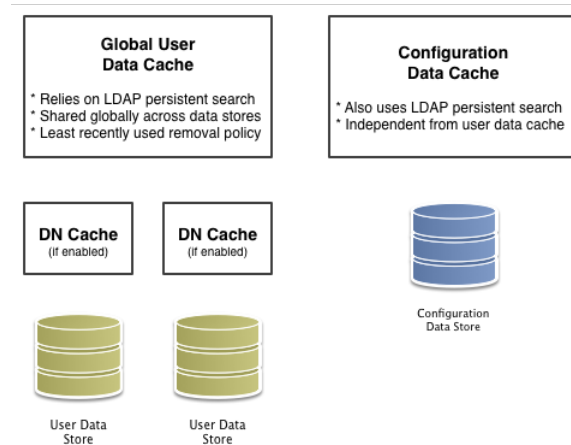
OpenAM has two kinds of cache on the server side that you can configure, one for configuration data and the other for user data. Generally use the default settings for configuration data cache. This section mainly covers the configuration choices you have for caching user data.

OpenAM implements the global user data cache for its user data stores. Prior to OpenAM 11.0, OpenAM supported a secondary Time-to-Live (TTL) data store caching layer, which has since been removed in OpenAM 11.0 and later versions.

The user data store also supports a DN Cache, used to cache DN lookups that tend to occur in bursts during authentication. The DN Cache can become out of date when a user is moved or renamed in the underlying LDAP store, events that are not always reflected in a persistent search result. You can enable the DN cache when the underlying LDAP store supports persistent search and `mod DN` operations (that is, move or rename DN).

The following diagram depicts the two kinds of cache, and also the two types of caching available for user data:

Figure 8.3. Caches



The rest of this section concerns mainly settings for global user data cache and for SDK clients. For a look at data store cache settings, see Table 8.3, "LDAP Data Store Settings".

### 8.4.3.1. Overall Server Cache Settings

By default OpenAM has caching enabled both for configuration data and also for user data. This setting is governed by the server property `com.iplanet.am.sdk.caching.enabled`, which by default is `true`. When you set this advanced property to `false`, then you can enable caching independently for configuration data and for user data.

#### Procedure 8.7. To Turn Off Global User Data Caching

**Disabling caching can have a severe negative impact on performance. This is because when caching is disabled, OpenAM must query a data store each time it needs data.**

If, however, you have at least one user data store that does not support LDAP persistent search, such as a relational database or an LDAP directory server that does not support persistent search, then you must disable the *global* cache for user data. Otherwise user data caches cannot stay in sync with changes to user data entries:

1. In the AM console, navigate to Deployment > Servers > *Server Name* > Advanced.
2. Set the value of the `com.iplanet.am.sdk.caching.enabled` property to `false` to disable caching overall.
3. Set the value of the `com.sun.identity.sm.cache.enabled` property to `true` to enable configuration data caching.

All supported configuration data stores support LDAP persistent search, so it is safe to enable configuration data caching.

You must explicitly set this property to `true`, because setting the value of the property `com.iplanet.am.sdk.caching.enabled` to `false` in the previous step disables both user and configuration data caching.

4. Save your work.
5. OpenAM starts persistent searches on user data stores when possible<sup>2</sup> in order to monitor changes. With user data store caching disabled, OpenAM still starts the persistent searches, even though it no longer uses the results.

Therefore, if you disable user data store caching, you should also disable persistent searches on user data stores in your deployment to improve performance. To disable persistent search on a user data store, remove the value of the Persistent Search Base DN configuration property and leave it blank. Locate this property under Realms > *Realm Name* > Data Stores > *Data Store Name* > Persistent Search Controls.

### Procedure 8.8. To Change the Maximum Size of Global User Data Cache

With a large user data store and active user base, the number of user entries in cache can grow large.

1. In the AM console, navigate to Configuration > Server Defaults > SDK.
2. Change the value of SDK Caching Maximum Size.

There is no corresponding setting for configuration data, as the number of configuration entries in a large deployment is not likely to grow nearly as large as the number of user entries.

### 8.4.3.2. Caching Properties For Java EE Policy Agents and SDK Clients

Policy agents and other OpenAM SDK clients can also cache user data, using most of the same properties as OpenAM server as described in Table 8.12, "Cache Properties". Clients, however, can receive updates by notification from OpenAM or, if notification fails, by polling OpenAM for changes.

### Procedure 8.9. To Enable Notification and Polling For Client Cache Updates

This procedure describes how to enable change notification and polling for policy agent user data cache updates. When configuring a custom OpenAM SDK client using a `.properties` file, use the same properties as for the policy agent configuration:

1. In the AM console, navigate to Realms > *Realm Name* > Applications > Agents > *Agent Type* > *Agent Name* to view and edit the policy agent profile.

<sup>2</sup> OpenAM starts persistent searches on user data stores on directory servers that support the `pssearch` control.



- On the Global tab page, check that the Agent Notification URL is set.

When notification is enabled, the agent registers a notification listener with OpenAM for this URL.

The corresponding property is `com.sun.identity.client.notification.url`.

- For any changes you make, Save your work.

You must restart the policy agent for the changes to take effect.

### 8.4.3.3. Cache Settings

The table below provides a quick reference, primarily for user data cache settings.

Notice that many properties for configuration data cache have `sm` (for Service Management) in their names, whereas those for user data have `idm` (for Identity Management) in their names:

*Table 8.12. Cache Properties*

Property	Description	Default	Applies To
<code>com.ipplanet.am.sdk.cache.maxSize</code>	Maximum number of user entries cached.	10000	Server and SDK
<code>com.ipplanet.am.sdk.caching.enabled</code>	<p>Whether to enable caching for both configuration data and also for user data.</p> <p>If <code>true</code>, this setting overrides <code>com.sun.identity.idm.cache.enabled</code> and <code>com.sun.identity.sm.cache.enabled</code>.</p> <p>If <code>false</code>, you can enable caching independently for configuration data and for user data using the aforementioned properties.</p>	<code>true</code>	Server & SDK
<code>com.ipplanet.am.sdk.remote.pollingTime</code>	<p>How often in minutes the SDK client, such as a policy agent should poll OpenAM for modified user data entries.</p> <p>The SDK also uses this value to determine the age of the oldest changes requested. The oldest changes requested are 2 minutes older than this setting. In other words, by default the SDK polls for entries changed in the last 3 minutes.</p> <p>Set this to 0 or a negative integer to disable polling.</p>	1 (minute)	SDK

Property	Description	Default	Applies To
<code>com.sun.am.event.notification.expire.time</code>	How long OpenAM stores a given change to a cached entry, so that clients polling for changes do not miss the change.	30 (minutes)	Server only
<code>com.sun.identity.idm.cache.enabled</code>	If <code>com.iplanet.am.sdk.caching.enabled</code> is <code>true</code> , this property is ignored.  Otherwise, set this to <code>true</code> to enable caching of user data.	<code>false</code>	Server & SDK
<code>com.sun.identity.idm.cache.entry.default.expire.time</code>	How many minutes to store a user data entry in the global user data cache.	30 (minutes)	Server & SDK
<code>com.sun.identity.idm.cache.entry.expire.enabled</code>	Whether user data entries in the global user data cache should expire over time.	<code>false</code>	Server & SDK
<code>com.sun.identity.idm.remote.notification.enabled</code>	Whether the SDK client, such as a policy agent should register a notification listener for user data changes with the OpenAM server.  The SDK client uses the URL specified by <code>com.sun.identity.client.notification.url</code> to register the listener so that OpenAM knows where to send notifications.  If notifications cannot be enabled for some reason, then the SDK client falls back to polling for changes.	<code>true</code>	SDK
<code>com.sun.identity.sm.cache.enabled</code>	If <code>com.iplanet.am.sdk.caching.enabled</code> is <code>true</code> , this property is ignored.  Otherwise, set this to <code>true</code> to enable caching of configuration data. It is recommended that you always set this to <code>true</code> .	<code>false</code>	Server & SDK
<code>sun-idrepo-ldapv3-dncache-enabled</code>	Set this to <code>true</code> to enable DN caching of user data.	<code>false</code>	Server & SDK
<code>sun-idrepo-ldapv3-dncache-size</code>	Sets the cache size.	1500	Server & SDK

## 8.5. Managing Sessions

The AM console lets the administrator view and manage active stateful user sessions by realm by navigating to Realms > *Realm Name* > Sessions.

Figure 8.4. Sessions Page

Sessions

Find sessions by entering a user ID.

amAdmin

x Invalidate Selected

AMADMIN		
IDLE TIME	IDLE TIME REMAINING	SESSION TIME REMAINING
<input type="checkbox"/> a few seconds	30 minutes	an hour

Your session

To search for active sessions, enter a username in the search box. OpenAM retrieves the sessions for the user and displays them within a table. If no active stateful session is found, OpenAM displays a session not found message.

You can end any sessions—except the current `amAdmin` user's session—by selecting it and clicking the Invalidate Selected button. As a result, the user has to authenticate again.

#### Note

Deleting a user does not automatically remove any of the user's stateful sessions. After deleting a user, check for any sessions for the user and remove them on the Sessions page.

## 8.6. Changing Host Names

When you change the OpenAM host name, you must make manual changes to the configuration. This chapter describes what to do. If you must also move an embedded configuration directory from one host to another, see the OpenDJ *Administration Guide* chapter, *Moving Servers*.

Changing OpenAM host names involves the following high-level steps.

- Adding the new host name to the Realm/DNS Aliases list.
- Exporting, editing, then importing the configuration.

This step relies on the **ssoadm** command, which you install separately from OpenAM as described in Section 2.3.1, "Setting up Administration Tools" in the *Installation Guide*.

- Stopping OpenAM and editing configuration files.
- Removing the old host name from the Realm/DNS Aliases list.

Before you start, make sure you have a current backup of your current installation. See Section 8.1, "Backing Up and Restoring Configurations" for instructions.

### *Procedure 8.10. To Add the New Host Name As an Alias*

1. Log in to the AM console as administrator, **amadmin**.
2. Under Realms > *Realm Name*, click Properties, add the new host name to the Realm/DNS Aliases list, and then save your work.

### *Procedure 8.11. To Export, Edit, and Import the Service Configuration*

1. Export the service configuration:

```
$ ssoadm \
  export-svc-cfg \
  --adminid amadmin \
  --encryptsecret myEncryptSecretString1234 \
  --password-file /tmp/pwd.txt \
  --outfile config.xml
```

Service Configuration was exported.

OpenAM uses the value entered in `--encryptsecret` to encrypt passwords stored in the backup file. It can be any value, and is required when restoring a configuration.

2. Edit the service configuration file:
  - Change the fully qualified domain name, such as **openam.example.com**, throughout the file.
  - If you are changing the context path, such as **/openam**, then make the following changes:
    - Change the value of **com.ipplanet.am.services.deploymentDescriptor**.
    - Change `contextPath` in the value of the **propertiesViewBeanURL="contextPath/auth/ACServiceInstanceList"**.
    - Change `contextPath` in the value of **propertiesViewBeanURL="contextPath/auth/ACModuleList"**.
    - Change the context path in a `<Value>` element that is a child of an `<AttributeValuePair>` element.

- Change the context path where it occurs throughout the file in the full URL to OpenAM, such as `http:&#47;&#47;openam.example.com:8080&#47;contextPath`.
- If you are changing the port number, then change the value of `com.ipplanet.am.server.port`.

Also change the port number in `host:port` combinations throughout the file.

- If you are changing the domain name, then change the cookie domain, such as `<Value>.example.com</Value>` throughout the file.

### 3. Import the updated service configuration:

```
$ ssoadm \
  import-svc-cfg \
  --adminid amadmin \
  --encryptsecret myEncryptSecretString1234 \
  --password-file /tmp/pwd.txt \
  --xmlfile config.xml
```

Directory Service contains existing data. Do you want to delete it? [y|N] **y**  
Please wait while we import the service configuration...  
Service Configuration was imported.

## Procedure 8.12. To Edit Configuration Files For the New Host Name

1. Stop OpenAM or the web container where it runs.
2. Edit the boot properties file, such as `/home/user/openam/boot.json`, changing the fully-qualified domain name (FQDN), port, and context path for OpenAM as necessary.
3. If you are changing the context path, then move the folder containing OpenAM configuration, such as `/home/user/openam/`, to match the new context path, such as `/home/user/openam2/`.
4. If you are changing the location or context path, change the name of the file in the `/home/user/.openamcfg` folder, such as `AMConfig_path_to_tomcat_webapps_openam_`, to match the new location and context path.

Also edit the path name in the file to match the change you made when moving the folder.

5. Restart OpenAM or the web container where it runs.

## Procedure 8.13. To Remove the Old Host Name As an Alias

1. Log in to the AM console as administrator, `amadmin`.
2. Under Realms > *Realm Name*, click Properties, remove the old host name from the Realm/DNS Aliases list, and then save your work.

## Chapter 9

# Troubleshooting

This chapter covers how to get debugging information and troubleshoot issues in deployments.

## 9.1. Is the Instance Running?

You can check over HTTP whether OpenAM is up, using `isAlive.jsp`. Point your application to the file under the deployment URL, such as `http://openam.example.com:8080/openam/isAlive.jsp`.

If you get a success code (with `Server is ALIVE:` in the body of the page returned), then the instance is in operation.

## 9.2. Debug Logging

OpenAM services capture a variety of information in debug logs. Unlike audit log records, debug log records are unstructured. Debug logs contain a variety of types of information that is useful when troubleshooting OpenAM, including stack traces. The level of debug log record output is configurable. Debug log records are always written to flat files.

### 9.2.1. Setting Debug Logging Levels

To adjust the debug level while OpenAM is running, login to the OpenAM console as OpenAM administrator and navigate to Deployment > Servers > *Server Name* > Debugging. The default level for debug logging is Error. This level is appropriate for normal production operations, in which case no debug log messages are expected.

Setting the debug log level to Warning increases the volume of messages. Setting the debug log level to Message dumps detailed trace messages. Unless told to do so by qualified support personnel, do not use Warning and Message levels in production.

By default, certain components that run in OpenAM's JVM—for example, embedded OpenDJ configuration stores—do not generate trace-level messages when you configure the debug log level to Message. If you need trace-level messages for these components, navigate to Deployment > Servers > *Server Name* > Advanced, create a `org.forgerock.openam.slf4j.enableTraceInMessage` property, and set its value to `true`.

### 9.2.2. Debug Logging to a Single File

During development, you might find it useful to log all debug messages to a single file. In order to do so, set Merge Debug Files to `on`.

OpenAM logs to a single file immediately after you change this property. You do not need to restart OpenAM or the container in which it runs for the change to take effect.

### 9.2.3. Debug Logging By Service

OpenAM lets you capture debug log messages selectively for a specific service. This can be useful when you must turn on debugging in a production system where you want to avoid excessive logging, but must gather messages when you reproduce a problem.

Perform these steps to capture debug messages for a specific service:

1. Log in to the AM console as administrator, `amadmin`.
2. Browse to `Debug.jsp`, for example `http://openam.example.com:8080/openam/Debug.jsp`.

No links to this page are provided in the AM console.

3. Select the service to debug and also the level required given the hints provided in the `Debug.jsp` page.

The changes takes effect immediately.

4. Promptly reproduce the problem you are investigating.
5. After reproducing the problem, immediately return to the `Debug.jsp` page, and revert to normal log levels to avoid filling up the disk where debug logs are stored.

### 9.2.4. Rotating Debug Logs

By default OpenAM does not rotate debug logs. To rotate debug logs, edit `WEB-INF/classes/debugconfig.properties` where OpenAM is deployed.

The `debugconfig.properties` file includes the following properties:

**`org.forgerock.openam.debug.prefix`**

Specifies the debug log file prefix applied when OpenAM rotates a debug log file. The property has no default. It takes a string as the property value.

**`org.forgerock.openam.debug.suffix`**

Specifies the debug log file suffix applied when OpenAM rotates a debug log file. The property takes a `SimpleDateFormat` string. The default is `-MM.dd.yyyy-kk.mm`.

**org.forgerock.openam.debug.rotation**

Specifies an interval in minutes between debug log rotations. Set this to a value greater than zero to enable debug log rotation based on time passed.

**org.forgerock.openam.debug.rotation.maxsize**

Specifies a maximum log file size in megabytes between debug log rotations. Set this to a value greater than zero to enable debug log rotation based on log file size.

Changes to properties in the `debugconfig.properties` file take effect immediately. You do not need to restart OpenAM or the container in which it runs for the changes to take effect.

## 9.3. Recording Troubleshooting Information

The OpenAM recording facility lets you initiate events to monitor OpenAM while saving output that is useful when performing troubleshooting.

OpenAM recording events save four types of information:

- OpenAM debug logs
- Thread dumps, which show you the status of every active thread, with output similar to a JStack stack trace
- Important run-time properties
- The OpenAM configuration

You initiate a recording event by invoking the **ssoadm start-recording** command or by using the **start** action of the `/json/records` REST API endpoint. Both methods use JSON to control the recording event.

This section describes starting and stopping recording using the **ssoadm** command, using a JSON file to configure the recording event, and locating the output recorded information. For information about using the `/json/records` REST API endpoint to activate and deactivate recording, see [Section 9.3.4](#), "RESTful Troubleshooting Information Recording". For general information about the REST API, see [Appendix A](#), "About the REST API".

### 9.3.1. Starting and Stopping Recording

Start OpenAM recording with the **ssoadm start-recording** command. For example:

```
$ ssoadm \
start-recording \
--servername http://openam.example.com:8080/openam \
--adminid amadmin \
--password-file /tmp/pwd.txt \
--jsonfile recording.json
```



```
{
  "recording":true,
  "record": {
    "issueID":103572,
    "referenceID":"policyEvalFails",
    "description":"Record everything",
    "zipEnable":false,
    "threadDump": {
      "enable":true,
      "delay": {
        "timeUnit":"SECONDS",
        "value":5
      }
    },
    "configExport": {
      "enable":true,
      "password":"admin password",
      "sharePassword":true
    },
    "debugLogs": {
      "debugLevel":"message",
      "autoStop": {
        "time": {
          "timeUnit":"MILLISECONDS",
          "value":15000
        },
        "fileSize": {
          "sizeUnit":"KB",
          "value":1048576
        }
      }
    },
    "status":"RUNNING",
    "folder":"/home/openam/debug/record/103572/policyEvalFails/"
  }
}
```

#### Note

The **ssoadm** command output in the preceding example is shown in indented format for ease of reading. The actual output is *not* indented.

In the preceding **ssoadm start-recording** command example, the `recording.json` file specifies the information to be recorded and under what conditions recording automatically terminates. This file is known as the *recording control file*. Section 9.3.2, "The Recording Control File" describes the format of recording control files and provides an annotated example.

An active recording event stops when:

- You explicitly tell OpenAM to stop recording by executing the **ssoadm stop-recording** command. See the **ssoadm(1)** in the *Reference* for details about this command.
- Another **ssoadm start-recording** command is sent to OpenAM that specifies an issue ID other that differs from the active recording event's issue ID. In this case, the initial recording session

terminates and the new recording event starts. Note that you can determine whether an OpenAM recording event is active by using the **ssoadm get-recording-status** command.

- A timer configured in the recording control file determines that the maximum amount of time for the recording event has been reached.
- A file size monitor configured in the recording control file determines that the maximum amount of information in debug logs has been reached.

### 9.3.2. The Recording Control File

A JSON file that is input to the **ssoadm start-recording** command controls the amount of information OpenAM records, the recording duration, and the location of recording output files.

For more information on the properties that comprise the recording control file, see the reference Section 10.3, "Record Control File Configuration Properties".

The following is an example of a recording control file:

```
{
  "issueID": 103572,
  "referenceID": "policyEvalFails",
  "description": "Troubleshooting artifacts in support of case 103572",
  "zipEnable": true,
  "configExport": {
    "enable": true,
    "password": "5x2RR70",
    "sharePassword": false
  },
  "debugLogs": {
    "debugLevel": "MESSAGE",
    "autoStop": {
      "time": {
        "timeUnit": "SECONDS",
        "value": 15
      },
      "fileSize": {
        "sizeUnit": "GB",
        "value": 1
      }
    }
  },
  "threadDump": {
    "enable": true,
    "delay": {
      "timeUnit": "SECONDS",
      "value": 5
    }
  }
}
```

The recording control file properties in the preceding example affect the recording output as follows:

Table 9.1. Recording Control File Example Properties and Their Effect on Recording Behavior

Recording Control File Property	Value	Effect
issueID, referenceID	103572, policyEvalFails	Recording output is stored at the path <code>debugFileLocation/record/103572/policyEvalFails_timestamp.zip</code> . For more information about the location of recording output, see Section 9.3.3, "Retrieving Recording Information".
Description	Troubleshooting artifacts in support of case 103572	No effect.
zipEnable	true	Recording output is compressed into a zip file.
configExport / enable	true	The OpenAM configuration is exported at the start of the recording event.
configExport / password	5x2RR70	Knowledge of this password will be required to access the OpenAM configuration that was saved during recording.
configExport / sharePassword	false	The password is not displayed in output messages displayed during the recording event or in the <code>info.json</code> file.
debugLogs / debugLevel	MESSAGE	Recording enables message-level debug logs during the recording event.
debugLogs / autoStop / time	SECONDS, 15	Because both the <code>time</code> and <code>fileSize</code> properties are set, recording stops after 15 seconds, or after the size of the debug logs exceeds 1 GB, whichever occurs first.
debugLogs / autoStop / fileSize	GB, 1	Because both the <code>time</code> and <code>fileSize</code> properties are set, recording stops after 15 seconds, or after the size of the debug logs exceeds 1 GB, whichever occurs first.
threadDump / enable	true	Thread dumps are taken throughout the recording event.
threadDump / delay	SECONDS, 5	The first thread dump is taken when the recording event starts. Additional thread dumps are taken every five seconds hence.

### 9.3.3. Retrieving Recording Information

Information recorded by OpenAM is stored at the path `debugFileLocation/record/issueID/referenceID`. For example, if the debug file location is `/home/openam/debug`, the issue ID `103572`, and the reference ID `policyEvalFails`, the path containing recorded information is `/home/openam/debug/record/103572/policyEvalFails`.

When there are multiple recording events with the same `issueID` and `referenceID`, OpenAM appends a timestamp to the `referenceID` of the earliest paths. For example, multiple recording events for issue ID `103572` and reference ID `policyEvalFails` might be stored at the following paths:

- Most recent recording: `debugFileLocation/record/103572/policyEvalFails`
- Next most recent recording: `debugFileLocation/record/103572/policyEvalFails_2015-10-24-11-48-51-902-PDT`
- Earliest recording: `debugFileLocation/record/103572/policyEvalFails_2015-08-10-15-15-10-140-PDT`

OpenAM compresses the output from recording events when you set the `zipEnable` property to `true`. The output file can be found at the path `debugFileLocation/record/issueID/referenceID_timestamp.zip`. For example, compressed output for a recording event for issue ID `103572` and reference ID `policyEvalFails` might be stored at the following path: `debugFileLocation/record/103572/policyEvalFails_2015-08-12-12-19-02-683-PDT.zip`.

Use the `referenceID` property value to segregate output from multiple problem recreations associated with the same case. For example, while troubleshooting case `103572`, you notice that you only have a problem when evaluating policy for members of the Finance realm. You could trigger two recording events as follows:

*Table 9.2. Segregating Recording Output Using the referenceID Value*

OpenAM Behavior	referenceIDValue	Recording Output Path
Policy evaluation behaves as expected for members of the Engineering realm.	<code>policyEvalSucceeds</code>	<code>debugFileLocation/record/103572/policyEvalSucceeds</code>
Policy evaluation unexpectedly fails for members of the Finance realm.	<code>policyEvalFails</code>	<code>debugFileLocation/record/103572/policyEvalFails</code>

### 9.3.4. RESTful Troubleshooting Information Recording

This section shows you how to start, stop, and get the status of a troubleshooting recording event using the REST API.

OpenAM provides the `/json/records` REST endpoint for the following:

- **Starting a recording event.** See Section 9.3.4.1, "Starting a Recording Event".
- **Getting the status of a recording event.** See Section 9.3.4.2, "Getting the Status of a Recording Event".
- **Stopping a recording event.** See Section 9.3.4.3, "Stopping a Recording Event".

You must authenticate to OpenAM as an administrative user to obtain an SSO token prior to calling the `/json/records` REST endpoint. You then pass the SSO token in the `iPlanetDirectoryPro` header as proof of authentication.

You can also record troubleshooting information by using the **ssoadm** command. For more information, see Section 9.3, "Recording Troubleshooting Information".

#### Note

The **curl** command output in the examples in this section is indented for ease of reading. The actual output is *not* indented, and the actions available from the **/json/records** endpoint do not support the **\_prettyPrint** parameter.

### 9.3.4.1. Starting a Recording Event

To start a recording event, perform an HTTP POST using the **/json/records** endpoint, specifying the **\_action=start** parameter in the URL. Specify a JSON payload identical in format to the input file for the **ssoadm start-recording** command, as described in Section 9.3.2, "The Recording Control File".

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "iPlanetDirectoryPro: AQIC5..." \
--data '{
  "issueID": 103572,
  "referenceID": "policyEvalFails",
  "description": "Troubleshooting artifacts in support of case 103572",
  "zipEnable": true,
  "configExport": {
    "enable": true,
    "password": "5x2RR70",
    "sharePassword": false
  },
  "debugLogs": {
    "debugLevel": "MESSAGE",
    "autoStop": {
      "time": {
        "timeUnit": "SECONDS",
        "value": 15
      },
    },
    "fileSize": {
      "sizeUnit": "GB",
      "value": 1
    }
  },
  "threadDump": {
    "enable": true,
    "delay": {
      "timeUnit": "SECONDS",
      "value": 5
    }
  }
}' \
https://openam.example.com:8443/openam/json/records?_action=start
{
  "recording":true,
  "record":{
```

```
{
  "issueID":103572,
  "referenceID":"policyEvalFails",
  "description":"Troubleshooting artifacts in support of case 103572",
  "zipEnable":true,
  "threadDump":{
    "enable":true,
    "delay":{
      "timeUnit":"SECONDS",
      "value":5
    }
  },
  "configExport":{
    "enable":true,
    "password":"xxxxxx",
    "sharePassword":false
  },
  "debugLogs":{
    "debugLevel":"message",
    "autoStop":{
      "time":{
        "timeUnit":"MILLISECONDS",
        "value":15000
      },
      "fileSize":{
        "sizeUnit":"KB",
        "value":1048576
      }
    }
  },
  "status":"RUNNING",
  "folder":"/opt/demo/openam/config/openam/debug/record/103572/policyEvalFails/"
}
```

### 9.3.4.2. Getting the Status of a Recording Event

To get the status of a recording event, perform an HTTP POST using the `/json/records` endpoint, specifying the `_action=status` parameter in the URL:

```
$ curl \
--request POST \
--header "iPlanetDirectoryPro: AQIC5..." \
https://openam.example.com:8443/openam/json/records?_action=status
```

If there is no active recording event, the following output appears:

```
{
  "recording":false
}
```

If there is an active recording event, output similar to the following appears:

```
{
  "recording":true,
  "record":{
    "issueID":103572,
```

```
{
  "referenceID": "policyEvalFails",
  "description": "Troubleshooting artifacts in support of case 103572",
  "zipEnable": true,
  "threadDump": {
    "enable": true,
    "delay": {
      "timeUnit": "SECONDS",
      "value": 5
    }
  },
  "configExport": {
    "enable": true,
    "password": "xxxxxx",
    "sharePassword": false
  },
  "debugLogs": {
    "debugLevel": "message",
    "autoStop": {
      "time": {
        "timeUnit": "MILLISECONDS",
        "value": 15000
      }
    },
    "fileSize": {
      "sizeUnit": "KB",
      "value": 1048576
    }
  }
},
{
  "status": "RUNNING",
  "folder": "/opt/demo/openam/config/openam/debug/record/103572/policyEvalFails/"
}
}
```

### 9.3.4.3. Stopping a Recording Event

To stop a recording event, perform an HTTP POST using the [/json/records](#) endpoint, specifying the [\\_action=stop](#) parameter in the URL:

```
$ curl \
--request POST \
\
--header "iPlanetDirectoryPro: AQIC5..." \
https://openam.example.com:8443/openam/json/records?_action=stop
```

If there is no active recording event, OpenAM returns a 400 error code.

If there is an active recording event, output similar to the following appears:

```
{
  "recording": false,
  "record": {
    "issueID": 103572,
    "referenceID": "policyEvalFails",
    "description": "Troubleshooting artifacts in support of case 103572",
    "zipEnable": true,
    "threadDump": {
      "enable": true,
```

```
"delay":{
  "timeUnit":"SECONDS",
  "value":5
},
"configExport":{
  "enable":true,
  "password":"xxxxxx",
  "sharePassword":false
},
"debugLogs":{
  "debugLevel":"message",
  "autoStop":{
    "time":{
      "timeUnit":"MILLISECONDS",
      "value":15000
    },
    "fileSize":{
      "sizeUnit":"KB",
      "value":1048576
    }
  }
},
"status":"STOPPED",
"folder":"/opt/demo/openam/config/openam/debug/record/103572/policyEvalFails/"
}
```



## Chapter 10

# Reference

This chapter covers OpenAM configuration properties accessible through the AM console, most of which can also be set by using the **ssoadm** command.

For reference information that is not contained on this guide, see the OpenAM Reference Guide or the Reference section included in each of the topic-oriented OpenAM guides.

## 10.1. Data Store Configuration Properties

Use the following reference to configure different data store types navigating to Realms > *Realm Name* > Data Stores.

### 10.1.1. Active Directory Configuration Properties

Use these attributes when configuring Active Directory data stores:

**ssoadm** service name: `sunIdentityRepositoryService`

#### Name

Name for the data store configuration

#### Load schema when finished

Add appropriate LDAP schema to the directory server when saving the configuration. The LDAP Bind DN user must have access to perform this operation.

This attribute is not available for use with the **ssoadm** command.

Default: false

#### LDAP Server

`host:port` to contact the directory server, with optional `|server_ID|site_ID` for deployments with multiple servers and sites.

OpenAM uses the optional settings to determine which directory server to contact first. OpenAM tries to contact directory servers in the following priority order, with highest priority first:

1. The first directory server in the list whose `server_ID` matches the current OpenAM server.

2. The first directory server in the list whose *site\_ID* matches the current OpenAM server.
3. The first directory server in the remaining list.

If the directory server is not available, OpenAM proceeds to the next directory server in the list.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-ldap-server`

Default: `host:port` of the initial directory server configured for this OpenAM server.

### LDAP Bind DN

Bind DN for connecting to the directory server. Some OpenAM capabilities require write access to directory entries.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-authid`

Default: `CN=Administrator,CN=Users,base-dn`

### LDAP Bind Password

Bind password for connecting to the directory server

**ssoadm** attribute: `sun-idrepo-ldapv3-config-authpw`

### LDAP Organization DN

The base DN under which to find user and group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-organization_name`

Default: `base-dn`

### LDAP Connection Mode

Whether to use LDAP, LDAPS or StartTLS to connect to the directory server. When LDAPS or StartTLS are enabled, OpenAM must be able to trust server certificates, either because the server certificates were signed by a CA whose certificate is already included in the trust store used by the container where OpenAM runs, or because you imported the certificates into the trust store.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-connection-mode`

Possible values: `LDAP`, `LDAPS`, and `StartTLS`

### LDAP Connection Pool Maximum Size

Maximum number of connections to the directory server. Make sure the directory service can cope with the maximum number of client connections across all servers.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-connection_pool_max_size`

Default: 10

## LDAP Connection Heartbeat Interval

How often to send a heartbeat request to the directory server to ensure that the connection does not remain idle. Some network administrators configure firewalls and load balancers to drop connections that are idle for too long. You can turn this off by setting the value to 0 or to a negative number. To set the units for the interval use LDAP Connection Heartbeat Time Unit.

**ssoadm** attribute: `openam-idrepo-ldapv3-heartbeat-interval`

Default: 10

## LDAP Connection Heartbeat Time Unit

Time unit for the LDAP Connection Heartbeat Interval setting.

**ssoadm** attribute: `openam-idrepo-ldapv3-heartbeat-timeunit`

Default: `second`

## Maximum Results Returned from Search

A cap for the number of search results to request. For example, when using the Subjects tab to view profiles, even if you set Configuration > Console > Administration > Maximum Results Returned from Search to a larger number, OpenAM does not exceed this setting. Rather than raise this number, consider narrowing your search to match fewer directory entries.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-max-result`

Default: 1000

## Search Timeout

Maximum time to wait for search results in seconds. Does not apply to persistent searches.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-time-limit`

Default: 10

## LDAPv3 Plugin Search Scope

LDAP searches can apply to a single entry (SCOPE\_BASE), entries directly below the search DN (SCOPE\_ONE), or all entries below the search DN (SEARCH\_SUB)

**ssoadm** attribute: `sun-idrepo-ldapv3-config-search-scope`

Default: `SCOPE_SUB`

## LDAPv3 Repository Plugin Class Name

OpenAM identity repository implementation.

**ssoadm** attribute: `sunIdRepoClass`

Default: `org.forgerock.openam.idrepo.ldap.DJLDAPv3Repo`

## Attribute Name Mapping

Map of OpenAM profile attribute names to directory server attribute names.

**ssoadm** attribute: `sunIdRepoAttributeMapping`

Default: `userPassword=unicodePwd`

## LDAPv3 Plugin Supported Types and Operations

Map of OpenAM operations that can be performed in the specified OpenAM contexts.

**ssoadm** attribute: `sunIdRepoSupportedOperations`

Default: `group=read,create,edit,delete, realm=read,create,edit,delete,service, user=read,create,edit,delete`

## LDAP Users Search Attribute

When searching for a user by name, match values against this attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-users-search-attribute`

Default: `cn`

## LDAP Users Search Filter

When searching for users, apply this LDAP search filter as well.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-users-search-filter`

Default: `(objectclass=person)`

## LDAP People Container Naming Attribute

RDN attribute of the LDAP base DN which contains user profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-people-container-name`

Default: `cn`

## LDAP People Container Value

RDN attribute value of the LDAP base DN which contains user profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-people-container-value`

Default: `users`

## LDAP User Object Class

User profiles have these LDAP object classes.

OpenAM handles only those attributes listed in this setting. OpenAM discards any such unlisted attributes from requests and the request proceeds without the attribute.

For example, with default settings, if you request that OpenAM execute a search that asks for the `mailAlternateAddress` attribute, OpenAM does the search, but does not request `mailAlternateAddress`. In the same way, OpenAM does perform an update operation with a request to set the value of an unlisted attribute like `mailAlternateAddress`, but it drops the unlisted attribute from the update request.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-user-objectclass`

Default: `organizationalPerson, person, top, User,`

## LDAP User Attributes

User profiles have these LDAP attributes.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-user-attributes`

Default: `assignedDashboard, cn, createTimeStamp, devicePrintProfiles, displayName, distinguishedName, dn, employeeNumber, givenName, iplanet-am-auth-configuration, iplanet-am-session-destroy-sessions, iplanet-am-session-get-valid-sessions, iplanet-am-session-max-caching-time, iplanet-am-session-max-idle-time, iplanet-am-session-max-session-time, iplanet-am-session-quota-limit, iplanet-am-session-service-status, iplanet-am-user-account-life, iplanet-am-user-admin-start-dn, iplanet-am-user-alias-list, iplanet-am-user-auth-config, iplanet-am-user-auth-modules, iplanet-am-user-failure-url, iplanet-am-user-federation-info, iplanet-am-user-federation-info-key, iplanet-am-user-login-status, iplanet-am-user-password-reset-force-reset, iplanet-am-user-password-reset-options, iplanet-am-user-password-reset-question-answer, iplanet-am-user-success-url, kbaActiveIndex, kbaInfo, mail, modifyTimestamp, name, oath2faEnabled, oathDeviceProfiles, objectGUID, objectclass, postalAddress, preferredLocale, preferredlanguage, preferredtimezone, pushDeviceProfiles, sAMAccountName, sn, sun-fm-saml2-nameid-info, sun-fm-saml2-nameid-infokey, sunAMAuthInvalidAttemptsData, sunIdentityMSISDNNumber, sunIdentityServerDiscoEntries, sunIdentityServerPPAddressCard, sunIdentityServerPPCommonNameAltCN, sunIdentityServerPPCommonNameCN, sunIdentityServerPPCommonNameFN, sunIdentityServerPPCommonNameMN, sunIdentityServerPPCommonNamePT, sunIdentityServerPPCommonNameSN, sunIdentityServerPPDemographicsAge, sunIdentityServerPPDemographicsBirthDay, sunIdentityServerPPDemographicsDisplayLanguage, sunIdentityServerPPDemographicsLanguage, sunIdentityServerPPDemographicsTimeZone, sunIdentityServerPPEmergencyContact, sunIdentityServerPPEmploymentIdentityAlt0, sunIdentityServerPPEmploymentIdentityJobTitle, sunIdentityServerPPEmploymentIdentityOrg, sunIdentityServerPPEncryPTKey, sunIdentityServerPPFacadeGreetSound, sunIdentityServerPPFacadeMugShot, sunIdentityServerPPFacadeNamePronounced, sunIdentityServerPPFacadeWebSite, sunIdentityServerPPFacadeGreetmesound, sunIdentityServerPPInformalName, sunIdentityServerPPLegalIdentityAltIdType, sunIdentityServerPPLegalIdentityAltIdValue, sunIdentityServerPPLegalIdentityDOB, sunIdentityServerPPLegalIdentityGender, sunIdentityServerPPLegalIdentityLegalName, sunIdentityServerPPLegalIdentityMaritalStatus, sunIdentityServerPPLegalIdentityVATIdType, sunIdentityServerPPLegalIdentityVATIdValue, sunIdentityServerPPMsgContact, sunIdentityServerPPSignKey, telephoneNumber, unicodePwd, userAccountControl, userPrincipalname, userpassword`

## Create User Attribute Mapping

When creating a user profile, apply this map of OpenAM profile attribute names to directory server attribute names.

The LDAP user profile entries require the Common Name (**cn**) and Surname (**sn**) attributes, so that LDAP constraint violations do not occur when performing an add operation.

The **cn** attribute gets its value from the **uid** attribute, which comes from the User Name field on the AM console's login page. The **sn** attribute gets the value of the **givenName** attribute. Attributes not mapped to another attribute and attributes mapped to themselves (for example, **cn=cn**) take the value of the username unless the attribute values are provided when creating the profile.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-createuser-attr-mapping`

Default: `cn, sn`

### Attribute Name of User Status

Attribute to check/set user status.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-isactive`

Default: `userAccountControl`

### User Status Active Value

Active users have the user status attribute set to this value.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-active`

Default: 544

### User Status Inactive Value

Inactive users have the user status attribute set to this value.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-inactive`

Default: 546

### Authentication Naming Attribute

RDN attribute for building the bind DN when given a username and password to authenticate a user against the directory server.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-auth-naming-attr`

Default: `cn`

### LDAP Groups Search Attribute

When searching for a group by name, match values against this attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-groups-search-attribute`

Default: `cn`

### LDAP Groups Search Filter

When searching for groups, apply this LDAP search filter as well.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-groups-search-filter`

Default: `(objectclass=group)`

## LDAP Groups Container Naming Attribute

RDN attribute of the LDAP base DN which contains group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-container-name`

Default: `cn`

## LDAP Groups Container Value

RDN attribute value of the LDAP base DN which contains group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-container-value`

Default: `users`

## LDAP Groups Object Class

Group profiles have these LDAP object classes.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-objectclass`

Default: `Group, top`

## LDAP Groups Attributes

Group profiles have these LDAP attributes.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-attributes`

Default: `cn, distinguishedName, dn, member, name, objectCategory, objectclass, sAMAccountName, sAMAccountType`

## Attribute Name for Group Membership

LDAP attribute in the member's LDAP entry whose values are the groups to which a member belongs.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-memberof`

## Attribute Name of Unique Member

Attribute in the group's LDAP entry whose values are the members of the group.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-uniquemember`

Default: `member`

## Persistent Search Base DN

Base DN for LDAP-persistent searches used to receive notification of changes in directory server data.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-psearchbase`

Default: `base-dn`

### Persistent Search Scope

LDAP searches can apply to a single entry (SCOPE\_BASE), entries directly below the search DN (SCOPE\_ONE), or all entries below the search DN (SEARCH\_SUB).

Specify either `SCOPE_BASE` or `SCOPE_ONE`. Do not specify `SCOPE_SUB`, as it can have a severe impact on Active Directory performance.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-psearch-scope`

Default: `SCOPE_SUB`

### The Delay Time Between Retries

How long to wait after receiving an error result that indicates OpenAM should try the LDAP operation again.

**ssoadm** attribute: `com.ipplanet.am.ldap.connection.delay.between.retries`

Default: 1000 milliseconds

### DN Cache Enabled

Whether to enable the DN cache, which is used to cache DN lookups that can happen in bursts during authentication. As the cache can become stale when a user is moved or renamed, enable DN caching when the directory service allows move/rename operations (Mod DN), and when OpenAM uses persistent searches to obtain notification of such updates.

**ssoadm** attribute: `sun-idrepo-ldapv3-dncache-enabled`

Default: false

### DN Cache Size

Maximum number of DN's cached when caching is enabled.

**ssoadm** attribute: `sun-idrepo-ldapv3-dncache-size`

Default: 1500 items

## 10.1.2. Active Directory Application Mode Configuration Properties

Use these attributes when configuring Active Directory Application Mode (ADAM) Data Stores:

**ssoadm** service name: `sunIdentityRepositoryService`

### Name

Name for the data store configuration.



## Load schema when finished

Add appropriate LDAP schema to the directory server when saving the configuration. The LDAP Bind DN user must have access to perform this operation.

This attribute is not available for use with the **ssoadm** command.

Default: false

## LDAP Server

**host:port** to contact the directory server, with optional **|server\_ID|site\_ID** for deployments with multiple servers and sites.

OpenAM uses the optional settings to determine which directory server to contact first. OpenAM tries to contact directory servers in the following priority order, with highest priority first:

1. The first directory server in the list whose *server\_ID* matches the current OpenAM server.
2. The first directory server in the list whose *site\_ID* matches the current OpenAM server.
3. The first directory server in the remaining list.

If the directory server is not available, OpenAM proceeds to the next directory server in the list.

**ssoadm** attribute: **sun-idrepo-ldapv3-config-ldap-server**

Default: **host:port** of the initial directory server configured for this OpenAM server.

## LDAP Bind DN

Bind DN for connecting to the directory server. Some OpenAM capabilities require write access to directory entries.

**ssoadm** attribute: **sun-idrepo-ldapv3-config-authid**

Default: **CN=Administrator,CN=Users,base-dn**

## LDAP Bind Password

Bind password for connecting to the directory server.

**ssoadm** attribute: **sun-idrepo-ldapv3-config-authpw**

## LDAP Organization DN

The base DN under which to find user and group profiles.

**ssoadm** attribute: **sun-idrepo-ldapv3-config-organization\_name**

Default: **base-dn**

## LDAP Connection Mode

Whether to use LDAP, LDAPS or StartTLS to connect to the directory server. When LDAPS or StartTLS are enabled, OpenAM must be able to trust server certificates, either because the

server certificates were signed by a CA whose certificate is already included in the trust store used by the container where OpenAM runs, or because you imported the certificates into the trust store.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-connection-mode`

Possible values: `LDAP`, `LDAPS`, and `StartTLS`

### LDAP Connection Pool Maximum Size

Maximum number of connections to the directory server. Make sure the directory service can cope with the maximum number of client connections across all servers.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-connection_pool_max_size`

Default: 10

### LDAP Connection Heartbeat Interval

How often to send a heartbeat request to the directory server to ensure that the connection does not remain idle. Some network administrators configure firewalls and load balancers to drop connections that are idle for too long. You can turn this off by setting the value to 0 or to a negative number. To set the units for the interval, use LDAP Connection Heartbeat Time Unit.

**ssoadm** attribute: `openam-idrepo-ldapv3-heartbeat-interval`

Default: 10

### LDAP Connection Heartbeat Time Unit

Time unit for the LDAP Connection Heartbeat Interval setting

**ssoadm** attribute: `openam-idrepo-ldapv3-heartbeat-timeunit`

Default: `second`

### Maximum Results Returned from Search

A cap for the number of search results to request. For example, when using the Subjects tab to view profiles, even if you set Configuration > Console > Administration > Maximum Results Returned from Search to a larger number, OpenAM does not exceed this setting. Rather than raise this number, consider narrowing your search to match fewer directory entries.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-max-result`

Default: 1000

### Search Timeout

Maximum time to wait for search results in seconds. Does not apply to persistent searches.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-time-limit`

Default: 10

### LDAPv3 Plugin Search Scope

LDAP searches can apply to a single entry (SCOPE\_BASE), entries directly below the search DN (SCOPE\_ONE), or all entries below the search DN (SEARCH\_SUB).

**ssoadm** attribute: `sun-idrepo-ldapv3-config-search-scope`

Default: `SCOPE_SUB`

### LDAPv3 Repository Plugin Class Name

OpenAM identity repository implementation.

**ssoadm** attribute: `sunIdRepoClass`

Default: `org.forgerock.openam.idrepo.ldap.DJLDAPv3Repo`

### Attribute Name Mapping

Map of OpenAM profile attribute names to directory server attribute names.

**ssoadm** attribute: `sunIdRepoAttributeMapping`

Default: `userPassword=unicodePwd`

### LDAPv3 Plugin Supported Types and Operations

Map of OpenAM operations that can be performed in the specified OpenAM contexts.

**ssoadm** attribute: `sunIdRepoSupportedOperations`

Default: `group=read,create,edit,delete, realm=read,create,edit,delete,service, user=read,create,edit,delete,service`

### LDAP Users Search Attribute

When searching for a user by name, match values against this attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-users-search-attribute`

Default: `cn`

### LDAP Users Search Filter

When searching for users, apply this LDAP search filter as well.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-users-search-filter`

Default: `(objectclass=person)`

### LDAP People Container Naming Attribute

RDN attribute of the LDAP base DN which contains user profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-people-container-name`

## LDAP People Container Value

RDN attribute value of the LDAP base DN which contains user profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-people-container-value`

## LDAP User Object Class

User profiles have these LDAP object classes.

OpenAM handles only those attributes listed in this setting. OpenAM discards any unlisted attributes from requests and the request proceeds without the attribute.

For example, with default settings, if you request that OpenAM execute a search that asks for the `mailAlternateAddress` attribute, OpenAM does the search, but does not request `mailAlternateAddress`. In the same way, OpenAM does perform an update operation with a request to set the value of an unlisted attribute like `mailAlternateAddress`, but it drops the unlisted attribute from the update request.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-user-objectclass`

Default: `devicePrintProfilesContainer, forgerock-am-dashboard-service, iPlanetPreferences, iplanet-am-auth-configuration-service, iplanet-am-managed-person, iplanet-am-user-service, kbaInfoContainer, oathDeviceProfilesContainer, organizationalPerson, person, pushDeviceProfilesContainer, sunAMAuthAccountLockout, sunFMSAML2NameIdentifier, sunFederationManagerDataStore, sunIdentityServerLibertyPPService, top, User`

## LDAP User Attributes

User profiles have these LDAP attributes.

OpenAM handles only those attributes listed in this setting. OpenAM discards any unlisted attributes from requests and the request proceeds without the attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-user-attributes`

Default: `assignedDashboard, cn, createTimestamp, devicePrintProfiles, displayName, distinguishedName, dn, employeeNumber, givenName, iplanet-am-auth-configuration, iplanet-am-session-destroy-sessions, iplanet-am-session-get-valid-sessions, iplanet-am-session-max-caching-time, iplanet-am-session-max-idle-time, iplanet-am-session-max-session-time, iplanet-am-session-quota-limit, iplanet-am-session-service-status, iplanet-am-user-account-life, iplanet-am-user-admin-start-dn, iplanet-am-user-alias-list, iplanet-am-user-auth-config, iplanet-am-user-auth-modules, iplanet-am-user-failure-url, iplanet-am-user-federation-info, iplanet-am-user-federation-info-key, iplanet-am-user-login-status, iplanet-am-user-password-reset-force-reset, iplanet-am-user-password-reset-options, iplanet-am-user-password-reset-question-answer, iplanet-am-user-success-url, kbaActiveIndex, kbaInfo, mail, modifyTimestamp, msDS-UserAccountDisabled, name, oath2faEnabled, oathDeviceProfiles, objectGUID, objectclass, postalAddress, preferredLocale, preferredlanguage, preferredtimezone, pushDeviceProfiles, sn, sun-fm-saml2-nameid-info, sun-fm-saml2-nameid-infokey, sunAMAuthInvalidAttemptsData, sunIdentityMSISDNNumber, sunIdentityServerDiscoEntries, sunIdentityServerPPAddressCard, sunIdentityServerPPCommonNameAltCN,`

sunIdentityServerPPCommonNameCN, sunIdentityServerPPCommonNameFN, sunIdentityServerPPCommonNameMN, sunIdentityServerPPCommonNamePT, sunIdentityServerPPCommonNameSN, sunIdentityServerPPDemographicsAge, sunIdentityServerPPDemographicsBirthDay, sunIdentityServerPPDemographicsDisplayLanguage, sunIdentityServerPPDemographicsLanguage, sunIdentityServerPPDemographicsTimeZone, sunIdentityServerPPEmergencyContact, sunIdentityServerPPEmploymentIdentityAlt0, sunIdentityServerPPEmploymentIdentityJobTitle, sunIdentityServerPPEmploymentIdentityOrg, sunIdentityServerPPEncryPTKey, sunIdentityServerPPFacadeGreetSound, sunIdentityServerPPFacadeMugShot, sunIdentityServerPPFacadeNamePronounced, sunIdentityServerPPFacadeWebSite, sunIdentityServerPPFacadegreetmesound, sunIdentityServerPPInformalName, sunIdentityServerPPLegalIdentityAltIdType, sunIdentityServerPPLegalIdentityAltIdValue, sunIdentityServerPPLegalIdentityDOB, sunIdentityServerPPLegalIdentityGender, sunIdentityServerPPLegalIdentityLegalName, sunIdentityServerPPLegalIdentityMaritalStatus, sunIdentityServerPPLegalIdentityVATIdType, sunIdentityServerPPLegalIdentityVATIdValue, sunIdentityServerPPMsgContact, sunIdentityServerPPSignKey, telephoneNumber, unicodePwd, userPrincipalname, userpassword

## Create User Attribute Mapping

When creating a user profile, apply this map of OpenAM profile attribute names to directory server attribute names.

Attributes not mapped to another attribute (for example, `cn`) and attributes mapped to themselves, (for example, `cn=cn`) take the value of the username unless the attribute values are provided when creating the profile. The object classes for user profile LDAP entries generally require Common Name (`cn`) and Surname (`sn`) attributes, so this prevents an LDAP constraint violation when performing the add operation.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-createuser-attr-mapping`

Default: `cn, sn`

## Attribute Name of User Status

Attribute to check/set user status.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-isactive`

Default: `msDS-UserAccountDisabled`

## User Status Active Value

Active users have the user status attribute set to this value.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-active`

Default: `FALSE`

## User Status Inactive Value

Inactive users have the user status attribute set to this value.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-inactive`

Default: `TRUE`

### Authentication Naming Attribute

RDN attribute for building the bind DN when given a username and password to authenticate a user against the directory server.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-auth-naming-attr`

Default: `cn`

### LDAP Groups Search Attribute

When searching for a group by name, match values against this attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-groups-search-attribute`

Default: `cn`

### LDAP Groups Search Filter

When searching for groups, apply this LDAP search filter as well.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-groups-search-filter`

Default: `(objectclass=group)`

### LDAP Groups Container Naming Attribute

RDN attribute of the LDAP base DN which contains group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-container-name`

Default: `cn`

### LDAP Groups Container Value

RDN attribute value of the LDAP base DN which contains group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-container-value`

### LDAP Groups Object Class

Group profiles have these LDAP object classes.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-objectclass`

Default: `Group, top`

### LDAP Groups Attributes

Group profiles have these LDAP attributes.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-attributes`

Default: `cn, distinguishedName, dn, member, name, objectCategory, objectclass, sAMAccountName, sAMAccountType`

### Attribute Name for Group Membership

LDAP attribute in the member's LDAP entry whose values are the groups to which a member belongs.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-memberof`

### Attribute Name of Unique Member

Attribute in the group's LDAP entry whose values are the members of the group.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-uniquemember`

Default: `member`

### Persistent Search Base DN

Base DN for LDAP-persistent searches used to receive notification of changes in directory server data.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-psearchbase`

Default: `base-dn`

### Persistent Search Scope

LDAP searches can apply to a single entry (SCOPE\_BASE), entries directly below the search DN (SCOPE\_ONE), or all entries below the search DN (SEARCH\_SUB).

Specify either `SCOPE_BASE` or `SCOPE_ONE`. Do not specify `SCOPE_SUB`, as it can have a severe impact on Active Directory performance.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-psearch-scope`

Default: `SCOPE_SUB`

### The Delay Time Between Retries

How long to wait after receiving an error result that indicates OpenAM should try the LDAP operation again.

**ssoadm** attribute: `com.ipplanet.am.ldap.connection.delay.between.retries`

Default: 1000 milliseconds

### DN Cache Enabled

Whether to enable the DN cache, which is used to cache DN lookups that can happen in bursts during authentication. As the cache can become stale when a user is moved or renamed, enable DN caching when the directory service allows move/rename operations (Mod DN), and when OpenAM uses persistent searches to obtain notification of such updates.

**ssoadm** attribute: `sun-idrepo-ldapv3-dncache-enabled`

Default: false

### DN Cache Size

Maximum number of DN's cached when caching is enabled.

**ssoadm** attribute: `sun-idrepo-ldapv3-dncache-size`

Default: 1500 items

## 10.1.3. Database Repository (Early Access) Configuration Properties

Use these attributes when configuring database repository (early access) data stores:

### Important

This feature is in Early Access, meaning it is not generally supported for use in production environments. If you expect to use a relational database as an identity repository other than for development or testing purposes, first confirm supportability of your configuration with an expert. You can contact ForgeRock at [info@forgerock.com](mailto:info@forgerock.com).

**ssoadm** service name: `sunIdentityRepositoryService`

### Name

Name for the data store configuration.

### Load schema when finished

Add the appropriate schema to the database on saving the configuration.

This attribute is not available for use with the **ssoadm** command.

Default: false

### Database Data Access Object Plugin Class Name

OpenAM data access implementation.

**ssoadm** attribute: `sun-opensso-database-dao-class-name`

Default: `com.sun.identity.idm.plugins.database.JdbcSimpleUserDao`

### Connection Type

Whether to connect directly to the database, or to connect through JNDI provided by the container where OpenAM runs.

**ssoadm** attribute: `sun-opensso-database-dao-JDBCConnectionType`



Default: Connection is retrieved via programmatic connection

### Database DataSource Name

Data source name from the container configuration when connecting over JNDI.

**ssoadm** attribute: `sun-opensso-database-DataSourceJndiName`

Default: `java:comp/env/jdbc/openssouserdb`

### JDBC Driver Class Name

Driver class used when connecting directly.

**ssoadm** attribute: `sun-opensso-database-JDBCDriver`

Default: `com.mysql.jdbc.Driver`

### JDBC Driver URL

URL used when connecting directly.

**ssoadm** attribute: `sun-opensso-database-JDBCUrl`

Default: `jdbc:mysql://127.0.0.1:3306/test`

### Connect This User to Database

Username used when connecting directly.

**ssoadm** attribute: `sun-opensso-database-JDBCDBuser`

Default: `root`

### Password for Connecting to Database

Password used when connecting directly.

**ssoadm** attribute: `sun-opensso-database-JDBCDBpassword`

### Maximum Results Returned from Search

A cap for the number of search results to request. For example, when using the Subjects tab to view profiles, even if you set Configuration > Console > Administration > Maximum Results Returned from Search to a larger number, OpenAM does not exceed this setting. Rather than raise this number, consider narrowing your search to match fewer profiles.

**ssoadm** attribute: `sun-opensso-database-config-max-result`

Default: 1000

### Database Repository Plugin Class Name

OpenAM identity repository implementation.

**ssoadm** attribute: `sunIdRepoClass`

Default: `com.sun.identity.idm.plugins.database.DatabaseRepo`

## Attribute Name Mapping

Map of OpenAM profile attribute names to database column names.

**ssoadm** attribute: `sunIdRepoAttributeMapping`

Default: `iplanet-am-user-account-life=iplanet_am_user_account_life, iplanet-am-user-alias-list=iplanet_am_user_alias_list, iplanet-am-user-auth-config=iplanet_am_user_auth_config, iplanet-am-user-failure-url=iplanet_am_user_failure_url, iplanet-am-user-password-reset-force-reset=iplanet_am_user_password_reset_force_reset, iplanet-am-user-password-reset-question-answer=iplanet_am_user_password_reset_question_answer, iplanet-am-user-password-resetoptions=iplanet_am_user_password_resetoptions, iplanet-am-user-success-url=iplanet_am_user_success_url`

## Database Plugin Supported Types and Operations

Map of OpenAM operations that can be performed in the specified OpenAM contexts.

**ssoadm** attribute: `sun-opensso-database-sunIdRepoSupportedOperations`

Default: `group=read,create,edit,delete, user=read,create,edit,delete,service`

## Database User Table Name

Table to store user profiles.

### Tip

A MySQL database table for storing user profiles could be created with the following example SQL statement:

```
CREATE TABLE `opensso_users` (
  `_id` int(10) NOT NULL AUTO_INCREMENT,
  `uid` varchar(35) DEFAULT NULL,
  `sn` varchar(35) DEFAULT NULL,
  `cn` varchar(75) DEFAULT NULL,
  `userpassword` varchar(45) DEFAULT NULL,
  `inetuserstatus` varchar(8) DEFAULT "Active",
  `mail` varchar(254) DEFAULT NULL,
  `givenname` varchar(35) DEFAULT NULL,
  `telephonenumber` varchar(15) DEFAULT NULL,
  `employeenumber` varchar(10) DEFAULT NULL,
  `postaladdress` varchar(175) DEFAULT NULL,
  `iplanet-am-user-account-life` varchar(19) DEFAULT NULL,
  PRIMARY KEY (`_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

**ssoadm** attribute: `sun-opensso-database-UserTableName`

Default: `opensso_users`

## List of User Attributes Names in Database

Columns for user profile attributes.

**ssoadm** attribute: `sun-opensso-database-UserAttr`

Default: `ChangePassword, cn, employeeenumber, givenname, inetuserstatus, iplanet_am_user_account_life, iplanet_am_user_alias_list, iplanet_am_user_auth_config, iplanet_am_user_failure_url, iplanet_am_user_password_reset_force_reset, iplanet_am_user_password_reset_question_answer, iplanet_am_user_password_resetoptions, iplanet_am_user_success_url, kbaActiveIndex, kbaInfo, mail, manager, postaladdress, preferredlocale, sn, sunIdentityMSISDNNumber, telephonenumber, uid, userpassword`

## User Password Attribute Name

Column for user passwords.

**ssoadm** attribute: `sun-opensso-database-UserPasswordAttr`

Default: `userpassword`

## User ID Attribute Name

Column for user IDs.

**ssoadm** attribute: `sun-opensso-database-UserIDAttr`

Default: `uid`

## Attribute Name of User Status

Column to check/set user status.

**ssoadm** attribute: `sun-opensso-database-UserStatusAttr`

Default: `inetuserstatus`

## User Status Active Value

Active users have the user status set to this value.

**ssoadm** attribute: `sun-opensso-database-activeValue`

Default: `Active`

## User Status Inactive Value

Inactive users have the user status set to this value.

**ssoadm** attribute: `sun-opensso-database-inactiveValue`

Default: `Inactive`

## Users Search Attribute in Database

Key for looking up user profiles by name.

**ssoadm** attribute: `sun-opensso-database-config-users-search-attribute`

Default: `cn`

### Database Membership table name

Table to store group profiles.

**ssoadm** attribute: `sun-opensso-database-MembershipTableName`

Default: `groups`

### Membership ID Attribute Name

Column for group IDs.

**ssoadm** attribute: `sun-opensso-database-MembershipIDAttr`

Default: `group_name`

### Membership Search Attribute in Database

Key for looking up group profiles by name.

**ssoadm** attribute: `sun-opensso-database-membership-search-attribute`

Default: `cn`

## 10.1.4. Generic LDAPv3 Configuration Properties

Use these attributes when configuring Generic LDAPv3 compliant data stores:

**ssoadm** service name: `sunIdentityRepositoryService`

### Name

Name for the data store configuration.

### Load schema when finished

Add appropriate LDAP schema to the directory server when saving the configuration. The LDAP Bind DN user must have access to perform this operation.

This attribute is not available for use with the **ssoadm** command.

Default: `false`

### LDAP Server

`host:port` to contact the directory server, with optional `|server_ID|site_ID` for deployments with multiple servers and sites.

OpenAM uses the optional settings to determine which directory server to contact first. OpenAM tries to contact directory servers in the following priority order, with highest priority first:

1. The first directory server in the list whose *server\_ID* matches the current OpenAM server.
2. The first directory server in the list whose *site\_ID* matches the current OpenAM server.
3. The first directory server in the remaining list.

If the directory server is not available, OpenAM proceeds to the next directory server in the list.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-ldap-server`

Default: `host:port` of the initial directory server configured for this OpenAM server

### LDAP Bind DN

Bind DN for connecting to the directory server. Some OpenAM capabilities require write access to directory entries.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-authid`

### LDAP Bind Password

Bind password for connecting to the directory server.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-authpw`

### LDAP Organization DN

The base DN under which to find user and group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-organization_name`

Default: `base-dn`

### LDAP Connection Mode

Whether to use LDAP, LDAPS or StartTLS to connect to the directory server. When LDAPS or StartTLS are enabled, OpenAM must be able to trust server certificates, either because the server certificates were signed by a CA whose certificate is already included in the trust store used by the container where OpenAM runs, or because you imported the certificates into the trust store.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-connection-mode`

Possible values: `LDAP`, `LDAPS`, and `StartTLS`

### LDAP Connection Pool Maximum Size

Maximum number of connections to the directory server. Make sure the directory service can cope with the maximum number of client connections across all servers.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-connection_pool_max_size`

Default: 10

## LDAP Connection Heartbeat Interval

How often to send a heartbeat request to the directory server to ensure that the connection does not remain idle. Some network administrators configure firewalls and load balancers to drop connections that are idle for too long. You can turn this off by setting the value to 0 or to a negative number. To set the units for the interval, use LDAP Connection Heartbeat Time Unit.

**ssoadm** attribute: `openam-idrepo-ldapv3-heartbeat-interval`

Default: 10

## LDAP Connection Heartbeat Time Unit

Time unit for the LDAP Connection Heartbeat Interval setting.

**ssoadm** attribute: `openam-idrepo-ldapv3-heartbeat-timeunit`

Default: `second`

## Maximum Results Returned from Search

A cap for the number of search results to request. For example, when using the Subjects tab to view profiles, even if you set Configuration > Console > Administration > Maximum Results Returned from Search to a larger number, OpenAM does not exceed this setting. Rather than raise this number, consider narrowing your search to match fewer directory entries.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-max-result`

Default: 1000

## Search Timeout

Maximum time to wait for search results in seconds. Does not apply to persistent searches.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-time-limit`

Default: 10

## LDAPv3 Plugin Search Scope

LDAP searches can apply to a single entry (SCOPE\_BASE), entries directly below the search DN (SCOPE\_ONE), or all entries below the search DN (SEARCH\_SUB).

**ssoadm** attribute: `sun-idrepo-ldapv3-config-search-scope`

Default: `SCOPE_SUB`

## LDAPv3 Repository Plugin Class Name

OpenAM identity repository implementation.

**ssoadm** attribute: `sunIdRepoClass`

Default: `org.forgerock.openam.idrepo.ldap.DJLDAPv3Repo`

## Attribute Name Mapping

Map of OpenAM profile attribute names to directory server attribute names.

**ssoadm** attribute: `sunIdRepoAttributeMapping`

## LDAPv3 Plugin Supported Types and Operations

Map of OpenAM operations that can be performed in the specified OpenAM contexts.

**ssoadm** attribute: `sunIdRepoSupportedOperations`

Default: `realm=read,create,edit,delete,service, user=read,create,edit,delete,service, group=read,create,edit,delete`

## LDAP Users Search Attribute

When searching for a user by name, match values against this attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-users-search-attribute`

Default: `uid`

## LDAP Users Search Filter

When searching for users, apply this LDAP search filter as well.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-users-search-filter`

Default: `(objectclass=inetorgperson)`

## LDAP People Container Naming Attribute

RDN attribute of the LDAP base DN which contains user profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-people-container-name`

## LDAP People Container Value

RDN attribute value of the LDAP base DN which contains user profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-people-container-value`

## LDAP User Object Class

User profiles have these LDAP object classes.

OpenAM handles only those attributes listed in this setting. OpenAM discards any unlisted attributes from requests and the request proceeds without the attribute.

For example, with default settings, if you request that OpenAM execute a search that asks for the `mailAlternateAddress` attribute, OpenAM does the search, but does not request `mailAlternateAddress`. In the same way, OpenAM does perform an update operation with a request to set the value of an unlisted attribute like `mailAlternateAddress`, but it drops the unlisted attribute from the update request.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-user-objectclass`

Default: `inetorgperson, inetUser, organizationalPerson, person, top,`

## LDAP User Attributes

User profiles have these LDAP attributes.

OpenAM handles only those attributes listed in this setting. OpenAM discards any unlisted attributes from requests and the request proceeds without the attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-user-attributes`

Default: `uid, caCertificate, authorityRevocationList, inetUserStatus, mail, sn, manager, userPassword, adminRole, objectClass, givenName, memberOf, cn, telephoneNumber, preferredlanguage, userCertificate, postalAddress, dn, employeeNumber, distinguishedName`

## Create User Attribute Mapping

When creating a user profile, apply this map of OpenAM profile attribute names to directory server attribute names.

Attributes not mapped to another attribute (for example, `cn`) and attributes mapped to themselves (for example, `cn=cn`) take the value of the username unless the attribute values are provided when creating the profile. The object classes for user profile LDAP entries generally require Common Name (`cn`) and Surname (`sn`) attributes, so this prevents an LDAP constraint violation when performing the add operation.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-createuser-attr-mapping`

Default: `cn, sn`

## Attribute Name of User Status

Attribute to check/set user status.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-isactive`

Default: `inetuserstatus`

## User Status Active Value

Active users have the user status attribute set to this value.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-active`

Default: `Active`

## User Status Inactive Value

Inactive users have the user status attribute set to this value.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-inactive`



Default: `Inactive`

### Authentication Naming Attribute

RDN attribute for building the bind DN when given a username and password to authenticate a user against the directory server.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-auth-naming-attr`

Default: `uid`

### LDAP Groups Search Attribute

When searching for a group by name, match values against this attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-groups-search-attribute`

Default: `cn`

### LDAP Groups Search Filter

When searching for groups, apply this LDAP search filter as well.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-groups-search-filter`

Default: `(objectclass=groupOfUniqueNames)`

### LDAP Groups Container Naming Attribute

RDN attribute of the LDAP base DN which contains group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-container-name`

Default: `ou`

### LDAP Groups Container Value

RDN attribute value of the LDAP base DN which contains group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-container-value`

Default: `groups`

### LDAP Groups Object Class

Group profiles have these LDAP object classes.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-objectclass`

Default: `groupofuniquenames, top`

### LDAP Groups Attributes

Group profiles have these LDAP attributes.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-attributes`

Default: `ou, cn, description, dn, objectclass, uniqueMember`

### Attribute Name for Group Membership

LDAP attribute in the member's LDAP entry whose values are the groups to which a member belongs.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-memberof`

### Attribute Name of Unique Member

Attribute in the group's LDAP entry whose values are the members of the group.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-uniquemember`

Default: `uniqueMember`

### Attribute Name of Group Member URL

Attribute in the dynamic group's LDAP entry whose value is a URL specifying the members of the group.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-memberurl`

Default: `memberUrl`

### Default Group Member's User DN

DN of member added to all newly created groups.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-dftgroupmember`

### Persistent Search Base DN

Base DN for LDAP-persistent searches used to receive notification of changes in directory server data.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-psearchbase`

Default: `base-dn`

### Persistent Search Filter

LDAP filter to apply when performing persistent searches.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-psearch-filter`

Default: `(objectclass=*)`

### Persistent Search Scope

LDAP searches can apply to a single entry (SCOPE\_BASE), entries directly below the search DN (SCOPE\_ONE), or all entries below the search DN (SEARCH\_SUB).

**ssoadm** attribute: `sun-idrepo-ldapv3-config-psearch-scope`

Default: `SCOPE_SUB`

### The Delay Time Between Retries

How long to wait after receiving an error result that indicates OpenAM should try the LDAP operation again.

**ssoadm** attribute: `com.ipplanet.am.ldap.connection.delay.between.retries`

Default: 1000 milliseconds

### DN Cache Enabled

Whether to enable the DN cache, which is used to cache DN lookups that can happen in bursts during authentication. As the cache can become stale when a user is moved or renamed, enable DN caching when the directory service allows move/rename operations (Mod DN), and when OpenAM uses persistent searches to obtain notification of such updates.

**ssoadm** attribute: `sun-idrepo-ldapv3-dncache-enabled`

Default: false

### DN Cache Size

Maximum number of DN's cached when caching is enabled.

**ssoadm** attribute: `sun-idrepo-ldapv3-dncache-size`

Default: 1500 items

## 10.1.5. OpenDJ Configuration Properties

Use these attributes when configuring OpenDJ data stores:

**ssoadm** service name: `sunIdentityRepositoryService`

### Name

Name for the data store configuration.

### Load schema when finished

Add appropriate LDAP schema to the directory server when saving the configuration. The LDAP Bind DN user must have access to perform this operation.

This attribute is not available for use with the **ssoadm** command.

Default: false

### LDAP Server

`host:port` to contact the directory server, with optional `|server_ID|site_ID` for deployments with multiple servers and sites.

OpenAM uses the optional settings to determine which directory server to contact first. OpenAM tries to contact directory servers in the following priority order, with highest priority first:

1. The first directory server in the list whose `server_ID` matches the current OpenAM server.
2. The first directory server in the list whose `site_ID` matches the current OpenAM server.
3. The first directory server in the remaining list.

If the directory server is not available, OpenAM proceeds to the next directory server in the list.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-ldap-server`

Default: `host:port` of the initial directory server configured for this OpenAM server

### LDAP Bind DN

Bind DN for connecting to the directory server. Some OpenAM capabilities require write access to directory entries.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-authid`

### LDAP Bind Password

Bind password for connecting to the directory server.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-authpw`

### LDAP Organization DN

The base DN under which to find user and group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-organization_name`

Default: `base-dn`

### LDAP Connection Mode

Whether to use LDAP, LDAPS or StartTLS to connect to the directory server. When LDAPS or StartTLS are enabled, OpenAM must be able to trust server certificates, either because the server certificates were signed by a CA whose certificate is already included in the trust store used by the container where OpenAM runs, or because you imported the certificates into the trust store.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-connection-mode`

Possible values: `LDAP`, `LDAPS`, and `StartTLS`

### LDAP Connection Pool Maximum Size

Maximum number of connections to the directory server. Make sure the directory service can cope with the maximum number of client connections across all servers.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-connection_pool_max_size`

Default: 10

### LDAP Connection Heartbeat Interval

How often to send a heartbeat request to the directory server to ensure that the connection does not remain idle. Some network administrators configure firewalls and load balancers to drop connections that are idle for too long. You can turn this off by setting the value to 0 or to a negative number. To set the units for the interval, use LDAP Connection Heartbeat Time Unit.

**ssoadm** attribute: `openam-idrepo-ldapv3-heartbeat-interval`

Default: 10

### LDAP Connection Heartbeat Time Unit

Time unit for the LDAP Connection Heartbeat Interval setting.

**ssoadm** attribute: `openam-idrepo-ldapv3-heartbeat-timeunit`

Default: `second`

### Maximum Results Returned from Search

A cap for the number of search results to request. For example, when using the Subjects tab to view profiles, even if you set Configuration > Console > Administration > Maximum Results Returned from Search to a larger number, OpenAM does not exceed this setting. Rather than raise this number, consider narrowing your search to match fewer directory entries.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-max-result`

Default: 1000

### Search Timeout

Maximum time to wait for search results in seconds. Does not apply to persistent searches.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-time-limit`

Default: 10

### LDAPv3 Plugin Search Scope

LDAP searches can apply to a single entry (SCOPE\_BASE), entries directly below the search DN (SCOPE\_ONE), or all entries below the search DN (SEARCH\_SUB).

**ssoadm** attribute: `sun-idrepo-ldapv3-config-search-scope`

Default: `SCOPE_SUB`

### LDAPv3 Repository Plugin Class Name

OpenAM identity repository implementation.

**ssoadm** attribute: `sunIdRepoClass`

Default: `org.forgerock.openam.idrepo.ldap.DJLDAPv3Repo`

### Attribute Name Mapping

Map of OpenAM profile attribute names to directory server attribute names.

**ssoadm** attribute: `sunIdRepoAttributeMapping`

### LDAPv3 Plugin Supported Types and Operations

Map of OpenAM operations that can be performed in the specified OpenAM contexts.

**ssoadm** attribute: `sunIdRepoSupportedOperations`

Default: `realm=read,create,edit,delete,service, user=read,create,edit,delete,service, group=read,create,edit,delete`

### LDAP Users Search Attribute

When searching for a user by name, match values against this attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-users-search-attribute`

Default: `uid`

### LDAP Users Search Filter

When searching for users, apply this LDAP search filter as well.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-users-search-filter`

Default: `(objectclass=inetorgperson)`

### LDAP People Container Naming Attribute

RDN attribute of the LDAP base DN which contains user profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-people-container-name`

Default: `ou`

### LDAP People Container Value

RDN attribute value of the LDAP base DN which contains user profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-people-container-value`

Default: `people`

### LDAP User Object Class

User profiles have these LDAP object classes.

OpenAM handles only those attributes listed in this setting. OpenAM discards any unlisted attributes from requests and the request proceeds without the attribute.

For example, with default settings, if you request that OpenAM execute a search that asks for the `mailAlternateAddress` attribute, OpenAM does the search, but does not request `mailAlternateAddress`. In the same way, OpenAM does perform an update operation with a request to set the value of an unlisted attribute like `mailAlternateAddress`, but it drops the unlisted attribute from the update request.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-user-objectclass`

Default: `devicePrintProfilesContainer, forgerock-am-dashboard-service, iPlanetPreferences, inetorgperson, inetuser, iplanet-am-auth-configuration-service, iplanet-am-managed-person, iplanet-am-user-service, kbaInfoContainer, oathDeviceProfilesContainer, organizationalperson, person, pushDeviceProfilesContainer, sunAMAuthAccountLockout, sunFMSAML2NameIdentifier, sunFederationManagerDataStore, sunIdentityServerLibertyPPService, top`

## LDAP User Attributes

User profiles have these LDAP attributes.

OpenAM handles only those attributes listed in this setting. OpenAM discards any unlisted attributes from requests and the request proceeds without the attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-user-attributes`

Default: `adminRole, assignedDashboard, authorityRevocationList, caCertificate, cn, createTimestamp, devicePrintProfiles, distinguishedName, dn, employeeNumber, givenName, inetUserHttpURL, inetUserStatus, iplanet-am-auth-configuration, iplanet-am-session-destroy-sessions, iplanet-am-session-get-valid-sessions, iplanet-am-session-max-caching-time, iplanet-am-session-max-idle-time, iplanet-am-session-max-session-time, iplanet-am-session-quota-limit, iplanet-am-session-service-status, iplanet-am-user-account-life, iplanet-am-user-admin-start-dn, iplanet-am-user-alias-list, iplanet-am-user-auth-config, iplanet-am-user-auth-modules, iplanet-am-user-failure-url, iplanet-am-user-federation-info, iplanet-am-user-federation-info-key, iplanet-am-user-login-status, iplanet-am-user-password-reset-force-reset, iplanet-am-user-password-reset-options, iplanet-am-user-password-reset-question-answer, iplanet-am-user-success-url, kbaActiveIndex, kbaInfo, mail, manager, memberOf, modifyTimestamp, oath2faEnabled, oathDeviceProfiles, objectClass, postalAddress, preferredLocale, preferredLanguage, preferredtimezone, pushDeviceProfiles, sn, sun-fm-saml2-nameid-info, sun-fm-saml2-nameid-infokey, sunAMAuthInvalidAttemptsData, sunIdentityMSISDNNumber, sunIdentityServerDiscoEntries, sunIdentityServerPPAddressCard, sunIdentityServerPPCommonNameAltCN, sunIdentityServerPPCommonNameCN, sunIdentityServerPPCommonNameFN, sunIdentityServerPPCommonNameMN, sunIdentityServerPPCommonNamePT, sunIdentityServerPPCommonNameSN, sunIdentityServerPPDemographicsAge, sunIdentityServerPPDemographicsBirthDay, sunIdentityServerPPDemographicsDisplayLanguage, sunIdentityServerPPDemographicsLanguage, sunIdentityServerPPDemographicsTimeZone, sunIdentityServerPPEmergencyContact, sunIdentityServerPPEmploymentIdentityAlt0, sunIdentityServerPPEmploymentIdentityJobTitle, sunIdentityServerPPEmploymentIdentityOrg, sunIdentityServerPPEncryPTKey, sunIdentityServerPPFacadeGreetSound, sunIdentityServerPPFacadeMugShot, sunIdentityServerPPFacadeNamePronounced, sunIdentityServerPPFacadeWebSite, sunIdentityServerPPFacadeGreetmesound, sunIdentityServerPPInformalName, sunIdentityServerPPLegalIdentityAltIdType, sunIdentityServerPPLegalIdentityAltIdValue, sunIdentityServerPPLegalIdentityDOB, sunIdentityServerPPLegalIdentityGender, sunIdentityServerPPLegalIdentityLegalName, sunIdentityServerPPLegalIdentityMaritalStatus,`

`sunIdentityServerPPLegalIdentityVATIdType`, `sunIdentityServerPPLegalIdentityVATIdValue`,  
`sunIdentityServerPPMsgContact`, `sunIdentityServerPPSignKey`, `telephoneNumber`, `uid`, `userCertificate`,  
`userPassword`

## Create User Attribute Mapping

When creating a user profile, apply this map of OpenAM profile attribute names to directory server attribute names.

Attributes not mapped to another attribute (for example, `cn`) and attributes mapped to themselves (for example, `cn=cn`) take the value of the username unless the attribute values are provided when creating the profile. The object classes for user profile LDAP entries generally require Common Name (`cn`) and Surname (`sn`) attributes, so this prevents an LDAP constraint violation when performing the add operation.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-createuser-attr-mapping`

Default: `cn`, `sn`

## Attribute Name of User Status

Attribute to check/set user status.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-isactive`

Default: `inetuserstatus`

## User Status Active Value

Active users have the user status attribute set to this value.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-active`

Default: `Active`

## User Status Inactive Value

Inactive users have the user status attribute set to this value.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-inactive`

Default: `Inactive`

## Authentication Naming Attribute

RDN attribute for building the bind DN when given a username and password to authenticate a user against the directory server.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-auth-naming-attr`

Default: `uid`

## LDAP Groups Search Attribute

When searching for a group by name, match values against this attribute.



**ssoadm** attribute: `sun-idrepo-ldapv3-config-groups-search-attribute`

Default: `cn`

### LDAP Groups Search Filter

When searching for groups, apply this LDAP search filter as well.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-groups-search-filter`

Default: `(objectclass=groupOfUniqueNames)`

### LDAP Groups Container Naming Attribute

RDN attribute of the LDAP base DN which contains group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-container-name`

Default: `ou`

### LDAP Groups Container Value

RDN attribute value of the LDAP base DN which contains group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-container-value`

Default: `groups`

### LDAP Groups Object Class

Group profiles have these LDAP object classes.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-objectclass`

Default: `groupofuniquenames, top`

### LDAP Groups Attributes

Group profiles have these LDAP attributes.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-attributes`

Default: `cn, dn, objectclass, uniqueMember`

### Attribute Name for Group Membership

LDAP attribute in the member's LDAP entry whose values are the groups to which a member belongs.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-memberof`

### Attribute Name of Unique Member

Attribute in the group's LDAP entry whose values are the members of the group.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-uniquemember`

Default: `uniqueMember`

### Persistent Search Base DN

Base DN for LDAP-persistent searches used to receive notification of changes in directory server data.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-psearchbase`

Default: `base-dn`

### Persistent Search Filter

LDAP filter to apply when performing persistent searches.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-psearch-filter`

Default: `(objectclass=*)`

### Persistent Search Scope

LDAP searches can apply to a single entry (SCOPE\_BASE), entries directly below the search DN (SCOPE\_ONE), or all entries below the search DN (SEARCH\_SUB).

**ssoadm** attribute: `sun-idrepo-ldapv3-config-psearch-scope`

Default: `SCOPE_SUB`

### The Delay Time Between Retries

How long to wait after receiving an error result that indicates OpenAM should try the LDAP operation again.

The OpenDJ data store uses this setting only for persistent searches.

**ssoadm** attribute: `com.ipplanet.am.ldap.connection.delay.between.retries`

Default: 1000 milliseconds

### DN Cache Enabled

Whether to enable the DN cache, which is used to cache DN lookups that can happen in bursts during authentication. As the cache can become stale when a user is moved or renamed, enable DN caching when the directory service allows move/rename operations (Mod DN), and when OpenAM uses persistent searches to obtain notification of such updates.

**ssoadm** attribute: `sun-idrepo-ldapv3-dncache-enabled`

Default: true

### DN Cache Size

Maximum number of DN's cached when caching is enabled.

**ssoadm** attribute: `sun-idrepo-ldapv3-dncache-size`

Default: 1500 items

### 10.1.6. Sun/Oracle DSEE Configuration Properties

Use these attributes when configuring data stores for Oracle DSEE or Sun DSEE using OpenAM schema:

**ssoadm** service name: `sunIdentityRepositoryService`

#### Name

Name for the data store configuration.

#### Load schema when finished

Add appropriate LDAP schema to the directory server when saving the configuration. The LDAP Bind DN user must have access to perform this operation.

This attribute is not available for use with the **ssoadm** command.

Default: false

#### LDAP Server

`host:port` to contact the directory server, with optional `|server_ID|site_ID` for deployments with multiple servers and sites.

OpenAM uses the optional settings to determine which directory server to contact first. OpenAM tries to contact directory servers in the following priority order, with highest priority first:

1. The first directory server in the list whose `server_ID` matches the current OpenAM server.
2. The first directory server in the list whose `site_ID` matches the current OpenAM server.
3. The first directory server in the remaining list.

If the directory server is not available, OpenAM proceeds to the next directory server in the list.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-ldap-server`

Default: `host:port` of the initial directory server configured for this OpenAM server.

#### LDAP Bind DN

Bind DN for connecting to the directory server. Some OpenAM capabilities require write access to directory entries.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-authid`

Default: `cn=dsameuser,ou=DSAME Users,base-dn`

## LDAP Bind Password

Bind password for connecting to the directory server.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-authpw`

## LDAP Organization DN

The base DN under which to find user and group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-organization_name`

Default: `base-dn`

## LDAP Connection Mode

Whether to use LDAP, LDAPS or StartTLS to connect to the directory server. When LDAPS or StartTLS are enabled, OpenAM must be able to trust server certificates, either because the server certificates were signed by a CA whose certificate is already included in the trust store used by the container where OpenAM runs, or because you imported the certificates into the trust store.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-connection-mode`

Possible values: `LDAP`, `LDAPS`, and `StartTLS`

## LDAP Connection Pool Maximum Size

Maximum number of connections to the directory server. Make sure the directory service can cope with the maximum number of client connections across all servers.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-connection_pool_max_size`

Default: 10

## LDAP Connection Heartbeat Interval

How often to send a heartbeat request to the directory server to ensure that the connection does not remain idle. Some network administrators configure firewalls and load balancers to drop connections that are idle for too long. You can turn this off by setting the value to 0 or to a negative number. To set the units for the interval, use LDAP Connection Heartbeat Time Unit.

**ssoadm** attribute: `openam-idrepo-ldapv3-heartbeat-interval`

Default: 10

## LDAP Connection Heartbeat Time Unit

Time unit for the LDAP Connection Heartbeat Interval setting.

**ssoadm** attribute: `openam-idrepo-ldapv3-heartbeat-timeunit`

Default: `second`

## Maximum Results Returned from Search

A cap for the number of search results to request. For example, when using the Subjects tab to view profiles, even if you set Configuration > Console > Administration > Maximum Results Returned from Search to a larger number, OpenAM does not exceed this setting. Rather than raise this number, consider narrowing your search to match fewer directory entries.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-max-result`

Default: 1000

## Search Timeout

Maximum time to wait for search results in seconds. Does not apply to persistent searches.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-time-limit`

Default: 10

## LDAPv3 Plugin Search Scope

LDAP searches can apply to a single entry (SCOPE\_BASE), entries directly below the search DN (SCOPE\_ONE), or all entries below the search DN (SEARCH\_SUB).

**ssoadm** attribute: `sun-idrepo-ldapv3-config-search-scope`

Default: `SCOPE_SUB`

## LDAPv3 Repository Plugin Class Name

OpenAM identity repository implementation.

**ssoadm** attribute: `sunIdRepoClass`

Default: `org.forgerock.openam.idrepo.ldap.DJLDAPv3Repo`

## Attribute Name Mapping

Map of OpenAM profile attribute names to directory server attribute names.

**ssoadm** attribute: `sunIdRepoAttributeMapping`

## LDAPv3 Plugin Supported Types and Operations

Map of OpenAM operations that can be performed in the specified OpenAM contexts.

**ssoadm** attribute: `sunIdRepoSupportedOperations`

Default: `filteredrole=read,create,edit,delete, group=read,create,edit,delete, realm=read,create,edit,delete,service, role=read,create,edit,delete, user=read,create,edit,delete,service`

## LDAP Users Search Attribute

When searching for a user by name, match values against this attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-users-search-attribute`

Default: `uid`

## LDAP Users Search Filter

When searching for users, apply this LDAP search filter as well.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-users-search-filter`

Default: `(objectclass=inetorgperson)`

## LDAP People Container Naming Attribute

RDN attribute of the LDAP base DN which contains user profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-people-container-name`

Default: `ou`

## LDAP People Container Value

RDN attribute value of the LDAP base DN which contains user profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-people-container-value`

Default: `people`

## LDAP User Object Class

User profiles have these LDAP object classes.

OpenAM handles only those attributes listed in this setting. OpenAM discards any unlisted attributes from requests and the request proceeds without the attribute.

For example, with default settings, if you request that OpenAM execute a search that asks for the `mailAlternateAddress` attribute, OpenAM does the search, but does not request `mailAlternateAddress`. In the same way, OpenAM does perform an update operation with a request to set the value of an unlisted attribute like `mailAlternateAddress`, but it drops the unlisted attribute from the update request.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-user-objectclass`

Default: `devicePrintProfilesContainer, forgerock-am-dashboard-service, iPlanetPreferences, inetadmin, inetorgperson, inetuser, iplanet-am-auth-configuration-service, iplanet-am-managed-person, iplanet-am-user-service, kbaInfoContainer, oathDeviceProfilesContainer, organizationalperson, person, pushDeviceProfilesContainer, sunAMAuthAccountLockout, sunFMSAML2NameIdentifier, sunFederationManagerDataStore, sunIdentityServerLibertyPPService, top`

## LDAP User Attributes

User profiles have these LDAP attributes.

OpenAM handles only those attributes listed in this setting. OpenAM discards any unlisted attributes from requests and the request proceeds without the attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-user-attributes`

Default: `assignedDashboard, authorityRevocationList, caCertificate, cn, createTimestamp, devicePrintProfiles, distinguishedName, adminRole, dn, employeeNumber, givenName, inetUserHttpURL, inetUserStatus, iplanet-am-auth-configuration, iplanet-am-session-destroy-sessions, iplanet-am-session-get-valid-sessions, iplanet-am-session-max-caching-time, iplanet-am-session-max-idle-time, iplanet-am-session-max-session-time, iplanet-am-session-quota-limit, iplanet-am-session-service-status, iplanet-am-static-group-dn, iplanet-am-user-account-life, iplanet-am-user-admin-start-dn, iplanet-am-user-alias-list, iplanet-am-user-auth-config, iplanet-am-user-auth-modules, iplanet-am-user-failure-url, iplanet-am-user-federation-info, iplanet-am-user-federation-info-key, iplanet-am-user-login-status, iplanet-am-user-password-reset-force-reset, iplanet-am-user-password-reset-options, iplanet-am-user-password-reset-question-answer, iplanet-am-user-success-url, kbaActiveIndex, kbaInfo, mail, manager, memberOf, modifyTimestamp, oath2faEnabled, oathDeviceProfiles, objectClass, postalAddress, preferredLocale, preferredlanguage, preferredtimezone, pushDeviceProfiles, sn, sun-fm-saml2-nameid-info, sun-fm-saml2-nameid-infokey, sunAMAuthInvalidAttemptsData, sunIdentityMSISDNNumber, sunIdentityServerDiscoEntries, sunIdentityServerPPAddressCard, sunIdentityServerPPCommonNameAltCN, sunIdentityServerPPCommonNameCN, sunIdentityServerPPCommonNameFN, sunIdentityServerPPCommonNameMN, sunIdentityServerPPCommonNamePT, sunIdentityServerPPCommonNameSN, sunIdentityServerPPDemographicsAge, sunIdentityServerPPDemographicsBirthDay, sunIdentityServerPPDemographicsDisplayLanguage, sunIdentityServerPPDemographicsLanguage, sunIdentityServerPPDemographicsTimeZone, sunIdentityServerPPEmergencyContact, sunIdentityServerPPEmploymentIdentityAlt0, sunIdentityServerPPEmploymentIdentityJobTitle, sunIdentityServerPPEmploymentIdentityOrg, sunIdentityServerPPEncryPTKey, sunIdentityServerPPFacadeGreetSound, sunIdentityServerPPFacadeMugShot, sunIdentityServerPPFacadeNamePronounced, sunIdentityServerPPFacadeWebSite, sunIdentityServerPPFacadeGreetmesound, sunIdentityServerPPInformalName, sunIdentityServerPPLegalIdentityAltIdType, sunIdentityServerPPLegalIdentityAltIdValue, sunIdentityServerPPLegalIdentityDOB, sunIdentityServerPPLegalIdentityGender, sunIdentityServerPPLegalIdentityLegalName, sunIdentityServerPPLegalIdentityMaritalStatus, sunIdentityServerPPLegalIdentityVATIdType, sunIdentityServerPPLegalIdentityVATIdValue, sunIdentityServerPPMsgContact, sunIdentityServerPPSignKey, telephoneNumber, uid, userCertificate, userPassword`

## Create User Attribute Mapping

When creating a user profile, apply this map of OpenAM profile attribute names to directory server attribute names.

Attributes not mapped to another attribute (for example, `cn`) and attributes mapped to themselves (for example, `cn=cn`) take the value of the username unless the attribute values are provided when creating the profile. The object classes for user profile LDAP entries generally require Common Name (`cn`) and Surname (`sn`) attributes, so this prevents an LDAP constraint violation when performing the add operation.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-createuser-attr-mapping`

Default: `cn, sn`

### Attribute Name of User Status

Attribute to check/set user status.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-isactive`

Default: `inetuserstatus`

### User Status Active Value

Active users have the user status attribute set to this value.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-active`

Default: `Active`

### User Status Inactive Value

Inactive users have the user status attribute set to this value.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-inactive`

Default: `Inactive`

### Authentication Naming Attribute

RDN attribute for building the bind DN when given a username and password to authenticate a user against the directory server.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-auth-naming-attr`

Default: `uid`

### LDAP Groups Search Attribute

When searching for a group by name, match values against this attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-groups-search-attribute`

Default: `cn`

### LDAP Groups Search Filter

When searching for groups, apply this LDAP search filter as well.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-groups-search-filter`

Default: `(objectclass=groupOfUniqueNames)`

### LDAP Groups Container Naming Attribute

RDN attribute of the LDAP base DN which contains group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-container-name`



Default: `ou`

### LDAP Groups Container Value

RDN attribute value of the LDAP base DN which contains group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-container-value`

Default: `groups`

### LDAP Groups Object Class

Group profiles have these LDAP object classes.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-objectclass`

Default: `groupofuniquenames, iplanet-am-managed-group, iplanet-am-managed-static-group, groupofurls, top`

### LDAP Groups Attributes

Group profiles have these LDAP attributes.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-attributes`

Default: `cn, iplanet-am-group-subscribable, dn, objectclass, uniqueMember`

### Attribute Name for Group Membership

LDAP attribute in the member's LDAP entry whose values are the groups to which a member belongs.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-memberof`

### Attribute Name of Unique Member

Attribute in the group's LDAP entry whose values are the members of the group.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-uniquemember`

Default: `uniqueMember`

### Attribute Name of Group Member URL

Attribute in the dynamic group's LDAP entry whose values are LDAP URLs specifying members of the group.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-memberurl`

Default: `memberUrl`

### LDAP Roles Search Attribute

When searching for a role by name, match values against this attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-roles-search-attribute`

Default: `cn`

### LDAP Roles Search Filter

When searching for roles, apply this LDAP search filter as well.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-roles-search-filter`

Default: `(&(objectclass=ldapsubentry)(objectclass=nsmanagedroledefinition))`

### LDAP Roles Object Class

Role profiles have these LDAP object classes.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-role-objectclass`

Default: `ldapsubentry, nsmanagedroledefinition, nsroledefinition, nssimpleroledefinition, top`

### LDAP Filter Roles Search Attribute

When searching for a filtered role by name, match values against this attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-filterroles-search-attribute`

Default: `cn`

### LDAP Filter Roles Search Filter

When searching for filtered roles, apply this LDAP search filter as well.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-filterroles-search-filter`

Default: `(&(objectclass=ldapsubentry)(objectclass=nsfilteredroledefinition))`

### LDAP Filter Roles Object Class

Filtered role profiles have these LDAP object classes.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-filterrole-objectclass`

Default: `ldapsubentry, nscomplexroledefinition, nsfilteredroledefinition, nsroledefinition`

### LDAP Filter Roles Attributes

Filtered role profiles have these LDAP attributes.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-filterrole-attributes`

Default: `nsRoleFilter`

### Attribute Name for Filtered Role Membership

LDAP attribute in the member's LDAP entry whose values are the filtered roles to which a member belongs.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-nsrole`

Default: `nsrole`

### Attribute Name of Role Membership

LDAP attribute in the member's LDAP entry whose values are the roles to which a member belongs.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-nsroledn`

Default: `nsRoleDN`

### Attribute Name of Filtered Role Filter

LDAP attribute whose values are the filters for filtered roles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-nsrolefilter`

Default: `nsRoleFilter`

### Persistent Search Base DN

Base DN for LDAP-persistent searches used to receive notification of changes in directory server data.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-psearchbase`

Default: `base-dn`

### Persistent Search Filter

LDAP filter to apply when performing persistent searches.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-psearch-filter`

Default: `(objectclass=*)`

### Persistent Search Scope

LDAP searches can apply to a single entry (SCOPE\_BASE), entries directly below the search DN (SCOPE\_ONE), or all entries below the search DN (SEARCH\_SUB).

**ssoadm** attribute: `sun-idrepo-ldapv3-config-psearch-scope`

Default: `SCOPE_SUB`

### The Delay Time Between Retries

How long to wait after receiving an error result that indicates OpenAM should try the LDAP operation again.

**ssoadm** attribute: `com.iplanet.am.ldap.connection.delay.between.retries`

Default: 1000 milliseconds

### DN Cache Enabled

Whether to enable the DN cache, which is used to cache DN lookups that can happen in bursts during authentication. As the cache can become stale when a user is moved or renamed, enable DN caching when the directory service allows move/rename operations (Mod DN), and when OpenAM uses persistent searches to obtain notification of such updates.

**ssoadm** attribute: `sun-idrepo-ldapv3-dncache-enabled`

Default: true

### DN Cache Size

Maximum number of DN's cached when caching is enabled.

**ssoadm** attribute: `sun-idrepo-ldapv3-dncache-size`

Default: 1500 items

## 10.1.7. Tivoli Directory Server Configuration Properties

Use these attributes when configuring Tivoli Directory Server data stores:

**ssoadm** service name: `sunIdentityRepositoryService`

### Name

Name for the data store configuration.

### Load schema when finished

Add appropriate LDAP schema to the directory server when saving the configuration. The LDAP Bind DN user must have access to perform this operation.

This attribute is not available for use with the **ssoadm** command.

Default: false

### LDAP Server

`host:port` to contact the directory server, with optional `|server_ID|site_ID` for deployments with multiple servers and sites.

OpenAM uses the optional settings to determine which directory server to contact first. OpenAM tries to contact directory servers in the following priority order, with highest priority first.

1. The first directory server in the list whose `server_ID` matches the current OpenAM server.
2. The first directory server in the list whose `site_ID` matches the current OpenAM server.

3. The first directory server in the remaining list.

If the directory server is not available, OpenAM proceeds to the next directory server in the list.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-ldap-server`

Default: `host:port` of the initial directory server configured for this OpenAM server

### LDAP Bind DN

Bind DN for connecting to the directory server. Some OpenAM capabilities require write access to directory entries.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-authid`

### LDAP Bind Password

Bind password for connecting to the directory server.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-authpw`

### LDAP Organization DN

The base DN under which to find user and group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-organization_name`

Default: `base-dn`

### LDAP Connection Mode

Whether to use LDAP, LDAPS or StartTLS to connect to the directory server. When LDAPS or StartTLS are enabled, OpenAM must be able to trust server certificates, either because the server certificates were signed by a CA whose certificate is already included in the trust store used by the container where OpenAM runs, or because you imported the certificates into the trust store.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-connection-mode`

Possible values: `LDAP`, `LDAPS`, and `StartTLS`

### LDAP Connection Pool Maximum Size

Maximum number of connections to the directory server. Make sure the directory service can cope with the maximum number of client connections across all servers.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-connection_pool_max_size`

Default: 10

### LDAP Connection Heartbeat Interval

How often to send a heartbeat request to the directory server to ensure that the connection does not remain idle. Some network administrators configure firewalls and load balancers to

drop connections that are idle for too long. You can turn this off by setting the value to 0 or to a negative number. To set the units for the interval, use LDAP Connection Heartbeat Time Unit.

**ssoadm** attribute: `openam-idrepo-ldapv3-heartbeat-interval`

Default: 10

### LDAP Connection Heartbeat Time Unit

Time unit for the LDAP Connection Heartbeat Interval setting.

**ssoadm** attribute: `openam-idrepo-ldapv3-heartbeat-timeunit`

Default: `second`

### Maximum Results Returned from Search

A cap for the number of search results to request. For example, when using the Subjects tab to view profiles, even if you set Configuration > Console > Administration > Maximum Results Returned from Search to a larger number, OpenAM does not exceed this setting. Rather than raise this number, consider narrowing your search to match fewer directory entries.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-max-result`

Default: 1000

### Search Timeout

Maximum time to wait for search results in seconds. Does not apply to persistent searches.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-time-limit`

Default: 10

### LDAPv3 Plugin Search Scope

LDAP searches can apply to a single entry (SCOPE\_BASE), entries directly below the search DN (SCOPE\_ONE), or all entries below the search DN (SEARCH\_SUB).

**ssoadm** attribute: `sun-idrepo-ldapv3-config-search-scope`

Default: `SCOPE_SUB`

### LDAPv3 Repository Plugin Class Name

OpenAM identity repository implementation.

**ssoadm** attribute: `sunIdRepoClass`

Default: `org.forgerock.openam.idrepo.ldap.DJLDAPv3Repo`

### Attribute Name Mapping

Map of OpenAM profile attribute names to directory server attribute names.

**ssoadm** attribute: `sunIdRepoAttributeMapping`

## LDAPv3 Plugin Supported Types and Operations

Map of OpenAM operations that can be performed in the specified OpenAM contexts.

**ssoadm** attribute: `sunIdRepoSupportedOperations`

Default: `group=read,create,edit,delete, realm=read,create,edit,delete,service, user=read,create,edit,delete,service`

## LDAP Users Search Attribute

When searching for a user by name, match values against this attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-users-search-attribute`

Default: `cn`

## LDAP Users Search Filter

When searching for users, apply this LDAP search filter as well.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-users-search-filter`

Default: `(objectclass=inetorgperson)`

## LDAP People Container Naming Attribute

RDN attribute of the LDAP base DN which contains user profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-people-container-name`

Default: `ou`

## LDAP People Container Value

RDN attribute value of the LDAP base DN which contains user profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-people-container-value`

## LDAP User Object Class

User profiles have these LDAP object classes.

OpenAM handles only those attributes listed in this setting. OpenAM discards any unlisted attributes from requests and the request proceeds without the attribute.

For example, with default settings if you request that OpenAM execute a search that asks for the `mailAlternateAddress` attribute, OpenAM does the search, but does not request `mailAlternateAddress`. In the same way, OpenAM does perform an update operation with a request to set the value of an unlisted attribute like `mailAlternateAddress`, but it drops the unlisted attribute from the update request.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-user-objectclass`

Default: `devicePrintProfilesContainer, forgerock-am-dashboard-service, inetorgperson, inetuser, iplanet-am-auth-configuration-service, iplanet-am-managed-person, iplanet-am-user-service, iPlanetPreferences, organizationalperson, person, sunAMAuthAccountLockout, sunFederationManagerDataStore, sunFMSAML2NameIdentifier, sunIdentityServerLibertyPPService, top`

## LDAP User Attributes

User profiles have these LDAP attributes.

OpenAM handles only those attributes listed in this setting. OpenAM discards any unlisted attributes from requests and the request proceeds without the attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-user-attributes`

Default: `adminRole, assignedDashboard, authorityRevocationList, caCertificate, cn, devicePrintProfiles, distinguishedName, dn, employeeNumber, givenName, inetUserHTTPURL, inetUserStatus, iplanet-am-auth-configuration, iplanet-am-session-add-session-listener-on-all-sessions, iplanet-am-session-destroy-sessions, iplanet-am-session-get-valid-sessions, iplanet-am-session-max-caching-time, iplanet-am-session-max-idle-time, iplanet-am-session-max-session-time, iplanet-am-session-quota-limit, iplanet-am-session-service-status, iplanet-am-user-account-life, iplanet-am-user-admin-start-dn, iplanet-am-user-alias-list, iplanet-am-user-auth-config, iplanet-am-user-auth-modules, iplanet-am-user-failure-url, iplanet-am-user-federation-info-key, iplanet-am-user-federation-info, iplanet-am-user-login-status, iplanet-am-user-password-reset-force-reset, iplanet-am-user-password-reset-options, iplanet-am-user-password-reset-question-answer, iplanet-am-user-success-url, mail, manager, memberOf, objectClass, postalAddress, preferredLanguage, preferredLocale, preferredTimezone, sn, sun-fm-saml2-nameid-info, sun-fm-saml2-nameid-infokey, sunAMAuthInvalidAttemptsData, sunIdentityMSISDNNumber, sunIdentityServerDiscoEntries, sunIdentityServerPPAddressCard, sunIdentityServerPPCommonNameAltCN, sunIdentityServerPPCommonNameCN, sunIdentityServerPPCommonNameFN, sunIdentityServerPPCommonNameMN, sunIdentityServerPPCommonNamePT, sunIdentityServerPPCommonNameSN, sunIdentityServerPPDemographicsAge, sunIdentityServerPPDemographicsBirthDay, sunIdentityServerPPDemographicsDisplayLanguage, sunIdentityServerPPDemographicsLanguage, sunIdentityServerPPDemographicsTimeZone, sunIdentityServerPPEmergencyContact, sunIdentityServerPPEmploymentIdentityAlt0, sunIdentityServerPPEmploymentIdentityJobTitle, sunIdentityServerPPEmploymentIdentityOrg, sunIdentityServerPPEncryPTKey, sunIdentityServerPPFacadegreetmesound, sunIdentityServerPPFacadeGreetSound, sunIdentityServerPPFacadeMugShot, sunIdentityServerPPFacadeNamePronounced, sunIdentityServerPPFacadeWebSite, sunIdentityServerPPInformalName, sunIdentityServerPPLegalIdentityAltIdType, sunIdentityServerPPLegalIdentityAltIdValue, sunIdentityServerPPLegalIdentityDOB, sunIdentityServerPPLegalIdentityGender, sunIdentityServerPPLegalIdentityLegalName, sunIdentityServerPPLegalIdentityMaritalStatus, sunIdentityServerPPLegalIdentityVATIdType, sunIdentityServerPPLegalIdentityVATIdValue, sunIdentityServerPPMsgContact, sunIdentityServerPPSignKey, telephoneNumber, uid, userCertificate, userPassword`

## Create User Attribute Mapping

When creating a user profile, apply this map of OpenAM profile attribute names to directory server attribute names.



Attributes not mapped to another attribute (for example, `cn`) and attributes mapped to themselves (for example, `cn=cn`) take the value of the username unless the attribute values are provided when creating the profile. The object classes for user profile LDAP entries generally require Common Name (`cn`) and Surname (`sn`) attributes, so this prevents an LDAP constraint violation when performing the add operation.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-createuser-attr-mapping`

Default: `cn, sn`

### Attribute Name of User Status

Attribute to check/set user status.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-isactive`

Default: `inetuserstatus`

### User Status Active Value

Active users have the user status attribute set to this value.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-active`

Default: `Active`

### User Status Inactive Value

Inactive users have the user status attribute set to this value.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-inactive`

Default: `Inactive`

### Authentication Naming Attribute

RDN attribute for building the bind DN when given a username and password to authenticate a user against the directory server.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-auth-naming-attr`

Default: `cn`

### LDAP Groups Search Attribute

When searching for a group by name, match values against this attribute.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-groups-search-attribute`

Default: `cn`

### LDAP Groups Search Filter

When searching for groups, apply this LDAP search filter as well.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-groups-search-filter`

Default: `(objectclass=groupOfNames)`

### LDAP Groups Container Naming Attribute

RDN attribute of the LDAP base DN which contains group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-container-name`

Default: `ou`

### LDAP Groups Container Value

RDN attribute value of the LDAP base DN which contains group profiles.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-container-value`

### LDAP Groups Object Class

Group profiles have these LDAP object classes.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-objectclass`

Default: `groupofnames, top`

### LDAP Groups Attributes

Group profiles have these LDAP attributes.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-group-attributes`

Default: `cn, description, dn, member, objectclass, ou`

### Attribute Name for Group Membership

LDAP attribute in the member's LDAP entry whose values are the groups to which a member belongs.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-memberof`

### Attribute Name of Unique Member

Attribute in the group's LDAP entry whose values are the members of the group.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-uniquemember`

Default: `member`

### Default Group Member's User DN

DN of member added to all newly created groups.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-dftgroupmember`

## Persistent Search Base DN

Base DN for LDAP-persistent searches used to receive notification of changes in directory server data.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-psearchbase`

Default: `base-dn`

## Persistent Search Filter

LDAP filter to apply when performing persistent searches.

**ssoadm** attribute: `sun-idrepo-ldapv3-config-psearch-filter`

Default: `(objectclass=*)`

## Persistent Search Scope

LDAP searches can apply to a single entry (SCOPE\_BASE), entries directly below the search DN (SCOPE\_ONE), or all entries below the search DN (SEARCH\_SUB).

**ssoadm** attribute: `sun-idrepo-ldapv3-config-psearch-scope`

Default: `SCOPE_SUB`

## The Delay Time Between Retries

How long to wait after receiving an error result that indicates OpenAM should try the LDAP operation again.

**ssoadm** attribute: `com.iplanet.am.ldap.connection.delay.between.retries`

Default: 1000 milliseconds

## DN Cache Enabled

Whether to enable the DN cache, which is used to cache DN lookups that can happen in bursts during authentication. As the cache can become stale when a user is moved or renamed, enable DN caching when the directory service allows move/rename operations (Mod DN), and when OpenAM uses persistent searches to obtain notification of such updates.

**ssoadm** attribute: `sun-idrepo-ldapv3-dncache-enabled`

Default: true

## DN Cache Size

Maximum number of DN's cached when caching is enabled.

**ssoadm** attribute: `sun-idrepo-ldapv3-dncache-size`

Default: 1500 items

## 10.2. Realm Privileges Configuration Reference

The following table describes privileges that you can assign in the AM console or by using the **ssoadm add-privileges** command:

*Table 10.1. Privileges*

Privilege as it Appears in the AM console	Privilege Name to Use With the ssoadm add-privileges Command	Notes
Read and write access to all realm and policy properties	<b>RealmAdmin</b>	Assign this privilege to administrators in order to let them modify or read any part of an OpenAM realm. Use this privilege when you do not require granularity in your delegation model. All other OpenAM privileges are included with this privilege. Administrators using the OpenAM administration console must have this privilege.
Read and write access to all log files	<b>LogAdmin</b>	Subset of the <b>RealmAdmin</b> privilege.
Read access to all log files	<b>LogRead</b>	Subset of the <b>RealmAdmin</b> privilege.
Write access to all log files	<b>LogWrite</b>	Subset of the <b>RealmAdmin</b> privilege.
Read and write access to all configured agents	<b>AgentAdmin</b>	Provides access to centralized agent configuration; subset of the <b>RealmAdmin</b> privilege.
Read and write access to all federation metadata configurations	<b>FederationAdmin</b>	Subset of the <b>RealmAdmin</b> privilege.
REST calls for reading realms	<b>RealmReadAccess</b>	Subset of the <b>RealmAdmin</b> privilege.
Read and write access only for policy properties, including REST calls	<b>PolicyAdmin</b>	Assign this privilege to policy administrators in order to let them modify or read any part of the OpenAM policy configuration. This privilege lets an administrator modify or read all policy components: policies, applications, subject types, condition types, subject attributes, and decision combinators. All other OpenAM privileges that affect policy components are included with this privilege. Subset of the <b>RealmAdmin</b> privilege.
REST calls for policy evaluation	<b>EntitlementRestAccess</b>	Subset of the <b>RealmAdmin</b> and <b>PolicyAdmin</b> privileges.

Privilege as it Appears in the AM console	Privilege Name to Use With the ssoadm add-privileges Command	Notes
REST calls for reading policies	<code>PrivilegeRestReadAccess</code>	Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.
REST calls for managing policies	<code>PrivilegeRestAccess</code>	Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.
REST calls for reading policy applications	<code>ApplicationReadAccess</code>	Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.
REST calls for modifying policy applications	<code>ApplicationModifyAccess</code>	Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.
REST calls for modifying policy resource types	<code>ResourceTypeModifyAccess</code>	Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.
REST calls for reading policy resource types	<code>ResourceTypeReadAccess</code>	Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.
REST calls for reading policy application types	<code>ApplicationTypesReadAccess</code>	Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.
REST calls for reading environment conditions	<code>ConditionTypesReadAccess</code>	Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.
REST calls for reading subject conditions	<code>SubjectTypesReadAccess</code>	Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.
REST calls for reading decision combiners	<code>DecisionCombinersReadAccess</code>	Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.
REST calls for reading subject attributes	<code>SubjectAttributesReadAccess</code>	Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.

## 10.3. Record Control File Configuration Properties

The following properties comprise the recording control file:

### `issueID`

Type: Number

*Required.* The issue identifier—a positive integer stored internally as a Java `long` data type. A case number is a good choice for the `issueID` value.

The `issueID` is a component of the path at which recorded information is stored. See Section 9.3.3, "Retrieving Recording Information" for more information.

### `referenceID`

Type: String

*Required.* A second identifier for the recording event. Use this property to segregate multiple recording events for the same issue.

The `referenceID` is a component of the path at which recorded information is stored. See Section 9.3.3, "Retrieving Recording Information" for more information.

Note that spaces are not allowed in the `referenceID` value.

#### Description

Type: String

*Required.* A textual description of the recording event.

#### zipEnable

Type: Boolean

*Required.* Whether to compress the output directory into a zip file when recording has stopped.

#### configExport

Type: Object

*Required.* An object containing the following properties:

##### enable

Type: Boolean

*Required.* Whether to export the OpenAM configuration upon completion of the recording event. Exporting the OpenAM configuration is a best practice, because it is extremely useful to have access to the configuration when troubleshooting.

##### password

Type: String

*Required* if `enable` is `true`. A key required to import the exported configuration. The key is used the same way that the `ssoadm export-svc-cfg` command uses the `-e` argument.

##### sharePassword

Type: Boolean

*Required* if `enable` is `true`. Whether to show the `password` value in the `ssoadm start-recording`, `ssoadm get-recording-status`, and `ssoadm stop-recording` output, and in the `info.json` file, which is output during recording events, and which contains run-time properties.

#### debugLogs

Type: Object

*Required.* An object containing the following properties:

##### debugLevel

Type: String

*Required.* The debug level to set for the recording event. Set the value of `debugLevel` to `MESSAGE` to get the most troubleshooting information from your recording period. Other acceptable but less commonly used values are `ERROR` and `WARNING`.

#### `autoStop`

Type: Object

*Optional.* Contains another object used to specify an event that automatically ends a recording period. For time-based termination, specify a `time` object; for termination based on uncompressed file size, specify a `fileSize` object. If you specify both `time` and `fileSize` objects, the event that occurs first causes recording to stop.

Specifying `fileSize` and `time` objects is a best practice, because it ensures that the recorded output does not occupy a larger than expected amount of space on your file system, and that recording events end in a timely fashion.

#### `time`

Type: Object

*Optional;* must be specified in the `autoStop` object if `fileSize` is not specified. Configures a recording period to terminate recording after this amount of time.

#### `timeUnit`

Type: String

*Required.* Acceptable values are `MILLISECONDS`, `SECONDS`, `MINUTES`, `HOURS`, and `DAYS`.

#### `value`

Type: Numeric

*Required.* Values in `MILLISECONDS` are rounded down to the second. The minimum acceptable value for `autoStop` is one second.

#### `fileSize`

Type: Object

*Optional;* must be specified in the `autoStop` object if `time` is not specified. Configures a recording period to terminate after the aggregate size of uncompressed debug logs has reached this size.

#### `sizeUnit`

Type: String

*Required.* Acceptable values are `B`, `KB`, `MB`, and `GB`.

#### `value`

Type: Numeric

*Required.*

#### **threadDump**

Type: Object

*Required.* An object containing the following properties:

#### **enable**

Type: Boolean

*Required.* Whether to dump threads during the recording event. Thread dumps are especially useful when troubleshooting performance issues and issues with unresponsive servers.

#### **delay**

Type: Object

*Required* if **enable** is **true**. Contains another object used to specify an interval at which thread dumps are taken. The initial thread dump is taken at the start of the recording event; subsequent thread dumps are taken at multiples of the **delay** interval.

#### **timeUnit**

Type: String

*Required.* Acceptable values are **MILLISECONDS**, **SECONDS**, **MINUTES**, **HOURS**, and **DAYS**.

#### **value**

Type: Numeric

*Required.* The minimum acceptable value is one second. Time units that are smaller than seconds, such as **MILLISECONDS**, are rounded to the closest second.

## 10.4. Audit Logging File Format

OpenAM writes log messages generated from audit events triggered by its components, instances, and other ForgeRock-based stack products.

### 10.4.1. Audit Log Format

This section presents the audit log format for each topic-based file, event names, and audit constants used in its log messages.



## 10.4.1.1. Access Log Format

Table 10.2. Access Log Format

Schema Property	Description
<code>_id</code>	Specifies a universally unique identifier (UUID) for the message object, such as <code>a568d4fe-d655-49a8-8290-bfc02095bec9-491</code> .
<code>timestamp</code>	Specifies the timestamp when OpenAM logged the message, in UTC format to millisecond precision: <code>yyyy-MM-ddTHH:mm:ss.msZ</code> . For example: <code>2015-11-14T00:16:04.653Z</code>
<code>eventName</code>	Specifies the name of the audit event. For example, <code>AM-ACCESS-ATTEMPT</code> and <code>AM-ACCESS-OUTCOME</code> .
<code>transactionId</code>	<p>Specifies the UUID of the transaction, which identifies an external request when it comes into the system boundary. Any events generated while handling that request will be assigned that transaction ID, so that you may see the same transaction ID even for different audit event topics. For example, <code>9c9e8d5c-2941-4e61-9c3c-8a990088e801</code>.</p> <p>OpenAM supports a feature where trusted OpenAM deployment with multiple instances, components, and ForgeRock stack products can propagate the transaction ID through each call across the stack. OpenAM reads the <code>X-ForgeRock-TransactionId</code> HTTP header and appends an integer to the transaction ID. Note that this feature is disabled by default. When enabled, this feature should filter the <code>X-ForgeRock-TransactionId</code> HTTP header for connections from untrusted sources.</p>
<code>user.id</code>	Specifies the universal identifier for authenticated users. For example, <code>id=scarter,ou=user,o=shop,ou=services,dc=example,dc=com</code> .
<code>trackingIds</code>	<p>Specifies a unique random string generated as an alias for each OpenAM session ID and OAuth 2.0 token. In releases prior to OpenAM 13.0.0, the <code>contextId</code> log property used a random string as an alias for the session ID. The <code>trackingIds</code> property also uses an alias when referring to session IDs, for example, <code>[ "45b17894529cf74301" ]</code>.</p> <p>OpenAM 13.0.0 extends this property to handle OAuth 2.0 tokens. In this case, whenever OpenAM generates an access or grant token, it also generates unique random value and logs it as an alias. In this way, it is possible to trace back an access token back to its originating grant token, trace the grant token back to the session in which it was created, and then trace how the session was authenticated. An example of a <code>trackingIds</code> property in an OAuth 2.0/ OpenID Connect 1.0 environment is: <code>[ "1979edf68543ead001", "8878e51a-f2aa-464f-b1cc-b12fd6daa415", "3df9a5c3-8d1e-4ee3-93d6-b9bbe58163bc" ]</code></p>
<code>server.ip</code>	Specifies the IP address of the OpenAM server. For example, <code>127.0.0.1</code> .
<code>server.port</code>	Specifies the port number used by the OpenAM server. For example, <code>8080</code> .
<code>client.host</code>	Specifies the client hostname. This field is only populated if reverse DNS lookup is enabled.
<code>client.ip</code>	Specifies the client IP address.
<code>client.port</code>	Specifies the client port number.

Schema Property	Description
<code>authorizationId.roles</code>	Specifies the list of roles for the authorized user.
<code>authorizationId.component</code>	Specifies the component part of the authorized ID, such as
<code>request.protocol</code>	Specifies the protocol associated with the request operation. Possible values: <b>CREST</b> and <b>PLL</b> .
<code>request.operation</code>	Specifies the request operation. For CREST operations, possible values: <b>READ</b> , <b>ACTION</b> , <b>QUERY</b> . For PLL operations, possible values: <b>LoginIndex</b> , <b>SubmitRequirements</b> , <b>GetSession</b> , <b>REQUEST_ADD_POLICY_LISTENER</b> .
<code>request.detail</code>	Specifies the detailed information about the request operation. For example, <code>{"action": "idFromSession"}, {"action": "validateGoto"}, {"action": "validate"}, {"action": "logout"}, {"action": "schema"}, {"action": "template"}</code> .
<code>http.method</code>	Specifies the HTTP method requested by the client. For example, <b>GET</b> , <b>POST</b> , <b>PUT</b> .
<code>http.path</code>	Specifies the path of the HTTP request. For example, <code>http://forgerock-am0.int.openrock.org:8080/openam/json/realms/root/authenticate</code> .
<code>http.queryParameters</code>	Specifies the HTTP query parameter string. For example, <code>{ "_action": [ "idFromSession" ] }, { "_queryFilter": [ "true" ] }, { "_action": [ "validate" ] }, { "_action": [ "logout" ] }, { "realm": [ "/shop" ] }, { "_action": [ "validateGoto" ] }</code> .
<code>http.headers</code>	Specifies the HTTP header for the request. For example, (Note: Line feeds added for readability purposes): <pre>{ "accept": [ "application/json, text/javascript, */*; q=0.01" ], "Accept- -API-Version": [ "protocol=1.0" ], "accept-encoding": [ "gzip, deflate" ] , "accept-language": [ "en-US;q=1,en;q=0.9" ], "cache-control": [ "no- cache" ], "connection": [ "Keep-Alive" ], "content-length": [ "0" ], "host": [ "forgerock-am.openrock.org" ], "pragma": [ "no-cache" ], "referer": [ "https://forgerock-am.openrock.org/openam/XUI/" ], "user-agent": [ "Mozilla /5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0" ], "x- nosession": [ "true" ], "x-requested-with": [ "XMLHttpRequest" ], "x- username": [ "anonymous" ] }</pre>
<code>http.request.cookies</code>	Specifies a JSON map of key-value pairs and appears as its own property to allow for blacklisting fields or values.
<code>http.response.cookies</code>	Not used in OpenAM.
<code>response.status</code>	Specifies the response status of the request. Normally, <b>SUCCESS</b> , <b>FAILURE</b> , or null.
<code>response.statusCode</code>	Specifies the response status code, depending on the protocol. For CREST, HTTP failure codes are displayed but not HTTP success codes. For PLL endpoints, PLL error codes are displayed.
<code>response.detail</code>	Specifies the message associated with <code>response.statusCode</code> . For example, the <code>response.statusCode</code> of <b>401</b> has a <code>response.detail</code> of <code>{ "reason": "Unauthorized" }</code> .
<code>response.elapsedTime</code>	Specifies the time to execute the access event, usually in millisecond precision.
<code>response.elapsedTimeUnits</code>	Specifies the elapsed time units of the response. For example, <b>MILLISECONDS</b> .

Schema Property	Description
<code>component</code>	Specifies the OpenAM service utilized. For example, <code>Server Info</code> , <code>Users</code> , <code>Config</code> , <code>Session</code> , <code>Authentication</code> , <code>Policy</code> , <code>OAuth</code> .
<code>realm</code>	Specifies the realm where the operation occurred. For example, the Top Level Realm (" <code>/</code> ") or the sub-realm name (" <code>/shop</code> ").

### 10.4.1.2. Activity Log Format

*Table 10.3. Activity Log Format*

Property	Description
<code>_id</code>	Specifies a universally unique identifier (UUID) for the message object, such as <code>a568d4fe-d655-49a8-8290-bfc02095bec9-487</code> .
<code>timestamp</code>	Specifies the timestamp when OpenAM logged the message, in UTC format to millisecond precision: <code>yyyy-MM-ddTHH:mm:ss.msZ</code> . For example: <code>2015-11-14T00:16:04.652Z</code>
<code>eventName</code>	Specifies the name of the audit event. For example, <code>AM-SESSION_CREATED</code> , <code>AM-SESSION-LOGGED_OUT</code> .
<code>transactionId</code>	Specifies the UUID of the transaction, which identifies an external request when it comes into the system boundary. Any events generated while handling that request will be assigned that transaction ID, so that you may see the same transaction ID for same even for different audit event topics. For example, <code>9c9e8d5c-2941-4e61-9c3c-8a990088e801</code> .
<code>user.id</code>	Specifies the universal identifier for authenticated users. For example, <code>id=scarter,ou=user,o=shop,ou=services,dc=example,dc=com</code> .
<code>trackingIds</code>	Specifies an array containing a random context ID that identifies the session and a random string generated from an OAuth 2.0/OpenID Connect 1.0 flow that could track an access token ID or an grant token ID. For example, <code>[ "45b17894529cf74301" ]</code> .
<code>runAs</code>	Specifies the user to run the activity as. May be used in delegated administration. For example, <code>id=dsameuser,ou=user,dc=example,dc=com</code> .
<code>objectId</code>	Specifies the identifier of an object that has been created, updated, or deleted. For OpenAM 13.0.0, only session changes are recorded, so that the session <code>trackingId</code> is used in this field. For example, <code>[ "45b17894529cf74301" ]</code>
<code>operation</code>	Specifies the state change operation invoked: <code>CREATE</code> , <code>MODIFY</code> , or <code>DELETE</code> .
<code>before</code>	Not used.
<code>after</code>	Not used.
<code>changedFields</code>	Not used.
<code>revision</code>	Not used.
<code>component</code>	Specifies the OpenAM service utilized. Normally, <code>SESSION</code> .

Property	Description
realm	Specifies the realm where the operation occurred. For example, the Top Level Realm ("/") or the sub-realm name ("/shop").

### 10.4.1.3. Authentication Log Format

Table 10.4. Authentication Log Format

Property	Description
_id	Specifies a universally unique identifier (UUID) for the message object, such as a568d4fe-d655-49a8-8290-bfc02095bec9-485.
timestamp	Specifies the timestamp when OpenAM logged the message, in UTC format to millisecond precision: yyyy-MM-ddTHH:mm:ss.msZ. For example: 2015-11-14T00:16:04.640Z
eventName	Specifies the name of the audit event. For example, AM-LOGOUT, AM-LOGIN-MODULE-COMPLETED, AM-LOGIN-CHAIN-COMPLETED.
transactionId	Specifies the UUID of the transaction, which identifies an external request when it comes into the system boundary. Any events generated while handling that request will be assigned that transaction ID, so that you may see the same transaction ID for same even for different audit event topics. For example, 9c9e8d5c-2941-4e61-9c3c-8a990088e801.
user.id	Specifies the universal identifier for authenticated users. For example, id=scarter,ou=user,o=shop,ou=services,dc=example,dc=com.
trackingIds	Specifies an array containing a random context ID that identifies the session and a random string generated from an OAuth 2.0/OpenID Connect 1.0 flow that could track an access token ID or an grant token ID. For example, [ "45b17894529cf74301" ].
result	Specifies the outcome of a single authentication module within a chain, either SUCCESSFUL or FAILED.
principal	Specifies the array of accounts used to authenticate, such as [ "amadmin" ], [ "scarter" ].
context	Not used
entries	Specifies the JSON representation of the details of an authentication module or chain. OpenAM creates an event as each module completes and a final event at the end of the chain. For example, [ { "moduleId": "DataStore", "info": { "moduleClass": "DataStore", "ipAddress": "127.0.0.1", "moduleName": "DataStore", "authLevel": "0" } } ]
component	Specifies the OpenAM service utilized. Normally, Authentication.
realm	Specifies the realm where the operation occurred. For example, the Top Level Realm ("/") or the sub-realm name ("/shop").

## 10.4.1.4. Config Log Format

Table 10.5. Config Log Format

Property	Description
<code>_id</code>	Specifies a universally unique identifier (UUID) for the message object. For example, <code>6a568d4fe-d655-49a8-8290-bfc02095bec9-843</code> .
<code>timestamp</code>	Specifies the timestamp when OpenAM logged the message, in UTC format to millisecond precision: <code>yyyy-MM-ddTHH:mm:ss.msZ</code> . For example, <code>2015-11-14T00:21:03.490Z</code>
<code>eventName</code>	Specifies the name of the audit event. For example, <code>AM-CONFIG-CHANGE</code> .
<code>transactionId</code>	Specifies the UUID of the transaction, which identifies an external request when it comes into the system boundary. Any events generated while handling that request will be assigned that transaction ID, so that you may see the same transaction ID for different audit event topics. For example, <code>301d1a6e-67f9-4e45-bfeb-5e4047a8b432</code> .
<code>user.id</code>	Not used. You can determine the value for this field by linking to the access event using the same <code>transactionId</code> .
<code>trackingIds</code>	Not used.
<code>runAs</code>	Specifies the user to run the activity as. May be used in delegated administration. For example, <code>id=amadmin,ou=user,dc=example,dc=com</code> .
<code>objectId</code>	Specifies the identifier of a system object that has been created, modified, or deleted. For example, <code>ou=SamuelTwo,ou=default,ou=OrganizationConfig,ou=1.0,ou=iPlanetAMAuthSAML2Service,ou=services,o=shop,ou=services,dc=example,dc=com</code> .
<code>operation</code>	Specifies the state change operation invoked: <code>CREATE</code> , <code>MODIFY</code> , or <code>DELETE</code> .
<code>before</code>	Specifies the JSON representation of the object prior to the activity. For example, <pre>{ "sunsmpriority":["0"], "objectclass":["top","sunServiceComponent","organizationalUnit"], "ou":["SamuelTwo"],"sunserviceID":["serverconfig"] }</pre>
<code>after</code>	Specifies the JSON representation of the object after the activity. For example, <pre>{ "sunKeyValue":["forgerock-am-auth-saml2-auth-level=0","forgerock-am-auth-saml2-meta-alias=/sp","forgerock-am-auth-saml2-entity-name=http://","forgerock-am-auth-saml2-authn-context-decl-ref=","forgerock-am-auth-saml2-force-authn=none","forgerock-am-auth-saml2-is-passive=none","forgerock-am-auth-saml2-login-chain=","forgerock-am-auth-saml2-auth-comparison=none","forgerock-am-auth-saml2-req-binding=urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect","forgerock-am-auth-saml2-binding=urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact","forgerock-am-auth-saml2-authn-context-class-ref=","forgerock-am-auth-saml2-slo-relay=http://","forgerock-am-auth-saml2-allow-create=false","forgerock-am-auth-saml2-name-id-format=urn:oasis:names:tc:SAML:2.0:nameid-format:persistent","</pre>
<code>changedFields</code>	Specifies the fields that were changed. For example, <code>[ "sunKeyValue" ]</code> .
<code>revision</code>	Not used.

Property	Description
component	Not used.
realm	Specifies the realm where the operation occurred. For example, the Top Level Realm ("/") or the sub-realm name ("/shop").

## 10.4.2. Audit Log Event Names

The following section presents the predefined names for the audit events:

*Table 10.6. Audit Log Event Names*

Topic	EventName
access	AM-ACCESS-ATTEMPT
access	AM-ACCESS-OUTCOME
activity	AM-SESSION-CREATED
activity	AM-SESSION-IDLE_TIME_OUT
activity	AM-SESSION-MAX_TIMED_OUT
activity	AM-SESSION-LOGGED_OUT
activity	AM-SESSION-DESTROYED
activity	AM-SESSION-PROPERTY_CHANGED
access	AM-LOGIN-MODULE-COMPLETED
access	AM-LOGIN-COMPLETED
access	AM-LOGOUT
config	AM-CONFIG-CHANGE

## 10.4.3. Audit Log Components

The following section presents the predefined audit event components that make up the log messages:

*Table 10.7. Audit Log Event Components*

Event Component	
OAuth	OAuth 2.0, OpenID Connect 1.0, and UMA
CTS	Core Token Service
Policy Agent	Web and Java EE policy agents
Authentication	Authentication service
Dashboard	Dashboard service

Event Component	
Server Info	Server information service
Users	Users component
Groups	Groups component
Oath	Mobile authentication
Devices	Trusted devices
Policy	Policies
Realms	Realms and sub-realms
Session	Session service
Script	Scripting service
Batch	Batch service
Config	Configuration
STS	Secure Token Service: REST and SOAP
Record	Recording service
Audit	Auditing service
Radius	RADIUS server

#### 10.4.4. Audit Log Failure Reasons

The following section presents the predefined audit event failure reasons:

*Table 10.8. Audit Log Event Authentication Failure Reasons*

Failure	Description
LOGIN_FAILED	Incorrect/invalid credentials presented.
INVALID_PASSWORD	Invalid credentials entered.
NO_CONFIG	Authentication chain does not exist.
NO_USER_PROFILE	No user profile found for this user.
USER_INACTIVE	User is not active.
LOCKED_OUT	Maximum number of failure attempts exceeded. User is locked out.
ACCOUNT_EXPIRED	User account has expired.
LOGIN_TIMEOUT	Login timed out.
MODULE_DENIED	Authentication module is denied.
MAX_SESSION_REACHED	Limit for maximum number of allowed sessions has been reached.
INVALID_REALM	Realm does not exist.
REALM_INACTIVE	Realm is not active.

Failure	Description
USER_NOTE_FOUND	Role-based authentication: user does not belong to this role.
AUTH_TYPE_DENIED	Authentication type is denied.
SESSION_CREATE_ERROR	Cannot create a session.
INVALID_LEVEL	Level-based authentication: Invalid authentication level.

## 10.5. Audit Logging

**ssoadm** service name: `audit`

### 10.5.1. Global Attributes

The following settings appear on the **Global Attributes** tab:

#### Audit logging

Enable audit logging in OpenAM.

Default value: `true`

**ssoadm** attribute: `auditEnabled`

#### Field exclusion policies

A list of fields or values (JSON pointers) to exclude from the audit event.

To specify a field or value within a field to be filtered out of the event, start the pointer with the event topic, for example access, activity, authentication, or config, followed by the field name or the path to the value in the field.

For example, to filter out the `userId` field in an access event the pointer will be `/access/userId`.

To filter out the `content-type` value in the `http.request.headers` field the pointer will be `/access/http/request/headers/content-type`.

Only values that are made up of JSON strings can be manipulated in this way.

Default value:

```
/access/http/request/queryParameters/tokenId
/access/http/request/headers/cache-control
/access/http/request/queryParameters/redirect_uri
/access/http/request/queryParameters/Login.Token1
/access/http/request/headers/accept-language
/config/before
/access/http/request/headers/%AM_AUTH_COOKIE_NAME%
/config/after
/access/http/request/queryParameters/access_token
/access/http/request/headers/X-OpenAM-Password
```



```
/access/http/request/queryParameters/id_token_hint
/access/http/request/headers/proxy-authorization
/access/http/request/queryParameters/IDToken1
/access/http/request/queryParameters/requester
/access/http/request/headers/connection
/access/http/request/queryParameters/sessionUpgradeSSOTokenId
/access/http/request/headers/content-type
/access/http/request/cookies/%AM_COOKIE_NAME%
/access/http/request/headers/accept-encoding
/access/http/request/headers/authorization
/access/http/request/headers/content-length
/access/http/request/headers/%AM_COOKIE_NAME%
```

**ssoadm** attribute: `fieldFilterPolicy`

## 10.5.2. Realm Defaults

The following settings appear on the *Realm Defaults* tab:

### Audit logging

Enable audit logging in OpenAM.

Default value: `true`

**ssoadm** attribute: `auditEnabled`

### Field exclusion policies

A list of fields or values (JSON pointers) to exclude from the audit event.

To specify a field or value within a field to be filtered out of the event, start the pointer with the event topic, for example `access`, `activity`, `authentication`, or `config`, followed by the field name or the path to the value in the field.

For example, to filter out the `userId` field in an access event the pointer will be `/access/userId`.

To filter out the `content-type` value in the `http.request.headers` field the pointer will be `/access/http/request/headers/content-type`.

Only values that are made up of JSON strings can be manipulated in this way.

Default value:

```
/access/http/request/queryParameters/tokenId
/access/http/request/headers/cache-control
/access/http/request/queryParameters/redirect_uri
/access/http/request/queryParameters/Login.Token1
/access/http/request/headers/accept-language
/config/before
/access/http/request/headers/%AM_AUTH_COOKIE_NAME%
/config/after
/access/http/request/queryParameters/access_token
/access/http/request/headers/X-OpenAM-Password
```

```
/access/http/request/queryParameters/id_token_hint  
/access/http/request/headers/proxy-authorization  
/access/http/request/queryParameters/IDToken1  
/access/http/request/queryParameters/requester  
/access/http/request/headers/connection  
/access/http/request/queryParameters/sessionUpgradeSS0TokenId  
/access/http/request/headers/content-type  
/access/http/request/cookies/%AM_COOKIE_NAME%  
/access/http/request/headers/accept-encoding  
/access/http/request/headers/authorization  
/access/http/request/headers/content-length  
/access/http/request/headers/%AM_COOKIE_NAME%
```

**ssoadm** attribute: `fieldFilterPolicy`

### 10.5.3. Secondary Configurations

This service has the following Secondary Configurations.

#### 10.5.3.1. JMS

A configured secondary instance of the JMS type has the following tabs:

##### General Handler Configuration

The General Handler Configuration tab contains the following secondary configuration properties:

##### Enabled

Enables or disables an audit event handler.

Default value: `true`

**ssoadm** attribute: `enabled`

##### Topics

List of topics handled by an audit event handler.

Default value:

```
access  
activity  
config  
authentication
```

**ssoadm** attribute: `topics`

##### Audit Event Handler Factory

The Audit Event Handler Factory tab contains the following secondary configuration properties:

## Factory Class Name

The fully qualified class name of the factory responsible for creating the Audit Event Handler. The class must implement `org.forgerock.openam.audit.AuditEventHandlerFactory`.

Default value: `org.forgerock.openam.audit.events.handlers.JmsAuditEventHandlerFactory`

**ssoadm** attribute: `handlerFactory`

## JMS Configuration

The JMS Configuration tab contains the following secondary configuration properties:

### Delivery Mode

Specifies whether JMS messages used to transmit audit events use persistent or non-persistent delivery.

With persistent delivery, the JMS provider ensures that messages are not lost in transit in case of a provider failure by logging messages to storage when they are sent.

Specify the delivery mode as persistent if it is unacceptable for delivery of audit events to be lost in JMS transit. If the possible loss of audit events is acceptable, choose non-persistent delivery, which provides better performance.

Default value: `NON_PERSISTENT`

**ssoadm** attribute: `deliveryMode`

### Session Mode

Specifies the JMS session acknowledgement mode: `AUTO`, `CLIENT`, or `DUPS_OK`.

- Auto mode guarantees once-only delivery of JMS messages used to transmit audit events.
- Duplicates OK mode ensures that messages are delivered at least once.
- Client mode does not ensure delivery.

Use the default setting unless your JMS broker implementation requires otherwise. See your broker documentation for more information.

Default value: `AUTO`

**ssoadm** attribute: `sessionMode`

### JNDI Context Properties

Specifies JNDI properties that OpenAM uses to connect to the JMS message broker to which OpenAM will publish audit events.

OpenAM acts as a JMS client, using a JMS connection factory to connect to your JMS message broker. In order for OpenAM to connect to the broker, the JNDI context properties must conform

to those needed by the broker. See the documentation for your JMS message broker for required values.

The default properties are example properties for connecting to Apache ActiveMQ.

Default value:

```
topic.audit=audit
java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory
java.naming.provider.url=tcp://localhost:61616
```

**ssoadm** attribute: `jndiContextProperties`

### JMS Topic Name

JNDI lookup name for the JMS topic

Default value: `audit`

**ssoadm** attribute: `jndiTopicName`

### JMS Connection Factory Name

Specifies the JNDI lookup name for the connection factory exposed by your JMS message broker. OpenAM performs a JNDI lookup on this name to locate your broker's connection factory.

See the documentation for your JMS message broker for the required value.

The default is the connection factory name for Apache ActiveMQ.

Default value: `ConnectionFactory`

**ssoadm** attribute: `jndiConnectionFactoryName`

## Batch Events

The Batch Events tab contains the following secondary configuration properties:

### Batch enabled

Boolean for batch delivery of audit events.

Default value: `true`

**ssoadm** attribute: `batchEnabled`

### Capacity

Maximum event count in the batch queue; additional events are dropped.

Default value: `1000`

**ssoadm** attribute: `batchCapacity`

### Max Batched

Maximum number of events per batch.

Default value: 100

**ssoadm** attribute: `maxBatchedEvents`

### Thread Count

Number of concurrent threads that pull events from the batch queue.

Default value: 3

**ssoadm** attribute: `batchThreadCount`

### Insert Timeout

Waiting period (seconds) for available capacity, when a new event enters the queue.

Default value: 60

**ssoadm** attribute: `insertTimeoutSec`

### Polling Timeout

Worker thread waiting period (seconds) for the next event, before going idle.

Default value: 10

**ssoadm** attribute: `pollTimeoutSec`

### Shutdown Timeout

Application waiting period (seconds) for worker thread termination.

Default value: 60

**ssoadm** attribute: `shutdownTimeoutSec`

## 10.5.3.2. Elasticsearch

A configured secondary instance of the Elasticsearch type has the following tabs:

### General Handler Configuration

The General Handler Configuration tab contains the following secondary configuration properties:

#### Enabled

Enables or disables an audit event handler.

Default value: `true`

**ssoadm** attribute: `enabled`

## Topics

List of topics handled by an audit event handler.

Default value:

```
access
activity
config
authentication
```

**ssoadm** attribute: `topics`

## Audit Event Handler Factory

The Audit Event Handler Factory tab contains the following secondary configuration properties:

### Factory Class Name

The fully qualified class name of the factory responsible for creating the Audit Event Handler. The class must implement `org.forgerock.openam.audit.AuditEventHandlerFactory`.

Default value: `org.forgerock.openam.audit.events.handlers.ElasticsearchAuditEventHandlerFactory`

**ssoadm** attribute: `handlerFactory`

## Elasticsearch Configuration

The Elasticsearch Configuration tab contains the following secondary configuration properties:

### Server Hostname

Host name or IP address of the Elasticsearch server.

**ssoadm** attribute: `host`

### Server Port

Specifies the port number used to access Elasticsearch's REST API.

**ssoadm** attribute: `port`

### SSL Enabled

Specifies whether SSL is configured on the Elasticsearch server.

If SSL is enabled, be sure to import the CA certificate used to sign Elasticsearch node certificates into the Java keystore on the host that runs OpenAM before attempting to log audit events to Elasticsearch.

Default value: `false`

**ssoadm** attribute: `sslEnabled`

### Elasticsearch Index

Specifies the name of the Elasticsearch index to be used for OpenAM audit logging.

**ssoadm** attribute: `index`

## Authentication

The Authentication tab contains the following secondary configuration properties:

### Username

Specifies the username to access the Elasticsearch server.

Required if Elasticsearch Shield authentication is configured.

**ssoadm** attribute: `username`

### Password

Specifies the password to access the Elasticsearch server.

Required if Elasticsearch Shield authentication is configured.

**ssoadm** attribute: `password`

## Buffering

The Buffering tab contains the following secondary configuration properties:

### Buffering Enabled

Default value: `true`

**ssoadm** attribute: `bufferingEnabled`

### Batch Size

Specifies the number of audit log events to hold in the buffer before writing them to Elasticsearch.

Default value: `500`

**ssoadm** attribute: `batchSize`

### Queue Capacity

Maximum number of audit logs in the batch queue. Additional audit events are dropped.

Default value: 10000

**ssoadm** attribute: maxEvents

### Write interval (in milliseconds)

Specifies the interval in milliseconds at which buffered events are written to Elasticsearch.

Default value: 250

**ssoadm** attribute: writeInterval

## 10.5.3.3. Syslog

A configured secondary instance of the Syslog type has the following tabs:

### General Handler Configuration

The General Handler Configuration tab contains the following secondary configuration properties:

#### Enabled

Enables or disables an audit event handler.

Default value: true

**ssoadm** attribute: enabled

#### Topics

List of topics handled by an audit event handler.

Default value:

```
access
activity
config
authentication
```

**ssoadm** attribute: topics

### Audit Event Handler Factory

The Audit Event Handler Factory tab contains the following secondary configuration properties:

#### Factory Class Name

The fully qualified class name of the factory responsible for creating the Audit Event Handler. The class must implement `org.forgerock.openam.audit.AuditEventHandlerFactory`.



Default value: `org.forgerock.openam.audit.events.handlers.SyslogAuditEventHandlerFactory`

**ssoadm** attribute: `handlerFactory`

## Syslog Configuration

The Syslog Configuration tab contains the following secondary configuration properties:

### Server hostname

Host name or IP address of receiving syslog server.

**ssoadm** attribute: `host`

### Server port

Port number of receiving syslog server.

**ssoadm** attribute: `port`

### Transport Protocol

Default value: `TCP`

**ssoadm** attribute: `transportProtocol`

### Connection timeout

Timeout for connecting to syslog server, in seconds.

**ssoadm** attribute: `connectTimeout`

### Facility

Syslog facility value to apply to all events.

Default value: `USER`

**ssoadm** attribute: `facility`

## Buffering

The Buffering tab contains the following secondary configuration properties:

### Buffering Enabled

Enables or disables audit event buffering.

Default value: `true`

**ssoadm** attribute: `bufferingEnabled`

### 10.5.3.4. CSV

A configured secondary instance of the CSV type has the following tabs:

#### General Handler Configuration

The General Handler Configuration tab contains the following secondary configuration properties:

##### Enabled

Enables or disables an audit event handler.

Default value: `true`

**ssoadm** attribute: `enabled`

##### Topics

List of topics handled by an audit event handler.

Default value:

```
access
activity
config
authentication
```

**ssoadm** attribute: `topics`

#### Audit Event Handler Factory

The Audit Event Handler Factory tab contains the following secondary configuration properties:

##### Factory Class Name

The fully qualified class name of the factory responsible for creating the Audit Event Handler. The class must implement `org.forgerock.openam.audit.AuditEventHandlerFactory`.

Default value: `org.forgerock.openam.audit.events.handlers.CsvAuditEventHandlerFactory`

**ssoadm** attribute: `handlerFactory`

#### CSV Configuration

The CSV Configuration tab contains the following secondary configuration properties:

##### Log Directory

Directory in which to store audit log CSV files.

Default value: `%BASE_DIR%/%SERVER_URI%/log/`

**ssoadm** attribute: `location`

## File Rotation

The File Rotation tab contains the following secondary configuration properties:

### Rotation Enabled

Enables and disables audit file rotation.

Default value: `true`

**ssoadm** attribute: `rotationEnabled`

### Maximum File Size

Maximum size, in bytes, which an audit file can grow to before rotation is triggered. A negative or zero value indicates this policy is disabled.

Default value: `100000000`

**ssoadm** attribute: `rotationMaxFileSize`

### File Rotation Prefix

Prefix to prepend to audit files when rotating audit files.

**ssoadm** attribute: `rotationFilePrefix`

### File Rotation Suffix

Suffix to append to audit files when they are rotated. Suffix should be a timestamp.

Default value: `-yyyy.MM.dd-HH.mm.ss`

**ssoadm** attribute: `rotationFileSuffix`

### Rotation Interval

Interval to trigger audit file rotations, in seconds. A negative or zero value disables this feature.

Default value: `-1`

**ssoadm** attribute: `rotationInterval`

### Rotation Times

Durations after midnight to trigger file rotation, in seconds.

**ssoadm** attribute: `rotationTimes`

## File Retention

The File Retention tab contains the following secondary configuration properties:

### Maximum Number of Historical Files

Maximum number of backup audit files allowed. A value of `-1` disables pruning of old history files.

Default value: `1`

**ssoadm** attribute: `retentionMaxNumberOfHistoryFiles`

### Maximum Disk Space

The maximum amount of disk space the audit files can occupy, in bytes. A negative or zero value indicates this policy is disabled.

Default value: `-1`

**ssoadm** attribute: `retentionMaxDiskSpaceToUse`

### Minimum Free Space Required

Minimum amount of disk space required, in bytes, on the system where audit files are stored. A negative or zero value indicates this policy is disabled.

Default value: `-1`

**ssoadm** attribute: `retentionMinFreeSpaceRequired`

## Buffering

The Buffering tab contains the following secondary configuration properties:

### Buffering Enabled

Enables or disables buffering.

Default value: `true`

**ssoadm** attribute: `bufferingEnabled`

### Flush Each Event Immediately

Performance may be improved by writing all buffered events before flushing.

Default value: `false`

**ssoadm** attribute: `bufferingAutoFlush`

## Tamper Evident Configuration

The Tamper Evident Configuration tab contains the following secondary configuration properties:

### Is Enabled

Enables the CSV tamper evident feature.

Default value: `false`

**ssoadm** attribute: `securityEnabled`

### Certificate Store Location

Path to Java keystore.

Default value: `%BASE_DIR%/%SERVER_URI%/Logger.jks`

**ssoadm** attribute: `securityFilename`

### Certificate Store Password

Password for Java keystore.

**ssoadm** attribute: `securityPassword`

### Signature Interval

Signature generation interval, in seconds.

Default value: `900`

**ssoadm** attribute: `securitySignatureInterval`

## 10.5.3.5. JDBC

A configured secondary instance of the JDBC type has the following tabs:

### General Handler Configuration

The General Handler Configuration tab contains the following secondary configuration properties:

#### Enabled

Enables or disables an audit event handler.

Default value: `true`

**ssoadm** attribute: `enabled`

#### Topics

List of topics handled by an audit event handler.

Default value:

```
access
activity
config
authentication
```

**ssoadm** attribute: `topics`

## Audit Event Handler Factory

The Audit Event Handler Factory tab contains the following secondary configuration properties:

### Factory Class Name

The fully qualified class name of the factory responsible for creating the Audit Event Handler. The class must implement `org.forgerock.openam.audit.AuditEventHandlerFactory`.

Default value: `org.forgerock.openam.audit.events.handlers.JdbcAuditEventHandlerFactory`

**ssoadm** attribute: `handlerFactory`

## Database Configuration

The Database Configuration tab contains the following secondary configuration properties:

### Database Type

Select the database to use for logging audit events.

Identifies the database in use, for example MySQL, Oracle, or SQL.

Default value: `oracle`

**ssoadm** attribute: `databaseType`

### JDBC Database URL

URL of the JDBC database.

**ssoadm** attribute: `jdbcUrl`

### JDBC Driver

Fully qualified JDBC driver class name.

**ssoadm** attribute: `driverClassName`

### Database Username

Specifies the username to access the database server.

**ssoadm** attribute: `username`

### Database Password

Specifies the password to access the database server.

**ssoadm** attribute: `password`

### Connection Timeout (seconds)

Specifies the maximum wait time before failing the connection, in seconds.

Default value: 30

**ssoadm** attribute: `connectionTimeout`

### Maximum Connection Idle Timeout (seconds)

Specifies the maximum idle time before the connection is closed, in seconds.

Default value: 600

**ssoadm** attribute: `idleTimeout`

### Maximum Connection Time (seconds)

Specifies the maximum time a JDBC connection can be open, in seconds.

Default value: 1800

**ssoadm** attribute: `maxLifetime`

### Minimum Idle Connections

Specifies the minimum number of idle connections in the connection pool.

Default value: 10

**ssoadm** attribute: `minIdle`

### Maximum Connections

Specifies the maximum number of connections in the connection pool.

Default value: 10

**ssoadm** attribute: `maxPoolSize`

## Buffering

The Buffering tab contains the following secondary configuration properties:

### Buffering Enabled

Enables or disables audit event buffering.

Default value: `true`

**ssoadm** attribute: `bufferingEnabled`

### Buffer Size (number of events)

Size of the queue where events are buffered before they are written to the database.

This queue has to be big enough to store all incoming events that have not yet been written to the database.

If the queue reaches capacity, the process will block until a write occurs.

Default value: 100000

**ssoadm** attribute: `bufferingMaxSize`

### Write Interval

Specifies the interval (seconds) at which buffered events are written to the database.

Default value: 5

**ssoadm** attribute: `bufferingWriteInterval`

### Writer Threads

Specifies the number of threads used to write the buffered events.

Default value: 1

**ssoadm** attribute: `bufferingWriterThreads`

### Max Batched Events

Specifies the maximum number of batched statements the database can support per connection.

Default value: 100

**ssoadm** attribute: `bufferingMaxBatchedEvents`

## 10.5.3.6. JSON

A configured secondary instance of the JSON type has the following tabs:

### General Handler Configuration

The General Handler Configuration tab contains the following secondary configuration properties:

#### Enabled

Enables or disables an audit event handler.

Default value: `true`

**ssoadm** attribute: `enabled`

#### Topics

List of topics handled by an audit event handler.

Default value:



```
access
activity
config
authentication
```

**ssoadm** attribute: `topics`

## Audit Event Handler Factory

The Audit Event Handler Factory tab contains the following secondary configuration properties:

### Factory Class Name

The fully qualified class name of the factory responsible for creating the Audit Event Handler. The class must implement `org.forgerock.openam.audit.AuditEventHandlerFactory`.

Default value: `org.forgerock.openam.audit.events.handlers.JsonAuditEventHandlerFactory`

**ssoadm** attribute: `handlerFactory`

## JSON Configuration

The JSON Configuration tab contains the following secondary configuration properties:

### Log Directory

Directory in which to store audit log JSON files.

Default value: `%BASE_DIR%/SERVER_URI%/log/`

**ssoadm** attribute: `location`

### ElasticSearch JSON Format Compatible

JSON format should be transformed to be compatible with ElasticSearch format restrictions.

Default value: `false`

**ssoadm** attribute: `elasticsearchCompatible`

### File Rotation Retention Check Interval

Interval to check time-based file rotation policies, in seconds.

Default value: `5`

**ssoadm** attribute: `rotationRetentionCheckInterval`

## File Rotation

The File Rotation tab contains the following secondary configuration properties:

## Rotation Enabled

Enables and disables audit file rotation.

Default value: `true`

**ssoadm** attribute: `rotationEnabled`

## Maximum File Size

Maximum size, in bytes, which an audit file can grow to before rotation is triggered. A negative or zero value indicates this policy is disabled.

Default value: `100000000`

**ssoadm** attribute: `rotationMaxFileSize`

## File Rotation Prefix

Prefix to prepend to audit files when rotating audit files.

**ssoadm** attribute: `rotationFilePrefix`

## File Rotation Suffix

Suffix to append to audit files when they are rotated. Suffix should be a timestamp.

Default value: `-yyyy.MM.dd-HH.mm.ss`

**ssoadm** attribute: `rotationFileSuffix`

## Rotation Interval

Interval to trigger audit file rotations, in seconds. A negative or zero value disables this feature.

Default value: `-1`

**ssoadm** attribute: `rotationInterval`

## Rotation Times

Durations after midnight to trigger file rotation, in seconds.

**ssoadm** attribute: `rotationTimes`

## File Retention

The File Retention tab contains the following secondary configuration properties:

### Maximum Number of Historical Files

Maximum number of backup audit files allowed. A value of `-1` disables pruning of old history files.

Default value: `1`

**ssoadm** attribute: `retentionMaxNumberOfHistoryFiles`

### Maximum Disk Space

The maximum amount of disk space the audit files can occupy, in bytes. A negative or zero value indicates this policy is disabled.

Default value: `-1`

**ssoadm** attribute: `retentionMaxDiskSpaceToUse`

### Minimum Free Space Required

Minimum amount of disk space required, in bytes, on the system where audit files are stored. A negative or zero value indicates this policy is disabled.

Default value: `-1`

**ssoadm** attribute: `retentionMinFreeSpaceRequired`

## Buffering

The Buffering tab contains the following secondary configuration properties:

### Batch Size

Maximum number of audit log events that can be buffered.

Default value: `100000`

**ssoadm** attribute: `bufferingMaxSize`

### Write interval

Interval at which buffered events are written to a file, in milliseconds.

Default value: `5`

**ssoadm** attribute: `bufferingWriteInterval`

## 10.5.3.7. Splunk

A configured secondary instance of the Splunk type has the following tabs:

### General Handler Configuration

The General Handler Configuration tab contains the following secondary configuration properties:

#### Enabled

Enables or disables an audit event handler.

Default value: `true`

**ssoadm** attribute: `enabled`

## Topics

List of topics handled by an audit event handler.

Default value:

```
access
activity
config
authentication
```

**ssoadm** attribute: `topics`

## Audit Event Handler Factory

The Audit Event Handler Factory tab contains the following secondary configuration properties:

### Factory Class Name

The fully qualified class name of the factory responsible for creating the Audit Event Handler. The class must implement `org.forgerock.openam.audit.AuditEventHandlerFactory`.

Default value: `org.forgerock.openam.audit.events.handlers.SplunkAuditEventHandlerFactory`

**ssoadm** attribute: `handlerFactory`

## Splunk Configuration

The Splunk Configuration tab contains the following secondary configuration properties:

### Authorization Token

Authorization token used to connect to Splunk HTTP Event Collector endpoint.

**ssoadm** attribute: `authzToken`

### Server Hostname

Host name or IP address of Splunk server.

**ssoadm** attribute: `host`

### Server Port

Port number of Splunk server.

**ssoadm** attribute: `port`

## SSL Enabled

Use HTTPS protocol for communication with Splunk.

Default value: `false`

**ssoadm** attribute: `sslEnabled`

## Buffering

The Buffering tab contains the following secondary configuration properties:

### Batch Size

Number of audit log events to batch before submitting to Splunk.

Default value: `500`

**ssoadm** attribute: `batchSize`

### Queue Capacity

Maximum number of audit events in the batch queue; additional events are dropped.

Default value: `10000`

**ssoadm** attribute: `maxEvents`

### Write interval (in milliseconds)

Interval at which buffered events are written to Splunk.

Default value: `250`

**ssoadm** attribute: `writeInterval`

## 10.6. Logging

**ssoadm** service name: `logging`

### 10.6.1. General

The following settings appear on the **General** tab:

#### Log Status

Enable the OpenAM logging system.

OpenAM supports two Audit Logging Services: the legacy Logging Service, which is based on a Java SDK and is available in OpenAM versions prior to OpenAM 13.5, and a new common REST-based Audit Logging Service available from OpenAM 13.5.

The legacy Logging Service will be deprecated in a future release.

The possible values for this property are:

```
ACTIVE
INACTIVE
```

Default value: **INACTIVE**

**ssoadm** attribute: **status**

### Log Record Resolve Host Name

Enable this to have OpenAM perform a DNS host lookup to populate the host name field for log records.

*Note:* Enabling this functionality will increase the load of the logging system and the OpenAM host must have DNS configured.

Default value: **false**

**ssoadm** attribute: **resolveHostName**

### Logging Type

Specifies whether to log to a database, Syslog, or to the filing system.

If you choose database then be sure to set the connection attributes correctly, including the JDBC driver to use.

The possible values for this property are:

```
File
DB
Syslog
```

Default value: **File**

**ssoadm** attribute: **type**

### Configurable Log Fields

Controls the fields that are logged by OpenAM.

This property is the list of fields that are logged by default. Administrators can choose to limit the information logged by OpenAM.

Default value:

```
IPAddr
LoggedBy
LoginID
NameID
ModuleName
```

```
ContextID
Domain
LogLevel
HostName
MessageID
```

**ssoadm** attribute: `fields`

## Log Verification Frequency

The frequency (in seconds) that OpenAM verifies security of the log files.

When secure logging is enabled, this is the period that OpenAM will check the integrity of the log files.

Default value: `3600`

**ssoadm** attribute: `verifyPeriod`

## Log Signature Time

The frequency (in seconds) that OpenAM will digitally sign the log records.

When secure logging is enabled, this is the period that OpenAM will digitally signed the contents of the log files. The log signatures form the basis of the log file integrity checking.

Default value: `900`

**ssoadm** attribute: `signaturePeriod`

## Secure Logging

Enable or Disable secure logging.

Enabling this setting will cause OpenAM to digitally sign and verify the contents of the log files to help prevent and detect log file tampering. A certificate must be configured for this functionality to be enabled.

The possible values for this property are:

```
ON
OFF
```

Default value: `OFF`

**ssoadm** attribute: `security`

## Secure Logging Signing Algorithm

Determines the algorithm used to digitally sign the log records.

The possible values for this property are:

```
MD2withRSA
```

```
MD5withRSA  
SHA1withDSA  
SHA1withRSA
```

Default value: `SHA1withRSA`

**ssoadm** attribute: `signingAlgorithm`

### Logging Certificate Store Location

The path to the Java keystore containing the logging system certificate.

The secure logging system will use the certificate alias of `Logger` to locate the certificate in the specified keystore.

Default value: `%BASE_DIR%/%SERVER_URI%/Logger.jks`

**ssoadm** attribute: `certificateStore`

### Number of Files per Archive

Controls the number of logs files that will be archived by the secure logging system.

Default value: `5`

**ssoadm** attribute: `filesPerKeystore`

### Buffer Size

The number of log records held in memory before the log records will be flushed to the logfile or the database.

Default value: `25`

**ssoadm** attribute: `bufferSize`

### Buffer Time

The maximum time (in seconds) OpenAM will hold log records in memory before flushing to the underlying repository.

Default value: `60`

**ssoadm** attribute: `bufferTime`

### Time Buffering

Enable or Disable log buffering

When enabled OpenAM holds all log records in a memory buffer that it periodically flush to the repository. The period is set in the *Buffer Time* property.

The possible values for this property are:

```
ON
```



OFF

Default value: **ON**

**ssoadm** attribute: **buffering**

### Logging Level

Control the level of JDK logging within OpenAM.

The possible values for this property are:

OFF  
SEVERE  
WARNING  
INFO  
CONFIG  
FINE  
FINER  
FINEST

Default value: **INFO**

**ssoadm** attribute: **jdkLoggingLevel**

## 10.6.2. File

The following settings appear on the **File** tab:

### Log Rotation

Enable log rotation to cause new log files to be created when configured thresholds are reached, such as *Maximum Log Size* or *Logfile Rotation Interval*.

Default value: **true**

**ssoadm** attribute: **rotationEnabled**

### Maximum Log Size

Maximum size of a log file, in bytes.

Default value: **100000000**

**ssoadm** attribute: **maxFileSize**

### Number of History Files

Sets the number of history files for each log that OpenAM keeps, including time-based histories.

The previously live file is moved and is included in the history count, and a new log is created to serve as the live log file. Any log file in the history count that goes over the number specified here will be deleted.

For time-based logs, a new set of logs will be created when OpenAM is started because of the time-based file names that are used.

Default value: `1`

**ssoadm** attribute: `numberHistoryFiles`

### Logfile Rotation Prefix

The name of the log files will be prefixed with the supplied value.

This field defines the log file prefix. The prefix will be added to the name of all logfiles.

*Note:* Only used when time-based log rotation is enabled.

**ssoadm** attribute: `prefix`

### Logfile Rotation Suffix

The name of the log files will be suffixed with the supplied value.

This field defines the log file suffix. If no suffix is provided, then the following default suffix format will be used: `-MM.dd.yy-kk.mm`. The suffix allows use of Date and Time patterns defined in `SimpleDateFormat`

*Note:* This field is only used if the time based rotation is enabled.

Default value: `-MM.dd.yy-kk.mm`

**ssoadm** attribute: `suffix`

### Logfile Rotation Interval

The rotation interval (in minutes).

The rotation interval determines the frequency of when the log files will be rotated. If the value is `-1`, then time based rotation is disabled and log file size based rotation is enabled.

Default value: `-1`

**ssoadm** attribute: `rotationInterval`

### Log File Location

The path to the location of the log files

This property controls the location of the log files; the value of this property varies on whether File or DB logging is in use:

- File: The full pathname to the directory containing the log files.
- DB: The JDBC URL to the database used to store the log file database.

Default value: `%BASE_DIR%/SERVER_URI/log/`

**ssoadm** attribute: `location`

### 10.6.3. Database

The following settings appear on the **Database** tab:

#### Database User Name

When logging to a database, set this to the user name used to connect to the database. If this attribute is incorrectly set, OpenAM performance suffers.

Default value: `dbuser`

**ssoadm** attribute: `user`

#### Database User Password

When logging to a database, set this to the password used to connect to the database. If this attribute is incorrectly set, OpenAM performance suffers.

**ssoadm** attribute: `password`

#### Database Driver Name

When logging to a database, set this to the class name of the JDBC driver used to connect to the database.

The default is for Oracle. OpenAM also works with the MySQL database driver.

Default value: `oracle.jdbc.driver.OracleDriver`

**ssoadm** attribute: `driver`

#### Maximum Number of Records

The maximum number of records read from the logs via the Logging API

Default value: `500`

**ssoadm** attribute: `maxRecords`

#### DB Failure Memory Buffer Size

Max number of log records held in memory if DB logging fails.

This is the maximum number of log records that will be held in memory if the database is unavailable. When the buffer is full, new log records cause the oldest record in the buffer to be cleared. OpenAM monitoring records the number of log entries cleared when the database was unavailable.

If the value of this property is less than that of the *Buffer Size* then the buffer size value will take precedence.

Default value: 2

**ssoadm** attribute: `databaseFailureMemoryBufferSize`

## 10.6.4. Syslog

The following settings appear on the **Syslog** tab:

### Syslog server host

The URL or IP address of the syslog server, for example `http://mysyslog.example.com`, or `localhost`.

Default value: `localhost`

**ssoadm** attribute: `host`

### Syslog server port

The port number the syslog server is configured to listen to.

Default value: 514

**ssoadm** attribute: `port`

### Syslog transport protocol

The protocol to use to connect to the syslog server.

The possible values for this property are:

```
UDP
TCP
```

Default value: `UDP`

**ssoadm** attribute: `protocol`

### Syslog facility

Syslog uses the facility level to determine the type of program that is logging the message.

The possible values for this property are:

```
kern
user
mail
daemon
auth
syslog
lpr
news
uucp
```

```
cron
authpriv
ftp
local0
local1
local2
local3
local4
local5
local6
local7
```

Default value: `local5`

**ssoadm** attribute: `facility`

### Syslog connection timeout

The amount of time to wait when attempting to connect to the syslog server before reporting a failure, in seconds.

Default value: `30`

**ssoadm** attribute: `timeout`

## 10.7. Monitoring

**ssoadm** service name: `monitoring`

The following settings are available in this service:

### Monitoring Status

Enable / Disable the monitoring system

Default value: `false`

**ssoadm** attribute: `enabled`

### Monitoring HTTP Port

Port number for the HTTP monitoring interface

Default value: `8082`

**ssoadm** attribute: `httpPort`

### Monitoring HTTP interface status

Enable / Disable the HTTP access to the monitoring system

Default value: `false`

**ssoadm** attribute: `httpEnabled`

### Monitoring HTTP interface authentication file path

Path to the monitoring system authentication file

The `openam_mon_auth` file contains the username and password of the account used to protect the monitoring interfaces. The default username is `demo` with a password of `changeit`. Use the `ampassword` command to encrypt a new password.

Default value: `%BASE_DIR%/%SERVER_URI%/openam_mon_auth`

**ssoadm** attribute: `authfilePath`

### Monitoring RMI Port

Port number for the JMX monitoring interface

Default value: `9999`

**ssoadm** attribute: `rmiPort`

### Monitoring RMI interface status

Enable / Disable the JMX access to the monitoring system

Default value: `false`

**ssoadm** attribute: `rmiEnabled`

### Monitoring SNMP Port

Port number for the SNMP monitoring interface

Default value: `8085`

**ssoadm** attribute: `snmpPort`

### Monitoring SNMP interface status

Enable / Disable the SNMP access to the monitoring system

Default value: `false`

**ssoadm** attribute: `snmpEnabled`

### Policy evaluation monitoring history size

Size of the window of most recent policy evaluations to record to expose via monitoring system. Valid range is 100 - 1000000.

Default value: `10000`

**ssoadm** attribute: `policyHistoryWindowSize`

### Session monitoring history size

Size of the window of most recent session operations to record to expose via monitoring system. Valid range is 100 - 1000000.

Default value: 10000

**ssoadm** attribute: `sessionHistoryWindowSize`

## 10.8. OAuth2 Provider

**ssoadm** service name: `oauth-oidc`

### 10.8.1. Global Attributes

The following settings appear on the **Global Attributes** tab:

#### Token Blacklist Cache Size

Number of blacklisted tokens to cache in memory to speed up blacklist checks and reduce load on the CTS.

Default value: 10000

**ssoadm** attribute: `blacklistCacheSize`

#### Blacklist Poll Interval (seconds)

How frequently to poll for token blacklist changes from other servers, in seconds.

How often each server will poll the CTS for token blacklist changes from other servers. This is used to maintain a highly compressed view of the overall current token blacklist improving performance. A lower number will reduce the delay for blacklisted tokens to propagate to all servers at the cost of increased CTS load. Set to 0 to disable this feature completely.

Default value: 60

**ssoadm** attribute: `blacklistPollInterval`

#### Blacklist Purge Delay (minutes)

Length of time to blacklist tokens beyond their expiry time.

Allows additional time to account for clock skew to ensure that a token has expired before it is removed from the blacklist.

Default value: 1

**ssoadm** attribute: `blacklistPurgeDelay`

### HMAC ID Token Authenticity Secret

A secret to use when signing a claim in HMAC-signed ID tokens so that authenticity can be assured when they are presented back to OpenAM.

Default value: `l6QZJe404be65x8TU7F2ihonPxCgimk5ekI0L+L50Zc=`

**ssoadm** attribute: `idTokenAuthenticitySecret`

### ID Token Signing Key Alias for Agent Clients

The alias for the RSA key that should be used signing ID tokens for Agent OAuth2 Clients

Default value: `test`

**ssoadm** attribute: `agentIdTokenSigningKeyAlias`

## 10.8.2. Core

The following settings appear on the **Core** tab:

### Use Stateless Access & Refresh Tokens

When enabled, OpenAM issues access and refresh tokens that can be inspected by resource servers.

Default value: `false`

**ssoadm** attribute: `statelessTokensEnabled`

### Authorization Code Lifetime (seconds)

The time an authorization code is valid for, in seconds.

Default value: `120`

**ssoadm** attribute: `codeLifetime`

### Refresh Token Lifetime (seconds)

The time in seconds a refresh token is valid for. If this field is set to `-1`, the token will never expire.

Default value: `604800`

**ssoadm** attribute: `refreshTokenLifetime`

### Access Token Lifetime (seconds)

The time an access token is valid for, in seconds.

Default value: `3600`

**ssoadm** attribute: `accessTokenLifetime`



## Issue Refresh Tokens

Whether to issue a refresh token when returning an access token.

Default value: `true`

**ssoadm** attribute: `issueRefreshToken`

## Issue Refresh Tokens on Refreshing Access Tokens

Whether to issue a refresh token when refreshing an access token.

Default value: `true`

**ssoadm** attribute: `issueRefreshTokenOnRefreshedToken`

## Saved Consent Attribute Name

Name of a multi-valued attribute on resource owner profiles where OpenAM can save authorization consent decisions.

When the resource owner chooses to save the decision to authorize access for a client application, then OpenAM updates the resource owner's profile to avoid having to prompt the resource owner to grant authorization when the client issues subsequent authorization requests.

**ssoadm** attribute: `savedConsentAttribute`

## 10.8.3. Advanced

The following settings appear on the **Advanced** tab:

### Custom Login URL Template

Custom URL for handling login, to override the default OpenAM login page.

Supports Freemarker syntax, with the following variables:

Variable	Description
<code>gotoUrl</code>	The URL to redirect to after login.
<code>acrValues</code>	The Authentication Context Class Reference (acr) values for the authorization request.
<code>realm</code>	The OpenAM realm the authorization request was made on.
<code>module</code>	The name of the OpenAM authentication module requested to perform resource owner authentication.
<code>service</code>	The name of the OpenAM authentication chain requested to perform resource owner authentication.

<code>locale</code>	A space-separated list of locales, ordered by preference.
---------------------	---

The following example template redirects users to a non-OpenAM front end to handle login, which will then redirect back to the `/oauth2/authorize` endpoint with any required parameters:

```
http://mylogin.com/login?goto=${goto}<#if acrValues??>&acr_values=${acrValues}</if><#if realm??>&realm=${realm}</if><#if module??>&module=${module}</if><#if service??>&service=${service}</if><#if locale??>&locale=${locale}</if>
```

**ssoadm** attribute: `customLoginUrlTemplate`

## Scope Implementation Class

The class that contains the required scope implementation, must implement the `org.forgerock.oauth2.core.ScopeValidator` interface.

Default value: `org.forgerock.openam.oauth2.OpenAMScopeValidator`

**ssoadm** attribute: `scopeImplementationClass`

## Response Type Plugins

List of plugins that handle the valid `response_type` values.

OAuth 2.0 clients pass response types as parameters to the OAuth 2.0 Authorization endpoint (`/oauth2/authorize`) to indicate which grant type is requested from the provider. For example, the client passes `code` when requesting an authorization code, and `token` when requesting an access token.

Values in this list take the form `response-type|plugin-class-name`.

Default value:

```
code|org.forgerock.oauth2.core.AuthorizationCodeResponseTypeHandler
device_code|org.forgerock.oauth2.core.TokenResponseTypeHandler
token|org.forgerock.oauth2.core.TokenResponseTypeHandler
```

**ssoadm** attribute: `responseTypeClasses`

## User Profile Attribute(s) the Resource Owner is Authenticated On

Names of profile attributes that resource owners use to log in. You can add others to the default, for example `mail`.

Default value: `uid`

**ssoadm** attribute: `authenticationAttributes`

## User Display Name attribute

The profile attribute that contains the name to be displayed for the user on the consent page.

Default value: `cn`

**ssoadm** attribute: `displayNameAttribute`

## Supported Scopes

The set of supported scopes, with translations.

Scopes may be entered as simple strings or pipe-separated strings representing the internal scope name, locale, and localized description.

For example: `read|en|Permission to view email messages in your account`

Locale strings are in the format: `language_country_variant`, for example `en`, `en_GB`, or `en_US_WIN`.

If the locale and pipe is omitted, the description is displayed to all users that have undefined locales.

If the description is also omitted, nothing is displayed on the consent page for the scope. For example specifying `read|` would allow the scope `read` to be used by the client, but would not display it to the user on the consent page when requested.

**ssoadm** attribute: `supportedScopes`

## Subject Types supported

List of subject types supported. Valid values are:

- `public` - Each client receives the same subject (`sub`) value.
- `pairwise` - Each client receives a different subject (`sub`) value, to prevent correlation between clients.

Default value: `public`

**ssoadm** attribute: `supportedSubjectTypes`

## Default Client Scopes

List of scopes a client will be granted if they request registration without specifying which scopes they want. Default scopes are NOT auto-granted to clients created through the OpenAM console.

**ssoadm** attribute: `defaultScopes`

## OAuth2 Token Signing Algorithm

Algorithm used to sign stateless OAuth 2.0 tokens in order to detect tampering.

OpenAM supports signing algorithms listed in JSON Web Algorithms (JWA): "alg" (Algorithm) Header Parameter Values for JWS:

- **HS256** - HMAC with SHA-256.
- **HS384** - HMAC with SHA-384.
- **HS512** - HMAC with SHA-512.
- **ES256** - ECDSA with SHA-256 and NIST standard P-256 elliptic curve.
- **ES384** - ECDSA with SHA-384 and NIST standard P-384 elliptic curve.
- **ES512** - ECDSA with SHA-512 and NIST standard P-521 elliptic curve.
- **RS256** - RSASSA-PKCS-v1\_5 using SHA-256.

The possible values for this property are:

```
HS256
HS384
HS512
RS256
ES256
ES384
ES512
```

Default value: **HS256**

**ssoadm** attribute: **tokenSigningAlgorithm**

### Stateless Token Compression

Whether stateless access and refresh tokens should be compressed.

**ssoadm** attribute: **tokenCompressionEnabled**

### Token Signing HMAC Shared Secret

Base64-encoded key used by HS256, HS384 and HS512.

Default value: **l6QZJe404be65x8TU7F2ihonPxGimk5ekI0L+L50Zc=**

**ssoadm** attribute: **tokenSigningHmacSharedSecret**

### Token Signing RSA public/private key pair

The public/private key pair used by RS256.

The public/private key pair will be retrieved from the keystore referenced by the property **com.sun.identity.saml.xmlsig.keystore**.

Default value: **test**

**ssoadm** attribute: **keypairName**

## Token Signing ECDSA public/private key pair alias

The list of public/private key pairs used for the elliptic curve algorithms (ES256/ES384/ES512). Add an entry to specify an alias for a specific elliptic curve algorithm, for example `ES256|es256Alias`.

Each of the public/private key pairs will be retrieved from the keystore referenced by the property `com.sun.identity.saml.xmlsig.keystore`.

Default value:

```
ES512|es512test
ES384|es384test
ES256|es256test
```

**ssoadm** attribute: `tokenSigningECDSAKeyAlias`

## Subject identifier hash salt

If *pairwise* subject types are supported, it is *STRONGLY RECOMMENDED* to change this value. It is used in the salting of hashes for returning specific `sub` claims to individuals using the same `request_uri` or `sector_identifier_uri`.

For example, you might set this property to: *changeme*

**ssoadm** attribute: `hashSalt`

## Code verifier parameter required

If enabled, requests using the authorization code grant require a `code_challenge` attribute.

For more information, read the [draft specification](#) for this feature.

Default value: `false`

**ssoadm** attribute: `codeVerifierEnforced`

## Modified Timestamp attribute name

The identity Data Store attribute used to return modified timestamp values.

**ssoadm** attribute: `modifiedTimestampAttribute`

## Created Timestamp attribute name

The identity Data Store attribute used to return created timestamp values.

**ssoadm** attribute: `createdTimestampAttribute`

## Allow clients to skip consent

If enabled, clients may be configured so that the resource owner will not be asked for consent during authorization flows.

Default value: `false`

**ssoadm** attribute: `clientsCanSkipConsent`

### Enable auth module messages for Password Credentials Grant

If enabled, authentication module failure messages are used to create Resource Owner Password Credentials Grant failure messages. If disabled, a standard authentication failed message is used.

The Password Grant Type requires the `grant_type=password` parameter.

Default value: `false`

**ssoadm** attribute: `moduleMessageEnabledInPasswordGrant`

## 10.8.4. OpenID Connect

The following settings appear on the **OpenID Connect** tab:

### OIDC Claims Script

The script that is run when issuing an ID token or making a request to the *userinfo* endpoint during OpenID requests.

The script gathers the scopes and populates claims, and has access to the access token, the user's identity and, if available, the user's session.

The possible values for this property are:

OIDC Claims Script

Default value: `OIDC Claims Script`

**ssoadm** attribute: `oidcClaimsScript`

### ID Token Signing Algorithms supported

Algorithms supported to sign OpenID Connect `id_tokens`.

OpenAM supports signing algorithms listed in JSON Web Algorithms (JWA): "alg" (Algorithm) Header Parameter Values for JWS:

- `HS256` - HMAC with SHA-256.
- `HS384` - HMAC with SHA-384.
- `HS512` - HMAC with SHA-512.
- `ES256` - ECDSA with SHA-256 and NIST standard P-256 elliptic curve.
- `ES384` - ECDSA with SHA-384 and NIST standard P-384 elliptic curve.
- `ES512` - ECDSA with SHA-512 and NIST standard P-521 elliptic curve.
- `RS256` - RSASSA-PKCS-v1\_5 using SHA-256.

Default value:

```
ES384
HS256
HS512
ES256
RS256
HS384
ES512
```

**ssoadm** attribute: `supportedIDTokenSigningAlgorithms`

## ID Token Encryption Algorithms supported

Encryption algorithms supported to encrypt OpenID Connect ID tokens in order to hide its contents.

OpenAM supports the following ID token encryption algorithms:

- `RSA-OAEP` - RSA with Optimal Asymmetric Encryption Padding (OAEP) with SHA-1 and MGF-1.
- `RSA-OAEP-256` - RSA with OAEP with SHA-256 and MGF-1.
- `A128KW` - AES Key Wrapping with 128-bit key derived from the client secret.
- `RSA1_5` - RSA with PKCS#1 v1.5 padding.
- `A256KW` - AES Key Wrapping with 256-bit key derived from the client secret.
- `dir` - Direct encryption with AES using the hashed client secret.
- `A192KW` - AES Key Wrapping with 192-bit key derived from the client secret.

Default value:

```
RSA-OAEP
RSA-OAEP-256
A128KW
RSA1_5
A256KW
dir
A192KW
```

**ssoadm** attribute: `supportedIDTokenEncryptionAlgorithms`

## ID Token Encryption Methods supported

Encryption methods supported to encrypt OpenID Connect ID tokens in order to hide its contents.

OpenAM supports the following ID token encryption algorithms:

- `A128GCM`, `A192GCM`, and `A256GCM` - AES in Galois Counter Mode (GCM) authenticated encryption mode.

- **A128CBC-HS256**, **A192CBC-HS384**, and **A256CBC-HS512** - AES encryption in CBC mode, with HMAC-SHA-2 for integrity.

Default value:

```
A256GCM
A192GCM
A128GCM
A128CBC-HS256
A192CBC-HS384
A256CBC-HS512
```

**ssoadm** attribute: **supportedIDTokenEncryptionMethods**

## Supported Claims

Set of claims supported by the OpenID Connect **/oauth2/userinfo** endpoint, with translations.

Claims may be entered as simple strings or pipe separated strings representing the internal claim name, locale, and localized description.

For example: **name|en|Your full name..**

Locale strings are in the format: **language + "\_" + country + "\_" + variant**, for example **en**, **en\_GB**, or **en\_US\_WIN**. If the locale and pipe is omitted, the description is displayed to all users that have undefined locales.

If the description is also omitted, nothing is displayed on the consent page for the claim. For example specifying **family\_name|** would allow the claim **family\_name** to be used by the client, but would not display it to the user on the consent page when requested.

**ssoadm** attribute: **supportedClaims**

## OpenID Connect JWT Token Lifetime (seconds)

The amount of time the JWT will be valid for, in seconds.

Default value: **3600**

**ssoadm** attribute: **jwtTokenLifetime**

## 10.8.5. Advanced OpenID Connect

The following settings appear on the **Advanced OpenID Connect** tab:

### Remote JSON Web Key URL

The Remote URL where the providers JSON Web Key can be retrieved.

If this setting is not configured, then OpenAM provides a local URL to access the public key of the private key used to sign ID tokens.



**ssoadm** attribute: `jkwsURI`

### Idtokeninfo endpoint requires client authentication

When enabled, the `/oauth2/idtokeninfo` endpoint requires client authentication if the signing algorithm is set to `HS256`, `HS384`, or `HS512`.

Default value: `true`

**ssoadm** attribute: `idTokenInfoClientAuthenticationEnabled`

### Enable "claims\_parameter\_supported"

If enabled, clients will be able to request individual claims using the `claims` request parameter, as per section 5.5 of the OpenID Connect specification.

Default value: `false`

**ssoadm** attribute: `claimsParameterSupported`

### Allow Open Dynamic Client Registration

Allow clients to register without an access token. If enabled, you should consider adding some form of rate limiting. See [Client Registration](#) in the OpenID Connect specification for details.

Default value: `false`

**ssoadm** attribute: `allowDynamicRegistration`

### Generate Registration Access Tokens

Whether to generate Registration Access Tokens for clients that register via open dynamic client registration. Such tokens allow the client to access the [Client Configuration Endpoint](#) as per the OpenID Connect specification. This setting has no effect if open dynamic client registration is disabled.

Default value: `true`

**ssoadm** attribute: `generateRegistrationAccessTokens`

### OpenID Connect acr\_values to Auth Chain Mapping

Maps OpenID Connect ACR values to authentication chains. See the `acr_values` parameter in the OpenID Connect authentication request specification for more details.

**ssoadm** attribute: `loaMapping`

### OpenID Connect default acr claim

Default value to use as the `acr` claim in an OpenID Connect ID Token when using the default authentication chain.

**ssoadm** attribute: `defaultACR`

### OpenID Connect id\_token amr values to Auth Module mappings

Specify `amr` values to be returned in the OpenID Connect `id_token`. Once authentication has completed, the authentication modules that were used from the authentication service will be mapped to the `amr` values. If you do not require `amr` values, or are not providing OpenID Connect tokens, leave this field blank.

**ssoadm** attribute: `amrMappings`

### Always return claims in ID Tokens

If enabled, include scope-derived claims in the `id_token`, even if an access token is also returned that could provide access to get the claims from the `userinfo` endpoint.

If not enabled, if an access token is requested the client must use it to access the `userinfo` endpoint for scope-derived claims, as they will not be included in the ID token.

Default value: `false`

**ssoadm** attribute: `alwaysAddClaimsToToken`

### Store Ops Tokens

Whether OpenAM will store the *ops* tokens corresponding to OpenID Connect sessions in the CTS store. Note that session management related endpoints will not work when this setting is disabled.

Default value: `true`

**ssoadm** attribute: `storeOpsTokens`

## 10.8.6. Device Flow

The following settings appear on the **Device Flow** tab:

### Verification URL

The URL that the user will be instructed to visit to complete their OAuth 2.0 login and consent when using the device code flow.

**ssoadm** attribute: `verificationUrl`

### Device Completion URL

The URL that the user will be sent to on completion of their OAuth 2.0 login and consent when using the device code flow.

**ssoadm** attribute: `completionUrl`

### Device Code Lifetime (seconds)

The lifetime of the device code, in seconds.

Default value: 300

**ssoadm** attribute: `deviceCodeLifetime`

### Device Polling Interval

The polling frequency for devices waiting for tokens when using the device code flow.

Default value: 5

**ssoadm** attribute: `devicePollInterval`

## 10.9. Dashboard

**ssoadm** service name: `dashboard`

### 10.9.1. Realm Defaults

The following settings appear on the *Realm Defaults* tab:

#### Available Dashboard Apps

List of application dashboard names available by default for realms with the Dashboard service configured.

**ssoadm** attribute: `assignedDashboard`

### 10.9.2. Secondary Configurations

This service has the following Secondary Configurations.

#### 10.9.2.1. instances

##### Dashboard Class Name

Identifies how to access the application, for example `SAML2ApplicationClass` for a SAML v2.0 application.

**ssoadm** attribute: `className`

##### Dashboard Name

The application name as it will appear to the administrator for configuring the dashboard.

**ssoadm** attribute: `name`

##### Dashboard Display Name

The application name that displays on the dashboard client.

**ssoadm** attribute: `displayName`

### Dashboard Icon

The icon name that will be displayed on the dashboard client identifying the application.

**ssoadm** attribute: `icon`

### Dashboard Login

The URL that takes the user to the application.

**ssoadm** attribute: `login`

### ICF Identifier

**ssoadm** attribute: `icfIdentifier`

## 10.10. Command-line Tool Reference

## Name

ampassword — change passwords for the OpenAM Administrator

## Synopsis

```
ampassword {options}
```

## Description

This command allows you to change passwords held in the configuration store, and to encrypt passwords.

## Options

The following options are supported.

```
-a | --admin [ -o | --old old-password-file -n | --new new-password-file ]
```

Change the password for `amAdmin` from the value stored in *old-password-file* to the value stored in *new-password-file*.

```
-p | --proxy [ -o | --old old-password-file -n | --new new-password-file ]
```

Change the password for the proxy administrator from the value stored in *old-password-file* to the value stored in *new-password-file*.

The proxy administrator password is shown encrypted in the output from **ssoadm get-svrcfg-xml**.

```
-e | --encrypt [ password-file ]
```

Display the password value provided encrypted with the key generated during OpenAM installation.

```
-h | --help
```

Display the usage message.

## Examples

The following example encrypts the password contained within a text file.

- Create a text file, for example `$HOME/.pwd.txt`, containing the password string on a single line.
- Encrypt the password by using the **ampassword** command:

```
$ ampassword -e $HOME/.pwd.txt
AQICkZs3qy5QUCXir9tebIEEZYGFxi2lCC4B
```

# Appendix A. About the REST API

This appendix shows how to use the RESTful interfaces for direct integration between web client applications and ForgeRock Access Management.

## A.1. Introducing REST

Representational State Transfer (REST) is an architectural style that sets certain constraints for designing and building large-scale distributed hypermedia systems.

As an architectural style, REST has very broad applications. The designs of both HTTP 1.1 and URIs follow RESTful principles. The World Wide Web is no doubt the largest and best known REST application. Many other web services also follow the REST architectural style. Examples include OAuth 2.0, OpenID Connect 1.0, and User-Managed Access (UMA).

The ForgeRock Common REST (CREST) API applies RESTful principles to define common verbs for HTTP-based APIs that access web resources and collections of web resources.

Interface Stability: [Evolving](#)

Most native OpenAM REST APIs use the CREST verbs. (In contrast, OAuth 2.0, OpenID Connect 1.0 and UMA APIs follow their respective standards.)

## A.2. About ForgeRock Common REST

ForgeRock® Common REST is a common REST API framework. It works across the ForgeRock platform to provide common ways to access web resources and collections of resources. Adapt the examples in this section to your resources and deployment.

### A.2.1. Common REST Resources

Servers generally return JSON-format resources, though resource formats can depend on the implementation.

Resources in collections can be found by their unique identifiers (IDs). IDs are exposed in the resource URIs. For example, if a server has a user collection under `/users`, then you can access a user at `/users/user-id`. The ID is also the value of the `_id` field of the resource.

Resources are versioned using revision numbers. A revision is specified in the resource's `_rev` field. Revisions make it possible to figure out whether to apply changes without resource locking and without distributed transactions.

### A.2.2. Common REST Verbs

The Common REST APIs use the following verbs, sometimes referred to collectively as CRUDPAQ. For details and HTTP-based examples of each, follow the links to the sections for each verb.

#### Create

Add a new resource.

This verb maps to HTTP PUT or HTTP POST.

For details, see [Section A.2.6, "Create"](#).

#### Read

Retrieve a single resource.

This verb maps to HTTP GET.

For details, see [Section A.2.7, "Read"](#).

#### Update

Replace an existing resource.

This verb maps to HTTP PUT.

For details, see [Section A.2.8, "Update"](#).

#### Delete

Remove an existing resource.

This verb maps to HTTP DELETE.

For details, see [Section A.2.9, "Delete"](#).

## Patch

Modify part of an existing resource.

This verb maps to HTTP PATCH.

For details, see [Section A.2.10, "Patch"](#).

## Action

Perform a predefined action.

This verb maps to HTTP POST.

For details, see [Section A.2.11, "Action"](#).

## Query

Search a collection of resources.

This verb maps to HTTP GET.

For details, see [Section A.2.12, "Query"](#).

## A.2.3. Common REST Parameters

Common REST reserved query string parameter names start with an underscore, `_`.

Reserved query string parameters include, but are not limited to, the following names:

```
_action  
_api  
_crestapi  
_fields  
_mimeType  
_pageSize  
_pagedResultsCookie  
_pagedResultsOffset  
_prettyPrint  
_queryExpression  
_queryFilter  
_queryId  
_sortKeys  
_totalPagedResultsPolicy
```

### Note

Some parameter values are not safe for URLs, so URL-encode parameter values as necessary.

Continue reading for details about how to use each parameter.



## A.2.4. Common REST Extension Points

The *action* verb is the main vehicle for extensions. For example, to create a new user with HTTP POST rather than HTTP PUT, you might use `/users?_action=create`. A server can define additional actions. For example, `/tasks/1?_action=cancel`.

A server can define *stored queries* to call by ID. For example, `/groups?_queryId=hasDeletedMembers`. Stored queries can call for additional parameters. The parameters are also passed in the query string. Which parameters are valid depends on the stored query.

## A.2.5. Common REST API Documentation

Common REST APIs often depend at least in part on runtime configuration. Many Common REST endpoints therefore serve *API descriptors* at runtime. An API descriptor documents the actual API as it is configured.

Use the following query string parameters to retrieve API descriptors:

### `_api`

Serves an API descriptor that complies with the OpenAPI specification.

This API descriptor represents the API accessible over HTTP. It is suitable for use with popular tools such as Swagger UI.

### `_crestapi`

Serves a native Common REST API descriptor.

This API descriptor provides a compact representation that is not dependent on the transport protocol. It requires a client that understands Common REST, as it omits many Common REST defaults.

#### Note

Consider limiting access to API descriptors in production environments in order to avoid unnecessary traffic.

To provide documentation in production environments, see Procedure A.1, "To Publish OpenAPI Documentation" instead.

### Procedure A.1. To Publish OpenAPI Documentation

In production systems, developers expect stable, well-documented APIs. Rather than retrieving API descriptors at runtime through Common REST, prepare final versions, and publish them alongside the software in production.

Use the OpenAPI-compliant descriptors to provide API reference documentation for your developers as described in the following steps:

1. Configure the software to produce production-ready APIs.

In other words, the software should be configured as in production so that the APIs are identical to what developers see in production.

2. Retrieve the OpenAPI-compliant descriptor.

The following command saves the descriptor to a file, `myapi.json`:

```
$ curl -o myapi.json endpoint?_api
```

3. If necessary, edit the descriptor.

For example, you might want to add security definitions to describe how the API is protected.

If you make any changes, then also consider using a source control system to manage your versions of the API descriptor.

4. Publish the descriptor using a tool such as [Swagger UI](#).

You can customize Swagger UI for your organization as described in the documentation for the tool.

## A.2.6. Create

There are two ways to create a resource, either with an HTTP POST or with an HTTP PUT.

To create a resource using POST, perform an HTTP POST with the query string parameter `_action=create` and the JSON resource as a payload. Accept a JSON response. The server creates the identifier if not specified:

```
POST /users?_action=create HTTP/1.1
Host: example.com
Accept: application/json
Content-Length: ...
Content-Type: application/json
{ JSON resource }
```

To create a resource using PUT, perform an HTTP PUT including the case-sensitive identifier for the resource in the URL path, and the JSON resource as a payload. Use the `If-None-Match: *` header. Accept a JSON response:

```
PUT /users/some-id HTTP/1.1
Host: example.com
Accept: application/json
Content-Length: ...
Content-Type: application/json
If-None-Match: *
{ JSON resource }
```

The `_id` and content of the resource depend on the server implementation. The server is not required to use the `_id` that the client provides. The server response to the create request indicates the resource location as the value of the `Location` header.

If you include the `If-None-Match` header, its value must be `*`. In this case, the request creates the object if it does not exist, and fails if the object does exist. If you include the `If-None-Match` header with any value other than `*`, the server returns an HTTP 400 Bad Request error. For example, creating an object with `If-None-Match: revision` returns a bad request error. If you do not include `If-None-Match: *`, the request creates the object if it does not exist, and *updates* the object if it does exist.

## Parameters

You can use the following parameters:

`_prettyPrint=true`

Format the body of the response.

`_fields=field[,field...]`

Return only the specified fields in the body of the response.

The `field` values are JSON pointers. For example if the resource is `{"parent":{"child":"value"}}`, `parent/child` refers to the `"child":"value"`.

### A.2.7. Read

To retrieve a single resource, perform an HTTP GET on the resource by its case-sensitive identifier (`_id`) and accept a JSON response:

```
GET /users/some-id HTTP/1.1
Host: example.com
Accept: application/json
```

## Parameters

You can use the following parameters:

`_prettyPrint=true`

Format the body of the response.

`_fields=field[,field...]`

Return only the specified fields in the body of the response.

The `field` values are JSON pointers. For example if the resource is `{"parent":{"child":"value"}}`, `parent/child` refers to the `"child":"value"`.

### `_mimeType=mime-type`

Some resources have fields whose values are multi-media resources such as a profile photo for example.

By specifying both a single *field* and also the *mime-type* for the response content, you can read a single field value that is a multi-media resource.

In this case, the content type of the field value returned matches the *mime-type* that you specify, and the body of the response is the multi-media resource.

The `Accept` header is not used in this case. For example, `Accept: image/png` does not work. Use the `_mimeType` query string parameter instead.

## A.2.8. Update

To update a resource, perform an HTTP PUT including the case-sensitive identifier (`_id`) as the final element of the path to the resource, and the JSON resource as the payload. Use the `If-Match: _rev` header to check that you are actually updating the version you modified. Use `If-Match: *` if the version does not matter. Accept a JSON response:

```
PUT /users/some-id HTTP/1.1
Host: example.com
Accept: application/json
Content-Length: ...
Content-Type: application/json
If-Match: _rev
{ JSON resource }
```

When updating a resource, include all the attributes to be retained. Omitting an attribute in the resource amounts to deleting the attribute unless it is not under the control of your application. Attributes not under the control of your application include private and read-only attributes. In addition, virtual attributes and relationship references might not be under the control of your application.

### Parameters

You can use the following parameters:

#### `_prettyPrint=true`

Format the body of the response.

#### `_fields=field[,field...]`

Return only the specified fields in the body of the response.

The `field` values are JSON pointers. For example if the resource is `{"parent":{"child":"value"}}`, `parent/child` refers to the `"child":"value"`.

### A.2.9. Delete

To delete a single resource, perform an HTTP DELETE by its case-sensitive identifier (`_id`) and accept a JSON response:

```
DELETE /users/some-id HTTP/1.1
Host: example.com
Accept: application/json
```

#### Parameters

You can use the following parameters:

`_prettyPrint=true`

Format the body of the response.

`_fields=field[,field...]`

Return only the specified fields in the body of the response.

The `field` values are JSON pointers. For example if the resource is `{"parent":{"child":"value"}}`, `parent/child` refers to the `"child":"value"`.

### A.2.10. Patch

To patch a resource, send an HTTP PATCH request with the following parameters:

- `operation`
- `field`
- `value`
- `from` (optional with copy and move operations)

You can include these parameters in the payload for a PATCH request, or in a JSON PATCH file. If successful, you'll see a JSON response similar to:

```
PATCH /users/some-id HTTP/1.1
Host: example.com
Accept: application/json
Content-Length: ...
Content-Type: application/json
If-Match: _rev
{ JSON array of patch operations }
```

PATCH operations apply to three types of targets:

- **single-valued**, such as an object, string, boolean, or number.

- **list semantics array**, where the elements are ordered, and duplicates are allowed.
- **set semantics array**, where the elements are not ordered, and duplicates are not allowed.

ForgeRock PATCH supports several different **operations**. The following sections show each of these operations, along with options for the **field** and **value**:

#### A.2.10.1. Patch Operation: Add

The **add** operation ensures that the target field contains the value provided, creating parent fields as necessary.

If the target field is single-valued, then the value you include in the PATCH replaces the value of the target. Examples of a single-valued field include: object, string, boolean, or number.

An **add** operation has different results on two standard types of arrays:

- **List semantic arrays**: you can run any of these **add** operations on that type of array:
  - If you **add** an array of values, the PATCH operation appends it to the existing list of values.
  - If you **add** a single value, specify an ordinal element in the target array, or use the **{-}** special index to add that value to the end of the list.
- **Set semantic arrays**: The list of values included in a patch are merged with the existing set of values. Any duplicates within the array are removed.

As an example, start with the following list semantic array resource:

```
{
  "fruits" : [ "orange", "apple" ]
}
```

The following add operation includes the pineapple to the end of the list of fruits, as indicated by the **-** at the end of the **fruits** array.

```
{
  "operation" : "add",
  "field" : "/fruits/-",
  "value" : "pineapple"
}
```

The following is the resulting resource:

```
{
  "fruits" : [ "orange", "apple", "pineapple" ]
}
```

#### A.2.10.2. Patch Operation: Copy

The copy operation takes one or more existing values from the source field. It then adds those same values on the target field. Once the values are known, it is equivalent to performing an **add** operation on the target field.

The following **copy** operation takes the value from a field named **mail**, and then runs a **replace** operation on the target field, **another\_mail**.

```
[
  {
    "operation": "copy",
    "from": "mail",
    "field": "another_mail"
  }
]
```

If the source field value and the target field value are configured as arrays, the result depends on whether the array has list semantics or set semantics, as described in Section A.2.10.1, "Patch Operation: Add".

### A.2.10.3. Patch Operation: Increment

The **increment** operation changes the value or values of the target field by the amount you specify. The value that you include must be one number, and may be positive or negative. The value of the target field must accept numbers. The following **increment** operation adds **1000** to the target value of **/user/payment**.

```
[
  {
    "operation" : "increment",
    "field" : "/user/payment",
    "value" : "1000"
  }
]
```

Since the **value** of the **increment** is a single number, arrays do not apply.

### A.2.10.4. Patch Operation: Move

The move operation removes existing values on the source field. It then adds those same values on the target field. It is equivalent to performing a **remove** operation on the source, followed by an **add** operation with the same values, on the target.

The following **move** operation is equivalent to a **remove** operation on the source field, **surname**, followed by a **replace** operation on the target field value, **lastName**. If the target field does not exist, it is created.

```
[
  {
    "operation": "move",
    "from": "surname",
    "field": "lastName"
  }
]
```

To apply a **move** operation on an array, you need a compatible single-value, list semantic array, or set semantic array on both the source and the target. For details, see the criteria described in Section A.2.10.1, "Patch Operation: Add".

### A.2.10.5. Patch Operation: Remove

The **remove** operation ensures that the target field no longer contains the value provided. If the remove operation does not include a value, the operation removes the field. The following **remove** deletes the value of the **phoneNumber**, along with the field.

```
[
  {
    "operation" : "remove",
    "field" : "phoneNumber"
  }
]
```

If the object has more than one **phoneNumber**, those values are stored as an array.

A **remove** operation has different results on two standard types of arrays:

- **List semantic arrays:** A **remove** operation deletes the specified element in the array. For example, the following operation removes the first phone number, based on its array index (zero-based):

```
[
  {
    "operation" : "remove",
    "field" : "/phoneNumber/0"
  }
]
```

- **Set semantic arrays:** The list of values included in a patch are removed from the existing array.

### A.2.10.6. Patch Operation: Replace

The **replace** operation removes any existing value(s) of the targeted field, and replaces them with the provided value(s). It is essentially equivalent to a **remove** followed by a **add** operation. If the arrays are used, the criteria is based on Section A.2.10.1, "Patch Operation: Add". However, indexed updates are not allowed, even when the target is an array.

The following **replace** operation removes the existing **telephoneNumber** value for the user, and then adds the new value of **+1 408 555 9999**.

```
[
  {
    "operation" : "replace",
    "field" : "/telephoneNumber",
    "value" : "+1 408 555 9999"
  }
]
```

A PATCH replace operation on a list semantic array works in the same fashion as a PATCH remove operation. The following example demonstrates how the effect of both operations. Start with the following resource:

```
{
  "fruits" : [ "apple", "orange", "kiwi", "lime" ],
}
```



Apply the following operations on that resource:

```
[
  {
    "operation" : "remove",
    "field" : "/fruits/0",
    "value" : ""
  },
  {
    "operation" : "replace",
    "field" : "/fruits/1",
    "value" : "pineapple"
  }
]
```

The PATCH operations are applied sequentially. The **remove** operation removes the first member of that resource, based on its array index, (**fruits/0**), with the following result:

```
[
  {
    "fruits" : [ "orange", "kiwi", "lime" ],
  }
]
```

The second PATCH operation, a **replace**, is applied on the second member (**fruits/1**) of the intermediate resource, with the following result:

```
[
  {
    "fruits" : [ "orange", "pineapple", "lime" ],
  }
]
```

#### A.2.10.7. Patch Operation: Transform

The **transform** operation changes the value of a field based on a script or some other data transformation command. The following **transform** operation takes the value from the field named **/objects**, and applies the **something.js** script as shown:

```
[
  {
    "operation" : "transform",
    "field" : "/objects",
    "value" : {
      "script" : {
        "type" : "text/javascript",
        "file" : "something.js"
      }
    }
  }
]
```

### A.2.10.8. Patch Operation Limitations

Some HTTP client libraries do not support the HTTP PATCH operation. Make sure that the library you use supports HTTP PATCH before using this REST operation.

For example, the Java Development Kit HTTP client does not support PATCH as a valid HTTP method. Instead, the method `URLConnection.setRequestMethod("PATCH")` throws `ProtocolException`.

#### *Parameters*

You can use the following parameters. Other parameters might depend on the specific action implementation:

`_prettyPrint=true`

Format the body of the response.

`_fields=field[,field...]`

Return only the specified fields in the body of the response.

The `field` values are JSON pointers. For example if the resource is `{"parent":{"child":"value"}}`, `parent/child` refers to the `"child":"value"`.

### A.2.11. Action

Actions are a means of extending Common REST APIs and are defined by the resource provider, so the actions you can use depend on the implementation.

The standard action indicated by `_action=create` is described in Section A.2.6, "Create".

#### *Parameters*

You can use the following parameters. Other parameters might depend on the specific action implementation:

`_prettyPrint=true`

Format the body of the response.

`_fields=field[,field...]`

Return only the specified fields in the body of the response.

The `field` values are JSON pointers. For example if the resource is `{"parent":{"child":"value"}}`, `parent/child` refers to the `"child":"value"`.

## A.2.12. Query

To query a resource collection (or resource container if you prefer to think of it that way), perform an HTTP GET and accept a JSON response, including at least a `_queryExpression`, `_queryFilter`, or `_queryId` parameter. These parameters cannot be used together:

```
GET /users?_queryFilter=true HTTP/1.1
Host: example.com
Accept: application/json
```

The server returns the result as a JSON object including a "results" array and other fields related to the query string parameters that you specify.

### Parameters

You can use the following parameters:

#### `_queryFilter=filter-expression`

Query filters request that the server return entries that match the filter expression. You must URL-escape the filter expression.

The string representation is summarized as follows. Continue reading for additional explanation:

```
Expr           = OrExpr
OrExpr         = AndExpr ( 'or' AndExpr ) *
AndExpr        = NotExpr ( 'and' NotExpr ) *
NotExpr        = '!' PrimaryExpr | PrimaryExpr
PrimaryExpr    = '(' Expr ')' | ComparisonExpr | PresenceExpr | LiteralExpr
ComparisonExpr = Pointer OpName JsonValue
PresenceExpr   = Pointer 'pr'
LiteralExpr    = 'true' | 'false'
Pointer        = JSON pointer
OpName         = 'eq' | # equal to
                'co' | # contains
                'sw' | # starts with
                'lt' | # less than
                'le' | # less than or equal to
                'gt' | # greater than
                'ge' | # greater than or equal to
                STRING # extended operator
JsonValue      = NUMBER | BOOLEAN | '""' UTF8STRING '""'
STRING         = ASCII string not containing white-space
UTF8STRING     = UTF-8 string possibly containing white-space
```

*JsonValue* components of filter expressions follow [RFC 7159: The JavaScript Object Notation \(JSON\) Data Interchange Format](#). In particular, as described in section 7 of the RFC, the escape character in strings is the backslash character. For example, to match the identifier `test\`, use `_id eq 'test\\'`. In the JSON resource, the `\` is escaped the same way: `"_id": "test\\"`.

When using a query filter in a URL, be aware that the filter expression is part of a query string parameter. A query string parameter must be URL encoded as described in RFC 3986: *Uniform Resource Identifier (URI): Generic Syntax*. For example, white space, double quotes ("), parentheses, and exclamation characters need URL encoding in HTTP query strings. The following rules apply to URL query components:

```
query      = *( pchar / "/" / "?" )
pchar     = unreserved / pct-encoded / sub-delims / ":" / "@"
unreserved = ALPHA / DIGIT / "-" / "." / "_" / "~"
pct-encoded = "%" HEXDIG HEXDIG
sub-delims = "!" / "$" / "&" / "'" / "(" / ")"
           / "*" / "+" / "," / ";" / "="
```

ALPHA, DIGIT, and HEXDIG are core rules of RFC 5234: *Augmented BNF for Syntax Specifications*:

```
ALPHA      = %x41-5A / %x61-7A ; A-Z / a-z
DIGIT      = %x30-39 ; 0-9
HEXDIG     = DIGIT / "A" / "B" / "C" / "D" / "E" / "F"
```

As a result, a backslash escape character in a *JsonValue* component is percent-encoded in the URL query string parameter as %5C. To encode the query filter expression `_id eq 'test\\'`, use `_id+eq+'test%5C%5C'`, for example.

A simple filter expression can represent a comparison, presence, or a literal value.

For comparison expressions use *json-pointer comparator json-value*, where the *comparator* is one of the following:

- eq (equals)
- co (contains)
- sw (starts with)
- lt (less than)
- le (less than or equal to)
- gt (greater than)
- ge (greater than or equal to)

For presence, use *json-pointer pr* to match resources where the JSON pointer is present.

Literal values include true (match anything) and false (match nothing).

Complex expressions employ **and**, **or**, and **!** (not), with parentheses, (*expression*), to group expressions.

#### **`_queryId=identifier`**

Specify a query by its identifier.

Specific queries can take their own query string parameter arguments, which depend on the implementation.

### `_pagedResultsCookie=string`

The string is an opaque cookie used by the server to keep track of the position in the search results. The server returns the cookie in the JSON response as the value of `pagedResultsCookie`.

In the request `_pageSize` must also be set and non-zero. You receive the cookie value from the provider on the first request, and then supply the cookie value in subsequent requests until the server returns a `null` cookie, meaning that the final page of results has been returned.

The `_pagedResultsCookie` parameter is supported when used with the `_queryFilter` parameter. The `_pagedResultsCookie` parameter is not guaranteed to work when used with the `_queryExpression` and `_queryId` parameters.

The `_pagedResultsCookie` and `_pagedResultsOffset` parameters are mutually exclusive, and not to be used together.

### `_pagedResultsOffset=integer`

When `_pageSize` is non-zero, use this as an index in the result set indicating the first page to return.

The `_pagedResultsCookie` and `_pagedResultsOffset` parameters are mutually exclusive, and not to be used together.

### `_pageSize=integer`

Return query results in pages of this size. After the initial request, use `_pagedResultsCookie` or `_pageResultsOffset` to page through the results.

### `_totalPagedResultsPolicy=string`

When a `_pageSize` is specified, and non-zero, the server calculates the "totalPagedResults", in accordance with the `totalPagedResultsPolicy`, and provides the value as part of the response. The "totalPagedResults" is either an estimate of the total number of paged results (`_totalPagedResultsPolicy=ESTIMATE`), or the exact total result count (`_totalPagedResultsPolicy=EXACT`). If no count policy is specified in the query, or if `_totalPagedResultsPolicy=NONE`, result counting is disabled, and the server returns value of -1 for "totalPagedResults".

### `_sortKeys=[+-]field[, [+ -]field...]`

Sort the resources returned based on the specified field(s), either in `+` (ascending, default) order, or in `-` (descending) order.

The `_sortKeys` parameter is not supported for predefined queries (`_queryId`).

### `_prettyPrint=true`

Format the body of the response.

### `_fields=field[, field...]`

Return only the specified fields in each element of the "results" array in the response.

The `field` values are JSON pointers. For example if the resource is `{"parent":{"child":"value"}}`, `parent/child` refers to the `"child":"value"`.

## A.2.13. HTTP Status Codes

When working with a Common REST API over HTTP, client applications should expect at least the following HTTP status codes. Not all servers necessarily return all status codes identified here:

### **200 OK**

The request was successful and a resource returned, depending on the request.

### **201 Created**

The request succeeded and the resource was created.

### **204 No Content**

The action request succeeded, and there was no content to return.

### **304 Not Modified**

The read request included an `If-None-Match` header, and the value of the header matched the revision value of the resource.

### **400 Bad Request**

The request was malformed.

### **401 Unauthorized**

The request requires user authentication.

### **403 Forbidden**

Access was forbidden during an operation on a resource.

### **404 Not Found**

The specified resource could not be found, perhaps because it does not exist.

### **405 Method Not Allowed**

The HTTP method is not allowed for the requested resource.

### **406 Not Acceptable**

The request contains parameters that are not acceptable, such as a resource or protocol version that is not available.

### **409 Conflict**

The request would have resulted in a conflict with the current state of the resource.

### **410 Gone**

The requested resource is no longer available, and will not become available again. This can happen when resources expire for example.

**412 Precondition Failed**

The resource's current version does not match the version provided.

**415 Unsupported Media Type**

The request is in a format not supported by the requested resource for the requested method.

**428 Precondition Required**

The resource requires a version, but no version was supplied in the request.

**500 Internal Server Error**

The server encountered an unexpected condition that prevented it from fulfilling the request.

**501 Not Implemented**

The resource does not support the functionality required to fulfill the request.

**503 Service Unavailable**

The requested resource was temporarily unavailable. The service may have been disabled, for example.

## A.3. REST API Versioning

In OpenAM 12.0.0 and later, REST API features are assigned version numbers.

Providing version numbers in the REST API helps ensure compatibility between OpenAM releases. The version number of a feature increases when OpenAM introduces a non-backwards-compatible change that affects clients making use of the feature.

OpenAM provides versions for the following aspects of the REST API.

***resource***

Any changes to the structure or syntax of a returned response will incur a *resource* version change. For example changing `errorMessage` to `message` in a JSON response.

***protocol***

Any changes to the methods used to make REST API calls will incur a *protocol* version change. For example changing `_action` to `$action` in the required parameters of an API feature.

### A.3.1. Supported REST API Versions

The REST API version numbers supported in AM 5 are as follows:

***Supported protocol versions***

The *protocol* versions supported in AM 5 are:

1.0

## Supported resource versions

The *resource* versions supported in AM 5 are shown in the following table.

*Table A.1. Supported resource Versions*

Base	End Point	Supported Versions
/json	/authenticate	1.1, 2.0
	/users	1.1, 1.2, 2.0, 2.1, 3.0
	/groups	1.1, 2.0, 2.1, 3.0
	/agents	1.1, 2.0, 2.1, 3.0
	/realms	1.0
	/dashboard	1.0
	/sessions	1.1
	/serverinfo/*	1.1
	/users/{user}/devices/trusted	1.0
	/users/{user}/uma/policies	1.0
	/applications	1.0, 2.0
	/resourcetypes	1.0
	/policies	1.0, 2.0
	/applicationtypes	1.0
	/conditiontypes	1.0
	/subjecttypes	1.0
	/subjectattributes	1.0
	/decisioncombiners	1.0
	/subjectattributes	1.0
/xacml	/policies	1.0
/frrest	/token	1.0
	/client	1.0

The *OpenAM Release Notes* section, Chapter 4, "*Changes and Deprecated Functionality*" in the *Release Notes* describes the differences between API versions.

### A.3.2. Specifying an Explicit REST API Version

You can specify which version of the REST API to use by adding an **Accept-API-Version** header to the request, as in the following example, which is requesting *resource* version 2.0 and *protocol* version 1.0:



```
$ curl \
--request POST \
--header "X-OpenAM-Username: demo" \
--header "X-OpenAM-Password: changeit" \
--header "Accept-API-Version: resource=2.0, protocol=1.0" \
https://openam.example.com:8443/openam/json/realms/root/authenticate
```

You can configure the default behavior OpenAM will take when a REST call does not specify explicit version information. For more information, see [Section A.3.3, "Configuring the Default REST API Version for a Deployment"](#).

### A.3.3. Configuring the Default REST API Version for a Deployment

You can configure the default behavior OpenAM will take when a REST call does not specify explicit version information using either of the following procedures:

- Procedure A.2, "Configure Versioning Behavior by using the AM Console"
- Procedure A.3, "Configure Versioning Behavior by using the ssoadm"

The available options for default behavior are as follows:

#### ***Latest***

The latest available supported version of the API is used.

This is the preset default for new installations of OpenAM.

#### ***Oldest***

The oldest available supported version of the API is used.

This is the preset default for upgraded OpenAM instances.

#### **Note**

The oldest supported version may not be the first that was released, as APIs versions become deprecated or unsupported. See [Section 4.2, "Deprecated Functionality"](#) in the *Release Notes*.

#### ***None***

No version will be used. When a REST client application calls a REST API without specifying the version, OpenAM returns an error and the request fails.

### *Procedure A.2. Configure Versioning Behavior by using the AM Console*

1. Log in as OpenAM administrator, `amadmin`.
2. Click **Configure > Global Services**, and then click **REST APIs**.

3. In Default Version, select the required response to a REST API request that does not specify an explicit version: `Latest`, `Oldest`, or `None`.
4. Optionally, enable `Warning Header` to include warning messages in the headers of responses to requests.
5. Save your work.

### Procedure A.3. Configure Versioning Behavior by using the `ssoadm`

- Use the `ssoadm set-attr-defs` command with the `openam-rest-apis-default-version` attribute set to either `LATEST`, `OLDEST` or `NONE`, as in the following example:

```
$ ssh openam.example.com
$ cd /path/to/openam-tools/admin/openam/bin
$ ./ssoadm \
  set-attr-defs \
  --adminid amadmin \
  --password-file /tmp/pwd.txt \
  --servicename RestApisService \
  --schematype Global \
  --attributevalues openam-rest-apis-default-version=NONE

Schema attribute defaults were set.
```

### A.3.4. REST API Versioning Messages

OpenAM provides REST API version messages in the JSON response to a REST API call. You can also configure OpenAM to return version messages in the response headers.

Messages include:

- Details of the REST API versions used to service a REST API call.
- Warning messages if REST API version information is not specified or is incorrect in a REST API call.

The `resource` and `protocol` version used to service a REST API call are returned in the `Content-API-Version` header, as shown below:

```
$ curl \
-i \
--request POST \
--header "X-OpenAM-Username: demo" \
--header "X-OpenAM-Password: changeit" \
--header "Accept-API-Version: resource=2.0, protocol=1.0" \
https://openam.example.com:8443/openam/json/realms/root/authenticate

HTTP/1.1 200 OK
Content-API-Version: protocol=1.0,resource=2.0
Server: Restlet-Framework/2.1.7
Content-Type: application/json;charset=UTF-8

{
  "tokenId":"AQIC5wM...TU30Q*",
  "successUrl":"/openam/console"
}
```

If the default REST API version behavior is set to **None**, and a REST API call does not include the **Accept-API-Version** header, or does not specify a **resource** version, then a **400 Bad Request** status code is returned, as shown below:

```
$ curl \
--header "Content-Type: application/json" \
--header "Accept-API-Version: protocol=1.0" \
https://openam.example.com:8443/openam/json/realms/root/serverinfo/*

{
  "code":400,
  "reason":"Bad Request",
  "message":"No requested version specified and behavior set to NONE."
}
```

If a REST API call does include the **Accept-API-Version** header, but the specified **resource** or **protocol** version does not exist in OpenAM, then a **404 Not Found** status code is returned, as shown below:

```
$ curl \
--header "Content-Type: application/json" \
--header "Accept-API-Version: protocol=1.0, resource=999.0" \
https://openam.example.com:8443/openam/json/realms/root/serverinfo/*

{
  "code":404,
  "reason":"Not Found",
  "message":"Accept-API-Version: Requested version \"999.0\" does not match any routes."
}
```

### Tip

For more information on setting the default REST API version behavior, see Section A.3.2, "Specifying an Explicit REST API Version".

## A.4. Specifying Realms in REST API Calls

This section describes how to work with realms when making REST API calls to OpenAM.

Realms can be specified in the following ways when making a REST API call to OpenAM:

### DNS Alias

When making a REST API call, the DNS alias of a realm can be specified in the subdomain and domain name components of the REST endpoint.

To list all users in the top-level realm use the DNS alias of the OpenAM instance, for example the REST endpoint would be:

```
https://openam.example.com:8443/openam/json/users?_queryId=*
```

To list all users in a realm with DNS alias `suppliers.example.com` the REST endpoint would be:

```
https://suppliers.example.com:8443/openam/json/users?_queryId=*
```

### Path

When making a REST API call, specify the realm in the path component of the endpoint. You must specify the entire hierarchy of the realm, starting at the top-level realm. Prefix each realm in the hierarchy with the `realms/` keyword. For example `/realms/root/realms/customers/realms/europe`.

To authenticate a user in the top-level realm, use the `root` keyword. For example:

```
https://openam.example.com:8443/openam/json/realms/root/authenticate
```

To authenticate a user in a subrealm named `customers` within the top-level realm, the REST endpoint would be:

```
https://openam.example.com:8443/openam/json/realms/root/realms/customers/authenticate
```

If realms are specified using both the DNS alias and path methods, the path is used to determine the realm.

For example, the following REST endpoint returns users in a subrealm of the top-level realm named `europe`, not the realm with DNS alias `suppliers.example.com`:

```
https://suppliers.example.com:8443/openam/json/realms/root/realms/europe/users?_queryId=*
```

## A.5. Authentication and Logout

You can use REST-like APIs under `/json/authenticate` and `/json/sessions` for authentication and for logout.

The `/json/authenticate` endpoint does not support the CRUDPAQ verbs and therefore does not technically satisfy REST architectural requirements. The term *REST-like* describes this endpoint better than *REST*.

The simplest user name/password authentication returns a `tokenId` that applications can present as a cookie value for other operations that require authentication. The type of `tokenId` returned varies depending on whether stateless sessions are enabled in the realm to which the user authenticates:

- If stateless sessions are not enabled, the `tokenId` is an OpenAM SSO token.
- If stateless sessions are enabled, the `tokenId` is an OpenAM SSO token that includes an encoded OpenAM session.

Developers should be aware that the size of the `tokenId` for stateless sessions—2000 bytes or greater—is considerably longer than for stateful sessions—approximately 100 bytes. For more information about stateful and stateless session tokens, see Section 1.8.1.6, "Session Cookies" in the *Authentication and Single Sign-On Guide*.

When authenticating with a user name and password, use HTTP POST to prevent the web container from logging the credentials. Pass the user name in an `X-OpenAM-Username` header, and the password in an `X-OpenAM-Password` header:

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "X-OpenAM-Username: demo" \
--header "X-OpenAM-Password: changeit" \
--data "{}" \
https://openam.example.com:8443/openam/json/realms/root/authenticate
{
  "tokenId": "AQIC5w...NTcy*",
  "successUrl": "/openam/console"
}
```

To use UTF-8 user names and passwords in calls to the `/json/authenticate` endpoint, base64-encode the string, and then wrap the string as described in RFC 2047:

```
encoded-word = "=?" charset "?" encoding "?" encoded-text "=?"
```

For example, to authenticate using a UTF-8 username, such as `dēmø`, perform the following steps:

1. Encode the string in base64 format: `yZfDq8mxw7g=`.
2. Wrap the base64-encoded string as per RFC 2047: `=?UTF-8?B?yZfDq8mxw7g=?`.
3. Use the result in the `X-OpenAM-Username` header passed to the authentication endpoint as follows:

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "X-OpenAM-Username: =?UTF-8?B?yZfDq8mxw7g=?=" \
--header "X-OpenAM-Password: changeit" \
--data "{}" \
https://openam.example.com:8443/openam/json/realms/root/authenticate
{
  "tokenId": "AQIC5w...NTcy*",
  "successUrl": "/openam/console"
}
```

This zero page login mechanism works only for name/password authentication. If you include a POST body with the request, it must be an empty JSON string as shown in the example. Alternatively, you can leave the POST body empty. Otherwise, OpenAM interprets the body as a continuation of an existing authentication attempt, one that uses a supported callback mechanism.

The authentication service at `/json/authenticate` supports callback mechanisms that make it possible to perform other types of authentication in addition to simple user name/password login.

Callbacks that are not completed based on the content of the client HTTP request are returned in JSON as a response to the request. Each callback has an array of output suitable for displaying to the end user, and input which is what the client must complete and send back to OpenAM. The default is still user name/password authentication:

```
$ curl \
--request POST \
https://openam.example.com:8443/openam/json/realms/root/authenticate
{
  "authId": "...jwt-value...",
  "template": "",
  "stage": "DataStore1",
  "callbacks": [
    {
      "type": "NameCallback",
      "output": [
        {
          "name": "prompt",
          "value": " User Name: "
        }
      ],
      "input": [
        {
          "name": "IDToken1",
          "value": ""
        }
      ]
    },
    {
      "type": "PasswordCallback",
      "output": [
        {
          "name": "prompt",
          "value": " Password: "
        }
      ]
    }
  ]
}
```

```
    ],
    "input": [
      {
        "name": "IDToken2",
        "value": ""
      }
    ]
  }
}
```

The `authID` value is a JSON Web Token (JWT) that uniquely identifies the authentication context to OpenAM, and so must also be sent back with the requests.

To respond to the callback, send back the JSON object with the missing values filled, as in this case where the user name is `demo` and the password is `changeit`:

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--data '{ "authId": "...jwt-value...", "template": "", "stage": "DataStore1",
"callbacks": [ { "type": "NameCallback", "output": [ { "name": "prompt",
"value": " User Name: " } ] }, "input": [ { "name": "IDToken1", "value": "demo" } ] },
{ "type": "PasswordCallback", "output": [ { "name": "prompt", "value": " Password: " } ] },
"input": [ { "name": "IDToken2", "value": "changeit" } ] } ] }' \
https://openam.example.com:8443/openam/json/realms/root/authenticate

{ "tokenId": "AQIC5wM2...U3MTE4NA..*", "successUrl": "/openam/console" }
```

The response is a token ID holding the SSO token value.

Alternatively, you can authenticate without requesting a session using the `noSession` query string parameter:

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--data '{ "authId": "...jwt-value...", "template": "", "stage": "DataStore1",
"callbacks": [ { "type": "NameCallback", "output": [ { "name": "prompt",
"value": " User Name: " } ] }, "input": [ { "name": "IDToken1", "value": "demo" } ] },
{ "type": "PasswordCallback", "output": [ { "name": "prompt", "value": " Password: " } ] },
"input": [ { "name": "IDToken2", "value": "changeit" } ] } ] }' \
https://openam.example.com:8443/openam/json/realms/root/authenticate?noSession=true

{ "message": "Authentication Successful", "successUrl": "/openam/console" }
```

OpenAM can be configured to return a failure URL value when authentication fails. No failure URL is configured by default. The Default Failure Login URL can be set per realm; see [Section 11.1.7, "Post Authentication Processing"](#) in the *Authentication and Single Sign-On Guide* for details.

Alternatively, failure URLs can be configured per authentication chain, which your client can specify using the `service` parameter described below. On failure OpenAM then returns HTTP status code 401 Unauthorized, and the JSON in the reply indicates the failure URL:

```
$ curl \
--request POST \
--header "X-OpenAM-Username: demo" \
--header "X-OpenAM-Password: badpassword" \
https://openam.example.com:8443/openam/json/realms/root/authenticate
{
  "code":401,
  "reason":"Unauthorized",
  "message":"Invalid Password!!",
  "failureUrl": "http://www.example.com/401.html"
}
```

When making a REST API call, specify the realm in the path component of the endpoint. You must specify the entire hierarchy of the realm, starting at the top-level realm. Prefix each realm in the hierarchy with the `realms/` keyword. For example `/realms/root/realms/customers/realms/europe`.

For example, to authenticate to a subrealm `customers` within the top-level realm, then the authentication endpoint URL is as follows: `https://openam.example.com:8443/openam/json/realms/root/realms/customers/authenticate`

The following additional parameters are supported:

You can use the `authIndexType` and `authIndexValue` query string parameters as a pair to provide additional information about how you are authenticating. The `authIndexType` can be one of the following types:

#### **composite**

Set the value to a composite advice string.

#### **level**

Set the value to the authentication level.

#### **module**

Set the value to the name of an authentication module.

#### **resource**

Set the value to a URL protected by an OpenAM policy.

#### **role**

Set the value to an OpenAM role.

#### **service**

Set the value to the name of an authentication chain.

#### **user**

Set the value to an OpenAM user ID.

You can use the query string parameter, `sessionUpgradeSS0TokenId=tokenId`, to request session upgrade. Before the `tokenId` is searched for in the query string for session upgrade, the token is grabbed



from the cookie. For an explanation of session upgrade, see [Section 1.8.2, "Session Upgrade"](#) in the *Authentication and Single Sign-On Guide*.

OpenAM uses the following callback types depending on the authentication module in use:

- **ChoiceCallback**: Used to display a list of choices and retrieve the selected choice.
- **ConfirmationCallback**: Used to ask for a confirmation such as Yes, No, or Cancel and retrieve the selection.
- **HiddenValueCallback**: Used to return form values that are not visually rendered to the end user.
- **HttpCallback**: Used for HTTP handshake negotiations.
- **LanguageCallback**: Used to retrieve the locale for localizing text presented to the end user.
- **NameCallback**: Used to retrieve a name string.
- **PasswordCallback**: Used to retrieve a password value.
- **RedirectCallback**: Used to redirect the client user-agent.
- **ScriptTextOutputCallback**: Used to insert a script into the page presented to the end user. The script can, for example, collect data about the user's environment.
- **TextInputCallback**: Used to retrieve text input from the end user.
- **TextOutputCallback**: Used to display a message to the end user.
- **X509CertificateCallback**: Used to retrieve the content of an x.509 certificate.

### A.5.1. Logout

Authenticated users can log out with the token cookie value and an HTTP POST to [/json/sessions/?\\_action=logout](#):

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "Cache-Control: no-cache" \
--header "iplanetDirectoryPro: AQIC5wM2...U3MTE4NA..*" \
https://openam.example.com:8443/openam/json/realms/root/sessions/?_action=logout

{"result":"Successfully logged out"}
```

### A.5.2. logoutByHandle

To log out a session using a session handle, first perform an HTTP GET to the resource URL, [/json/sessions/](#), using the **queryFilter** action to get the session handle:

```
$ curl \
--request GET \
--header "Content-Type: application/json" \
--header "Cache-Control: no-cache" \
--header "iPlanetDirectoryPro: AQICS...NzEz*" \
http://openam.example.com:8080/openam/json/realms/root/sessions?_queryFilter=username%20eq%20%22demo%22%20and%20realm%20eq%20%22%2F%22
{
  "result": [
    {
      "username": "demo",
      "universalId": "id=demo,ou=user,dc=openam,dc=forgerock,dc=org",
      "realm": "\\",
      "sessionHandle": "shandle:AQIC5w...MTY3*",
      "latestAccessTime": "2016-11-09T14:14:11Z",
      "maxIdleExpirationTime": "2016-11-09T14:44:11Z",
      "maxSessionExpirationTime": "2016-11-09T16:14:11Z"
    }
  ],
  "resultCount": 1,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

To log out a session using a session handle, perform an HTTP POST to the resource URL, `/json/sessions/`, using the `logoutByHandle` action.

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "Cache-Control: no-cache" \
--header "iPlanetDirectoryPro: AQIC5w...NTcy*" \
--data '{"sessionHandles": ["shandle:AQIC5w...MTY3*", "shandle:AQIC5w...NDcx*"]}' \
http://openam.example.com:8080/openam/json/realms/root/sessions/?_action=logoutByHandle
{
  "result": {
    "shandle:AQIC5w...NDcx*": true,
    "shandle:AQIC5w...MTY3*": true
  }
}
```

### A.5.3. Load Balancer and Proxy Layer Requirements

When authentication depends on the client IP address and OpenAM lies behind a load balancer or proxy layer, configure the load balancer or proxy to send the address by using the `X-Forwarded-For` header, and configure OpenAM to consume and forward the header as necessary. For details, see Section 2.2.4, "Handling HTTP Request Headers" in the *Installation Guide*.

## A.5.4. Windows Desktop SSO Requirements

When authenticating with Windows Desktop SSO, add an **Authorization** header containing the string **Basic**, followed by a base64-encoded string of the username, a colon character, and the password. In the following example, the credentials **demo:changeit** are base64-encoded into the string **ZGVtbzpjagFuZ2VpdA==**:

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "X-OpenAM-Username: demo" \
--header "X-OpenAM-Password: changeit" \
--header "Authorization: Basic ZGVtbzpjagFuZ2VpdA==" \
--data "{}" \
https://openam.example.com:8443/openam/json/realms/root/authenticate
{ "tokenId": "AQIC5w...NTcy*", "successUrl": "/openam/console" }
```

## A.6. Using the Session Token After Authentication

The following is a common scenario when accessing OpenAM by using REST API calls:

- First, call the **/json/authenticate** endpoint to log a user in to OpenAM. This REST API call returns a **tokenId** value, which is used in subsequent REST API calls to identify the user:

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "X-OpenAM-Username: demo" \
--header "X-OpenAM-Password: changeit" \
--data "{}" \
https://openam.example.com:8443/openam/json/realms/root/authenticate
{ "tokenId": "AQIC5w...NTcy*", "successUrl": "/openam/console" }
```

The returned **tokenId** is known as a session token (also referred to as an SSO token). REST API calls made after successful authentication to OpenAM must present the session token in the HTTP header as proof of authentication.

- Next, call one or more additional REST APIs on behalf of the logged-in user. Each REST API call passes the user's **tokenId** back to OpenAM in the HTTP header as proof of previous authentication.

The following is a *partial* example of a **curl** command that inserts the token ID returned from a prior successful OpenAM authentication attempt into the HTTP header:

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "iPlanetDirectoryPro: AQIC5w...NTcy*" \
--data '{
  ...
}
```

Observe that the session token is inserted into a header field named `iPlanetDirectoryPro`. This header field name must correspond to the name of the OpenAM session cookie—by default, `iPlanetDirectoryPro`. You can find the cookie name in the AM console by navigating to Deployment > Servers > *Server Name* > Security > Cookie, in the Cookie Name field of the AM console.

Once a user has authenticated, it is *not* necessary to insert login credentials in the HTTP header in subsequent REST API calls. Note the absence of `X-OpenAM-Username` and `X-OpenAM-Password` headers in the preceding example.

Users are required to have appropriate privileges in order to access OpenAM functionality using the REST API. For example, users who lack administrative privileges cannot create OpenAM realms. For more information on the OpenAM privilege model, see Section 2.4.1, "Delegating Realm Administration Privileges".

- Finally, call the REST API to log the user out of OpenAM as described in Section A.5, "Authentication and Logout". As with other REST API calls made after a user has authenticated, the REST API call to log out of OpenAM requires the user's `tokenId` in the HTTP header.

## A.7. Server Information

You can retrieve OpenAM server information by using HTTP GET on `/json/serverinfo/*` as follows:

```
$ curl https://openam.example.com:8443/openam/json/serverinfo/*
{
  "domains": [
    ".example.com"
  ],
  "protectedUserAttributes": [],
  "cookieName": "iPlanetDirectoryPro",
  "secureCookie": false,
  "forgotPassword": "false",
  "forgotUsername": "false",
  "kbaEnabled": "false",
  "selfRegistration": "false",
  "lang": "en-US",
  "successfulUserRegistrationDestination": "default",
  "socialImplementations": [
    {
      "iconPath": "XUI/images/logos/facebook.png",
      "authnChain": "FacebookSocialAuthenticationService",
    }
  ]
}
```

```
        "displayName": "Facebook",
        "valid": true
    }
},
"referralsEnabled": "false",
"zeroPageLogin": {
    "enabled": false,
    "referrerWhitelist": [
        ""
    ],
    "allowedWithoutReferer": true
},
"realm": "/",
"xuiUserSessionValidationEnabled": true,
"FQDN": "openam.example.com"
}
```

## A.8. Token Encoding

Valid tokens in OpenAM requires configuration either in percent encoding or in *C66Encode* format. C66Encode format is encouraged. It is the default token format for OpenAM, and is used in this section. The following is an example token that has not been encoded:

```
AQIC5wM2LY4SfczntBbXvEA0uECbqMY3J4NW3byH6xwgkGE=@AAJTSQACMDE=#
```

This token includes reserved characters such as `+`, `/`, and `=` (The `@`, `#`, and `*` are not reserved characters per se, but substitutions are still required). To c66encode this token, you would substitute certain characters for others, as follows:

- `+` is replaced with `-`
- `/` is replaced with `_`
- `=` is replaced with `.`
- `@` is replaced with `*`
- `#` is replaced with `*`
- `*` (first instance) is replaced with `@`
- `*` (subsequent instances) is replaced with `#`

In this case, the translated token would appear as shown here:

```
AQIC5wM2LY4SfczntBbXvEA0uECbqMY3J4NW3byH6xwgkGE.*AAJTSQACMDE.*
```

## A.9. Logging

AM 5 supports two Audit Logging Services: a new common REST-based Audit Logging Service, and the legacy Logging Service, which is based on a Java SDK and is available in OpenAM versions prior to OpenAM 13. The legacy Logging Service is deprecated.

Both audit facilities log OpenAM REST API calls.

### A.9.1. Common Audit Logging of REST API Calls

OpenAM logs information about all REST API calls to the **access** topic. For more information about OpenAM audit topics, see Section 6.1.2, "Audit Log Topics".

Locate specific REST endpoints in the **http.path** log file property.

### A.9.2. Legacy Logging of REST API Calls

OpenAM logs information about REST API calls to two files:

- **amRest.access**. Records accesses to a CREST endpoint, regardless of whether the request successfully reached the endpoint through policy authorization.

An **amRest.access** example is as follows:

```
$ cat openam/openam/log/amRest.access

#Version: 1.0
#Fields: time Data LoginID ContextID IPAddr LogLevel Domain LoggedBy MessageID ModuleName
NameID HostName
"2011-09-14 16:38:17" /home/user/openam/openam/log/ "cn=dsameuser,ou=DSAME Users,o=openam"
aa307b2dcb721d4201 "Not Available" INFO o=openam "cn=dsameuser,ou=DSAME Users,o=openam"
LOG-1 amRest.access "Not Available" 192.168.56.2
"2011-09-14 16:38:17" "Hello World" id=bjensen,ou=user,o=openam 8a4025a2b3af291d01 "Not Available"
INFO o=openam id=amadmin,ou=user,o=openam "Not Available" amRest.access "Not Available"
192.168.56.2
```

- **amRest.authz**. Records all CREST authorization results regardless of success. If a request has an entry in the **amRest.access** log, but no corresponding entry in **amRest.authz**, then that endpoint was not protected by an authorization filter and therefore the request was granted access to the resource.

The **amRest.authz** file contains the **Data** field, which specifies the authorization decision, resource, and type of action performed on that resource. The **Data** field has the following syntax:

```

("GRANT"|"DENY") > "RESOURCE | ACTION"

where
  "GRANT" > " is prepended to the entry if the request was allowed
  "DENY" > " is prepended to the entry if the request was not allowed
  "RESOURCE" is "ResourceLocation | ResourceParameter"
    where
      "ResourceLocation" is the endpoint location (e.g., subrealm/applicationtypes)
      "ResourceParameter" is the ID of the resource being touched
        (e.g., myApplicationType) if applicable. Otherwise, this field is empty
        if touching the resource itself, such as in a query.

  "ACTION" is "ActionType | ActionParameter"
    where
      "ActionType" is "CREATE||READ||UPDATE||DELETE||PATCH||ACTION||QUERY"
      "ActionParameter" is one of the following depending on the ActionType:
        For CREATE: the new resource ID
        For READ: empty
        For UPDATE: the revision of the resource to update
        For DELETE: the revision of the resource to delete
        For PATCH: the revision of the resource to patch
        For ACTION: the actual action performed (e.g., "forgotPassword")
        For QUERY: the query ID if any

```

```
$ cat openam/openam/log/amRest.authz
```

```

#Version: 1.0
#Fields: time Data ContextID LoginID IPAddr LogLevel Domain MessageID LoggedBy NameID
ModuleName HostName
"2014-09-16 14:17:28" /var/root/openam/openam/log/ 7d3af9e799b6393301
"cn=dsameuser,ou=DSAME Users,dc=openam,dc=forgerock,dc=org" "Not Available" INFO
dc=openam,dc=forgerock,dc=org LOG-1 "cn=dsameuser,ou=DSAME Users,dc=openam,dc=forgerock,dc=org"
"Not Available" amRest.authz 10.0.1.5
"2014-09-16 15:56:12" "GRANT > sessions|ACTION|logout|AdminOnlyFilter" d3977a55a2ee18c201
id=amadmin,ou=user,dc=openam,dc=forgerock,dc=org "Not Available" INFO dc=openam,dc=forgerock,dc=org
OAuth2Provider-2 "cn=dsameuser,ou=DSAME Users,dc=openam,dc=forgerock,dc=org" "Not Available"
amRest.authz 127.0.0.1
"2014-09-16 15:56:40" "GRANT > sessions|ACTION|logout|AdminOnlyFilter" eedbc205bf51780001
id=amadmin,ou=user,dc=openam,dc=forgerock,dc=org "Not Available" INFO dc=openam,dc=forgerock,dc=org
OAuth2Provider-2 "cn=dsameuser,ou=DSAME Users,dc=openam,dc=forgerock,dc=org" "Not Available"
amRest.authz 127.0.0.1

```

OpenAM also provides additional information in its debug notifications for accesses to any endpoint, depending on the message type (error, warning or message) including realm, user, and result of the operation.

## A.10. Reference

This reference section covers return codes and system settings relating to REST API support in OpenAM.

## A.10.1. REST APIs

**ssoadm** service name: `rest`

The following settings are available in this service:

### Default Resource Version

The API resource version to use when the REST request does not specify an explicit version. Choose from:

- `Latest`. If an explicit version is not specified, the latest resource version of an API is used.
- `Oldest`. If an explicit version is not specified, the oldest supported resource version of an API is used. Note that since APIs may be deprecated and fall out of support, the oldest *supported* version may not be the first version.
- `None`. If an explicit version is not specified, the request will not be handled and an error status is returned.

The possible values for this property are:

```
Latest
Oldest
None
```

Default value: `Latest`

**ssoadm** attribute: `defaultVersion`

### Warning Header

Whether to include a warning header in the response to a request which fails to include the `Accept-API-Version` header.

Default value: `false`

**ssoadm** attribute: `warningHeader`

### API Descriptions

Whether API Explorer and API Docs are enabled in OpenAM and how the documentation for them is generated. Dynamic generation includes descriptions from any custom services and authentication modules you may have added. Static generation only includes services and authentication modules that were present when OpenAM was built. Note that dynamic documentation generation may not work in some application containers.

The possible values for this property are:

```
DYNAMIC
STATIC
DISABLED
```



Default value: `STATIC`

**ssoadm** attribute: `descriptionsState`

## Default Protocol Version

The API protocol version to use when a REST request does not specify an explicit version. Choose from:

- `Oldest`. If an explicit version is not specified, the oldest protocol version is used.
- `Latest`. If an explicit version is not specified, the latest protocol version is used.
- `None`. If an explicit version is not specified, the request will not be handled and an error status is returned.

The possible values for this property are:

```
Oldest
Latest
None
```

Default value: `Latest`

**ssoadm** attribute: `defaultProtocolVersion`

## Appendix B. Getting Support

For more information or resources about OpenAM and ForgeRock Support, see the following sections:

### B.1. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

- The ForgeRock [Knowledge Base](#) offers a large and increasing number of up-to-date, practical articles that help you deploy and manage ForgeRock software.
- ForgeRock core documentation, such as this document, aims to be technically accurate and complete with respect to the software documented. It is visible to everyone and covers all product features and examples of how to use them.

Core documentation therefore follows a three-phase review process designed to eliminate errors:

- Product managers and software architects review project documentation design with respect to the readers' software lifecycle needs.
- Subject matter experts review proposed documentation changes for technical accuracy and completeness with respect to the corresponding software.
- Quality experts validate implemented documentation changes for technical accuracy, completeness in scope, and usability for the readership.

The review process helps to ensure that documentation published for a ForgeRock release is technically accurate and complete.

Fully reviewed, published core documentation is available at <http://backstage.forgerock.com/>. Use this documentation when working with a ForgeRock Identity Platform release.

## B.2. Joining the ForgeRock Community

Visit the [Community resource center](#) where you can find information about each project, download trial builds, browse the resource catalog, ask and answer questions on the forums, find community events near you, and find the source code for open source software.

## B.3. Getting Support and Contacting ForgeRock

ForgeRock provides support services, professional services, classes through ForgeRock University, and partner services to assist you in setting up and maintaining your deployments. For a general overview of these services, see <https://www.forgerock.com>.

ForgeRock has staff members around the globe who support our international customers and partners. For details, visit <https://www.forgerock.com>, or send an email to ForgeRock at [info@forgerock.com](mailto:info@forgerock.com).

# Glossary

Access control	Control to grant or to deny access to a resource.
Account lockout	The act of making an account temporarily or permanently inactive after successive authentication failures.
Actions	Defined as part of policies, these verbs indicate what authorized subjects can do to resources.
Advice	In the context of a policy decision denying access, a hint to the policy enforcement point about remedial action to take that could result in a decision allowing access.
Agent administrator	User having privileges only to read and write policy agent profile configuration information, typically created to delegate policy agent profile creation to the user installing a policy agent.
Agent authenticator	Entity with read-only access to multiple agent profiles defined in the same realm; allows an agent to read web service profiles.
Application	<p>In general terms, a service exposing protected resources.</p> <p>In the context of OpenAM policies, the application is a template that constrains the policies that govern access to protected resources. An application can have zero or more policies.</p>
Application type	<p>Application types act as templates for creating policy applications.</p> <p>Application types define a preset list of actions and functional logic, such as policy lookup and resource comparator logic.</p>

	Application types also define the internal normalization, indexing logic, and comparator logic for applications.
Attribute-based access control (ABAC)	Access control that is based on attributes of a user, such as how old a user is or whether the user is a paying customer.
Authentication	The act of confirming the identity of a principal.
Authentication chaining	A series of authentication modules configured together which a principal must negotiate as configured in order to authenticate successfully.
Authentication level	Positive integer associated with an authentication module, usually used to require success with more stringent authentication measures when requesting resources requiring special protection.
Authentication module	OpenAM authentication unit that handles one way of obtaining and verifying credentials.
Authorization	The act of determining whether to grant or to deny a principal access to a resource.
Authorization Server	In OAuth 2.0, issues access tokens to the client after authenticating a resource owner and confirming that the owner authorizes the client to access the protected resource. OpenAM can play this role in the OAuth 2.0 authorization framework.
Auto-federation	Arrangement to federate a principal's identity automatically based on a common attribute value shared across the principal's profiles at different providers.
Bulk federation	Batch job permanently federating user profiles between a service provider and an identity provider based on a list of matched user identifiers that exist on both providers.
Circle of trust	Group of providers, including at least one identity provider, who have agreed to trust each other to participate in a SAML v2.0 provider federation.
Client	In OAuth 2.0, requests protected web resources on behalf of the resource owner given the owner's authorization. OpenAM can play this role in the OAuth 2.0 authorization framework.
Conditions	Defined as part of policies, these determine the circumstances under which which a policy applies.  Environmental conditions reflect circumstances like the client IP address, time of day, how the subject authenticated, or the authentication level achieved.

	Subject conditions reflect characteristics of the subject like whether the subject authenticated, the identity of the subject, or claims in the subject's JWT.
Configuration datastore	LDAP directory service holding OpenAM configuration data.
Cross-domain single sign-on (CDSSO)	OpenAM capability allowing single sign-on across different DNS domains.
Delegation	Granting users administrative privileges with OpenAM.
Entitlement	Decision that defines which resource names can and cannot be accessed for a given subject in the context of a particular application, which actions are allowed and which are denied, and any related advice and attributes.
Extended metadata	Federation configuration information specific to OpenAM.
Extensible Access Control Markup Language (XACML)	Standard, XML-based access control policy language, including a processing model for making authorization decisions based on policies.
Federation	Standardized means for aggregating identities, sharing authentication and authorization data information between trusted providers, and allowing principals to access services across different providers without authenticating repeatedly.
Fedlet	Service provider application capable of participating in a circle of trust and allowing federation without installing all of OpenAM on the service provider side; OpenAM lets you create Java Fedlets.
Hot swappable	Refers to configuration properties for which changes can take effect without restarting the container where OpenAM runs.
Identity	Set of data that uniquely describes a person or a thing such as a device or an application.
Identity federation	Linking of a principal's identity across multiple providers.
Identity provider (IdP)	Entity that produces assertions about a principal (such as how and when a principal authenticated, or that the principal's profile has a specified attribute value).
Identity repository	Data store holding user profiles and group information; different identity repositories can be defined for different realms.
Java EE policy agent	Java web application installed in a web container that acts as a policy agent, filtering requests to other applications in the container with policies based on application resource URLs.

Metadata	Federation configuration information for a provider.
Policy	Set of rules that define who is granted access to a protected resource when, how, and under what conditions.
Policy Agent	Agent that intercepts requests for resources, directs principals to OpenAM for authentication, and enforces policy decisions from OpenAM.
Policy Administration Point (PAP)	Entity that manages and stores policy definitions.
Policy Decision Point (PDP)	Entity that evaluates access rights and then issues authorization decisions.
Policy Enforcement Point (PEP)	Entity that intercepts a request for a resource and then enforces policy decisions from a PDP.
Policy Information Point (PIP)	Entity that provides extra information, such as user profile attributes that a PDP needs in order to make a decision.
Principal	<p>Represents an entity that has been authenticated (such as a user, a device, or an application), and thus is distinguished from other entities.</p> <p>When a <b>Subject</b> successfully authenticates, OpenAM associates the Subject with the Principal.</p>
Privilege	In the context of delegated administration, a set of administrative tasks that can be performed by specified subjects in a given realm.
Provider federation	Agreement among providers to participate in a circle of trust.
Realm	<p>OpenAM unit for organizing configuration and identity information.</p> <p>Realms can be used for example when different parts of an organization have different applications and user data stores, and when different organizations use the same OpenAM deployment.</p> <p>Administrators can delegate realm administration. The administrator assigns administrative privileges to users, allowing them to perform administrative tasks within the realm.</p>
Resource	<p>Something a user can access over the network such as a web page.</p> <p>Defined as part of policies, these can include wildcards in order to match multiple actual resources.</p>
Resource owner	In OAuth 2.0, entity who can authorize access to protected web resources, such as an end user.

Resource server	In OAuth 2.0, server hosting protected web resources, capable of handling access tokens to respond to requests for such resources.
Response attributes	Defined as part of policies, these allow OpenAM to return additional information in the form of "attributes" with the response to a policy decision.
Role based access control (RBAC)	Access control that is based on whether a user has been granted a set of permissions (a role).
Security Assertion Markup Language (SAML)	Standard, XML-based language for exchanging authentication and authorization data between identity providers and service providers.
Service provider (SP)	Entity that consumes assertions about a principal (and provides a service that the principal is trying to access).
Session	The interval that starts with the user authenticating through OpenAM and ends when the user logs out, or when their session is terminated. For browser-based clients, OpenAM manages user sessions across one or more applications by setting a session cookie. See also <a href="#">Stateful session</a> and <a href="#">Stateless session</a> .
Session high availability	Capability that lets any OpenAM server in a clustered deployment access shared, persistent information about users' sessions from the CTS token store. The user does not need to log in again unless the entire deployment goes down.
Session token	Unique identifier issued by OpenAM after successful authentication. For a <a href="#">Stateful session</a> , the session token is used to track a principal's session.
Single log out (SLO)	Capability allowing a principal to end a session once, thereby ending her session across multiple applications.
Single sign-on (SSO)	Capability allowing a principal to authenticate once and gain access to multiple applications without authenticating again.
Site	<p>Group of OpenAM servers configured the same way, accessed through a load balancer layer.</p> <p>The load balancer handles failover to provide service-level availability. Use sticky load balancing based on <code>amlbcookie</code> values to improve site performance.</p> <p>The load balancer can also be used to protect OpenAM services.</p>
Standard metadata	Standard federation configuration information that you can share with other access management software.
Stateful session	An OpenAM session that resides in the Core Token Service's token store. Stateful sessions might also be cached in memory on one or



more OpenAM servers. OpenAM tracks stateful sessions in order to handle events like logout and timeout, to permit session constraints, and to notify applications involved in SSO when a session ends.

Stateless session

An OpenAM session for which state information is encoded in OpenAM and stored on the client. The information from the session is not retained in the CTS token store. For browser-based clients, OpenAM sets a cookie in the browser that contains the session information.

Subject

Entity that requests access to a resource

When a subject successfully authenticates, OpenAM associates the subject with the [Principal](#) that distinguishes it from other subjects. A subject can be associated with multiple principals.

User data store

Data storage service holding principals' profiles; underlying storage can be an LDAP directory service, a relational database, or a custom [IdRepo](#) implementation.

Web policy agent

Native library installed in a web server that acts as a policy agent with policies based on web page URLs.