



# OpenAM Administration Guide

Version 12.0.0

Mark Craig  
David Goldsmith  
Gene Hirayama  
Mike Jang  
Chris Lee  
Vanessa Richie

ForgeRock AS  
33 New Montgomery St.,  
Suite 1500  
San Francisco, CA 94105, USA  
+1 415-599-1100 (US)  
[www.forgerock.com](http://www.forgerock.com)

---

Copyright © 2011-2014 ForgeRock, Inc.

## Abstract

Guide to configuring and using OpenAM features. OpenAM provides open source Authentication, Authorization, Entitlement and Federation software.



This work is licensed under the [Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License](http://creativecommons.org/licenses/by-nc-nd/3.0/).

To view a copy of this license, visit <http://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™ is the trademark 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.

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.

---

---

# Table of Contents

|  |      |
|--|------|
| Preface .....  | vii  |
| 1. Who Should Use this Guide .....                                 | vii  |
| 2. Formatting Conventions .....                                    | viii |
| 3. Accessing Documentation Online .....                            | ix   |
| 4. Joining the ForgeRock Community .....                           | ix   |
| 1. Administration Interfaces & Tools .....                         | 1    |
| 1.1. OpenAM Web-Based Console .....                                | 1    |
| 1.2. OpenAM Command-Line Tools .....                               | 3    |
| 1.3. OpenAM ssoadm.jsp .....                                       | 5    |
| 2. Defining Authentication Services .....                          | 7    |
| 2.1. About Authentication in OpenAM .....                          | 7    |
| 2.2. Configuring Social Authentication .....                       | 9    |
| 2.3. Configuring Authentication Modules .....                      | 17   |
| 2.4. Configuring Authentication Chains .....                       | 80   |
| 2.5. Post Authentication Plugins .....                             | 82   |
| 2.6. Authenticating To OpenAM .....                                | 83   |
| 2.7. Authentication Levels & Session Upgrade .....                 | 86   |
| 2.8. Configuring Account Lockout .....                             | 87   |
| 2.9. Configuring Session Quotas .....                              | 88   |
| 2.10. Configuring Valid goto URL Resources .....                   | 90   |
| 3. Defining Authorization Policies .....                           | 93   |
| 3.1. About Authorization in OpenAM .....                           | 93   |
| 3.2. How OpenAM Reaches Policy Decisions .....                     | 97   |
| 3.3. Configuring Applications, Policies, and Referrals .....       | 98   |
| 3.4. Delegating Policy Management .....                            | 117  |
| 4. Configuring Realms .....  | 119  |
| 4.1. Managing Realms .....   | 120  |
| 4.2. Working With Realms and Policy Agents .....                   | 123  |
| 4.3. Configuring Data Stores .....                                 | 124  |
| 5. Configuring Policy Agent Profiles .....                         | 181  |
| 5.1. Open Identity Gateway or Policy Agent? .....                  | 181  |
| 5.2. Kinds of Agent Profiles .....                                 | 182  |
| 5.3. Creating Agent Profiles .....                                 | 183  |
| 5.4. Delegating Agent Profile Creation .....                       | 185  |
| 5.5. Configuring Web Policy Agents .....                           | 186  |
| 5.6. Configuring Java EE Policy Agents .....                       | 210  |
| 5.7. Configuring Web Service Provider Policy Agents .....          | 236  |
| 5.8. Configuring Web Service Client Policy Agents .....            | 240  |
| 5.9. Configuring Security Token Service Client Policy Agents ..... | 243  |
| 5.10. Configuring Version 2.2 Policy Agents .....                  | 247  |
| 5.11. Configuring OAuth 2.0 & OpenID Connect 1.0 Clients .....     | 247  |
| 5.12. Configuring Agent Authenticators .....                       | 251  |
| 6. Configuring Audit Logging .....                                 | 253  |

|  |     |
|--|-----|
| 6.1. Configuring Audit Logging .....                                     | 253 |
| 6.2. Audit Logging to Flat Files .....                                   | 254 |
| 6.3. Audit Logging to a Syslog Server .....                              | 254 |
| 6.4. Audit Logging in OpenAM Policy Agents .....                         | 255 |
| 7. Working with Mobile Devices & Applications .....                      | 257 |
| 7.1. Simplifying Access on Mobile Devices .....                          | 257 |
| 7.2. Protecting Access for Mobile Users .....                            | 260 |
| 7.3. Simplifying Access with REST APIs .....                             | 261 |
| 7.4. Getting Source Code for Sample Mobile Applications .....            | 263 |
| 8. Configuring User Self-Service Features .....                          | 265 |
| 8.1. Configuring User Self-Registration .....                            | 265 |
| 8.2. About Password Reset .....  | 266 |
| 8.3. Resetting Forgotten Passwords (Legacy) .....                        | 267 |
| 9. Configuring Single Sign-On within One Domain .....                    | 275 |
| 9.1. The Basics of the HTTP Cookie .....                                 | 275 |
| 9.2. Cookies and the SSO Session Process .....                           | 276 |
| 9.3. Potential Problems .....  | 277 |
| 9.4. Configure SSO on One Domain .....                                   | 278 |
| 10. Configuring Cross-Domain Single Sign On .....                        | 281 |
| 11. Managing SAML 2.0 Federation .....                                   | 289 |
| 11.1. About SAML 2.0 SSO & Federation .....                              | 289 |
| 11.2. Setting Up SAML 2.0 SSO .....                                      | 290 |
| 11.3. Configuring Identity Providers .....                               | 301 |
| 11.4. Configuring Service Providers .....                                | 305 |
| 11.5. Configuring Circles of Trust .....                                 | 310 |
| 11.6. Configuring Providers for Failover .....                           | 311 |
| 11.7. Configuring Google Apps as a Remote Service Provider .....         | 312 |
| 11.8. Configuring Salesforce CRM as a Remote Service Provider .....      | 313 |
| 11.9. Using SAML 2.0 Single Sign-On & Single Logout .....                | 317 |
| 11.10. Configuring OpenAM For the ECP Profile .....                      | 323 |
| 11.11. Managing Federated Accounts .....                                 | 324 |
| 11.12. Using SAML 2.0 Artifacts .....                                    | 334 |
| 11.13. SAML 2.0 & Policy Agents .....                                    | 334 |
| 12. Managing OAuth 2.0 Authorization .....                               | 337 |
| 12.1. About OAuth 2.0 Support in OpenAM .....                            | 337 |
| 12.2. Configuring the OAuth 2.0 Authorization Service .....              | 346 |
| 12.3. Registering OAuth 2.0 Clients With the Authorization Service ..... | 349 |
| 12.4. Managing OAuth 2.0 Tokens .....                                    | 350 |
| 12.5. Security Considerations .....                                      | 361 |
| 13. Managing OpenID Connect 1.0 Authorization .....                      | 363 |
| 13.1. About OpenID Connect 1.0 Support in OpenAM .....                   | 364 |
| 13.2. Configuring OpenAM As OpenID Provider .....                        | 367 |
| 13.3. Configuring OpenAM For OpenID Connect Discovery .....              | 368 |
| 13.4. Registering OpenID Connect Relying Parties .....                   | 370 |
| 13.5. Managing User Sessions .....                                       | 372 |
| 13.6. Relying Party Examples .....                                       | 372 |

|  |     |
|--|-----|
| 13.7. Security Considerations .....                      | 379 |
| 13.8. Using OpenAM with Mobile Connect .....             | 379 |
| 14. Managing SAML 1.x Single Sign-On .....               | 385 |
| 14.1. About SAML 1.x .....                               | 385 |
| 14.2. Gathering Configuration Information .....          | 388 |
| 14.3. Preparing To Secure SAML 1.x Communications .....  | 391 |
| 14.4. Configuring SAML 1.x For Your Site .....           | 391 |
| 14.5. Configuring SAML 1.x Trusted Partners .....        | 394 |
| 14.6. Testing SAML 1.x Web SSO .....                     | 396 |
| 15. The RESTful Security Token Service .....             | 401 |
| 15.1. About the REST-STS .....                           | 401 |
| 15.2. Configuring the Security Token Service .....       | 402 |
| 16. Configuring the Dashboard Service .....              | 409 |
| 16.1. About the Dashboard Service .....                  | 409 |
| 16.2. Setting Up the Dashboard Service .....             | 410 |
| 16.3. Configuring Dashboard Service for a Realm .....    | 411 |
| 16.4. Adding Applications to a User's Dashboard .....    | 411 |
| 17. Configuring REST APIs .....                          | 413 |
| 18. Backing Up and Restoring OpenAM Configurations ..... | 417 |
| 19. Managing Certificates .....                          | 423 |
| 20. Monitoring OpenAM Services .....                     | 431 |
| 20.1. Monitoring Interfaces .....                        | 431 |
| 20.2. Monitoring CTS Tokens .....                        | 435 |
| 20.3. Is OpenAM Running? .....                           | 439 |
| 20.4. Debug Logging .....                                | 439 |
| 20.5. Session Management .....                           | 441 |
| 21. Tuning OpenAM .....                                  | 443 |
| 21.1. OpenAM Server Settings .....                       | 444 |
| 21.2. Java Virtual Machine Settings .....                | 449 |
| 21.3. Caching in OpenAM .....                            | 451 |
| 22. Changing Host Names .....                            | 457 |
| 23. Securing OpenAM .....                                | 461 |
| 23.1. Avoiding Obvious Defaults .....                    | 461 |
| 23.2. Protecting Network Access .....                    | 462 |
| 23.3. Securing OpenAM Administration .....               | 465 |
| 23.4. Securing Communications .....                      | 465 |
| 23.5. Administering the amadmin Account .....            | 466 |
| 24. Troubleshooting .....                                | 469 |
| OpenAM Glossary .....                                    | 477 |
| A. Release Levels & Interface Stability .....            | 483 |
| A.1. ForgeRock Product Release Levels .....              | 483 |
| A.2. ForgeRock Product Interface Stability .....         | 484 |
| Index .....  | 485 |



---

# Preface

This guide shows you how to configure, maintain, and troubleshoot OpenAM for single sign on and authorization, password reset, account lockout, cross-domain single sign on, and federation.

## 1 Who Should Use this Guide

This guide is written for access management designers and administrators who build, deploy, and maintain OpenAM services for their organizations. This guide covers the tasks you might repeat throughout the life cycle of an OpenAM release used in your organization.

This guide starts by introducing the OpenAM administrative interfaces and tools, and by showing how to manage OpenAM services. This guide continues by showing how to configure the principle features of OpenAM. It then demonstrates how to backup, restore, monitor, tune, and troubleshoot, OpenAM services.

You do not need to be an OpenAM wizard to learn something from this guide, though a background in access management and maintaining web application software can help. You do need some background in managing services on your operating systems and in your application servers. You can nevertheless get started with this guide, and then learn more as you go along.

## 2 Formatting Conventions

Most examples in the documentation are created on GNU/Linux or Mac OS X. Where it is helpful to make a distinction between operating environments, examples for UNIX, GNU/Linux, Mac OS X, and so forth are labeled (UNIX). Mac OS X specific examples can be labeled (Mac OS X). Examples for Microsoft Windows can be labeled (Windows). To avoid repetition, however, file system directory names are often given only in UNIX format as in /path/to/server, even if the text applies to C:\path\to\server as well.

Absolute path names usually begin with the placeholder /path/to/. This path might translate to /opt/, C:\Program Files\, or somewhere else on your system.

Command line, terminal sessions are formatted as follows.

```
$ echo $JAVA_HOME  
/path/to/jdk
```

Command output is sometimes formatted for narrower, more readable output even though formatting parameters are not shown in the command. In the following example, the query string parameter \_prettyPrint=true is omitted.

```
$ curl https://bjensen:hifalutin@opendj.example.com:8443/users/newuser  
{  
    "_rev" : "00000005b337348",  
    "schemas" : [ "urn:scim:schemas:core:1.0" ],  
    "contactInformation" : {  
        "telephoneNumber" : "+1 408 555 1212",  
        "emailAddress" : "newuser@example.com"  
    },  
    "_id" : "newuser",  
    "name" : {  
        "familyName" : "New",  
        "givenName" : "User"  
    },  
    "userName" : "newuser@example.com",  
    "displayName" : "New User",  
    "meta" : {  
        "created" : "2014-06-03T09:58:27Z"  
    },  
    "manager" : [ {  
        "_id" : "kvaughan",  
        "displayName" : "Kirsten Vaughan"  
    } ]  
}
```

Program listings are formatted as follows.

```
class Test {  
    public static void main(String [] args) {  
        System.out.println("This is a program listing.");  
    }  
}
```

## 3 Accessing Documentation Online

ForgeRock core documentation, such as what you are now reading, aims to be technically accurate and complete with respect to the software documented.

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 users' 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 validity with respect to the software, technical completeness with respect to the scope of the document, and usability for the expected audience.

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://docs.forgerock.org/>. Use this documentation when working with a ForgeRock Enterprise release.

In-progress documentation can be found at each project site under the [Developer Community](#) projects page. Use this documentation when trying a nightly build.

The ForgeRock [Community Wikis](#) and provide additional, user-created information. We encourage you to [join the community](#), so that you can update the Wikis, too.

## 4 Joining the ForgeRock Community

After you [sign up](#) to join the ForgeRock community, you can edit the [Community Wikis](#), and also log bugs and feature requests in the [issue tracker](#).

If you have a question regarding a project but cannot find an answer in the project documentation or Wiki, browse to the [Developer Community](#) page for the project, where you can find details on joining the project mailing lists, and find links to mailing list archives. You can also suggest updates to documentation through the [ForgeRock docs mailing list](#).

The Community Wikis describe how to check out and build source code. Should you want to contribute a patch, test, or feature, or want to author part of the core documentation, first have a look on the ForgeRock site at [how to get involved](#).



---

## **Chapter 1**

# **Administration Interfaces & Tools**

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

## **1.1 OpenAM Web-Based Console**

After you install OpenAM, login to the web-based 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`).

## OpenAM Web-Based Console

The screenshot shows the OpenAM Web-Based Console interface. At the top, there's a header bar with the OpenAM logo, the URL 'openam.example.com:8080/openam/task/Home', and a 'LOG OUT' button. Below the header is a navigation bar with tabs: 'Common Tasks', 'Access Control', 'Federation', 'Configuration', and 'Sessions'. The main content area contains several sections, each with a title, a brief description, and a 'Configure' button:

- Create SAMLv2 Providers**: Use these work flows to create hosted or remote identity and service providers for SAMLv2 Federation.
  - Create Hosted Identity Provider
  - Create Hosted Service Provider
  - Register Remote Identity Provider
  - Register Remote Service Provider
- Configure OAuth2/OpenID Connect**: This task configures OAuth2/OpenID Connect per realm. Each realm can act as an authorization server.
  - Configure OAuth2/OpenID Connect
- Create Fedlet**: Create a Fedlet to enable federation between an identity provider hosted on this instance of OpenAM and a remote service provider that does not have a federation solution. Before beginning, a hosted identity provider must be configured.
  - Create Fedlet
- Configure Google Apps**: Integrate OpenAM with Google Apps web applications to create a single sign-on environment. Before beginning, a hosted identity provider and Circle of Trust must be configured.
  - Configure Google Apps
- Configure Salesforce CRM**: Integrate OpenAM with Salesforce CRM to create a single sign-on environment. Before beginning, a SAMLv2 hosted identity provider and Circle of Trust must be configured.
  - Configure Salesforce CRM
- Configure Social Authentication**: Add social authentication options per realm. This task configures authentication through third parties such as Facebook, Google and Microsoft.
  - Configure Facebook Authentication
  - Configure Google Authentication
  - Configure Microsoft Authentication
  - Configure Other Authentication
- Test Federation Connectivity**: Use this automated test to determine if federation connections are being made successfully, or to identify where issues might be located.
  - Test Federation Connectivity
- Get Product Documentation**: Launch the OpenAM product documentation page.
  - Get Product Documentation

When you login as the OpenAM Administrator, `amadmin`, you have access to the complete OpenAM 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 login to the OpenAM console as a non-administrative end user, you do not have access to the administrative console. Your access is limited to a configuration page with your account information.

The screenshot shows a configuration form for a user named 'Babs Jensen'. The form includes fields for First Name, Last Name, Full Name, Password, Email Address, Telephone Number, Home Address, Password Reset Options, and Universal ID. There are 'Save' and 'Reset' buttons at the top right, and a note below them stating '\* Indicates required field'.

|                         |  |
|-------------------------|--|
| First Name:             | Barbara  |
| * Last Name:            | Jensen   |
| * Full Name:            | Babs Jensen                                      |
| Password:               | <a href="#">Edit</a>                             |
| Email Address:          | bjensen@example.com                              |
| Telephone Number:       | +1 415 523 0772                                  |
| Home Address:           | 500 3rd St San Francisco, CA 94107               |
| Password Reset Options: | <a href="#">Edit</a>                             |
| Universal ID:           | id=bjensen,ou=user,dc=openam,dc=forgerock,dc=org |

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

If you configure OpenAM to grant administrative capabilities to another user, then that user also sees the console after login. For instance, the OpenAM Administrator granted Kirsten Vaughan privileges to administer the OpenAM Top Level Realm. (This can be done through the console under Access Control > / (Top Level Realm) > Privileges. Kirsten has authorization to read and write policy properties and configured policy agent properties.) When Kirsten logs in, she sees only part of the console capabilities.<sup>2</sup>

The screenshot shows the 'Access Control' interface with a 'Realms' section. A message at the top states: 'A realm is the unit that OpenAM uses to organize configuration information. Authentication properties, authorization policies, data stores, subjects and other data can be defined within the realm. The top level realm is created when you deploy OpenAM. The top level realm is the root of the OpenAM instance and contains OpenAM configuration data.' Below this, a table lists one item: '/ (Top Level Realm)' under the 'Realm Name' column and '/' under the 'Location' column. There are 'New...' and 'Delete' buttons above the table.

## 1.2 OpenAM Command-Line Tools

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

You can install the following OpenAM command-line tools.

### **agentadmin**

This tool lets you manage OpenAM policy agent installations.

Unpack this tool as part of policy agent installation.

### **ampassword**

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

Install this from the SSOAdminTools-12.0.0.zip.

### **amverifyarchive**

This tool checks log archives for tampering.

Install this from SSOAdminTools-12.0.0.zip.

### **openam-distribution-configuration-12.0.0.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

---

<sup>2</sup>For more on delegated administration, see the chapter covering realms.

*config.file*. The sample configuration 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-12.0.0.zip**.

### **ssoadm**

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

In a test environment you can activate **ssoadm.jsp** to access the same functionality in your browser. Once active, you can use many features of the **ssoadm** command, by navigating to the **ssoadm.jsp** URI, in a URL such as <http://openam.example.com:8080/openam/ssoadm.jsp>.

Install this from **SSOAdminTools-12.0.0.zip**.

To translate settings applied in OpenAM console to service attributes for use with **ssoadm**, login to the OpenAM 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.ipplanet.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
```

## 1.3 OpenAM ssoadm.jsp

You can use the **ssoadm.jsp** page to access a large subset of the configuration capabilities of the **ssoadm** command. Yet, **ssoadm.jsp** is disabled by default to prevent potential misuse.

### Procedure 1.1. To Enable ssoadm.jsp

1. Login as OpenAM administrator, amadmin.
2. Click Configuration > Servers and Sites > Servers > *URL of your server*.
3. Click Advanced to display the Advanced Properties table, and then click Add. In the text boxes that appear, include the following information, and then click Save.

|                |                 |
|----------------|-----------------|
| Property Name  |                 |
|                | ssoadm.disabled |
| Property Value |                 |
|                | false           |

4. To see if the change worked, navigate to the URL of OpenAM with the / **ssoadm.jsp** URI. For the aforementioned URL, you would navigate to <http://openam.example.com:8080/openam/ssoadm.jsp>.



---

## **Chapter 2**

# Defining Authentication Services

An *authentication* service confirms the identity of a user or a client application.

This chapter describes how to configure authentication in OpenAM.

## **2.1**

### **About Authentication in OpenAM**

Access management is about controlling access to resources. OpenAM plays a role similar to border control at an international airport. Instead of having each and every airline company deal with access to each destination, all airlines redirect passengers to border control. Border control then determines who each passenger is according to passport credentials. Border control also checks whether the identified passenger is authorized to fly to the destination corresponding to the ticket, perhaps based on visa credentials. Then, at the departure gate, an agent enforces the authorization from border control, allowing the passenger to board the plane as long as the passenger has not gotten lost, or tried to board the wrong plane, or swapped tickets with someone else. Thus, border control handles access management at the airport.

OpenAM is most frequently used to protect web-accessible resources. Users browse to a protected web application page. An agent installed on the server with the web application redirects the user to OpenAM for access management. OpenAM determines who the user is, and whether the user has the right to access the protected page. OpenAM then redirects the user back to the protected page, with authorization credentials that can be verified by the agent. The agent allows OpenAM authorized users access the page.

Notice that OpenAM basically needs to determine two things for access management: the identity of the user, and whether the user has access rights to the protected page. *Authentication* is how OpenAM identifies the user. This chapter covers how to set up the authentication process. *Authorization* is how OpenAM determines whether a user has access to a protected resource. Authorization is covered later.

For authentication, OpenAM uses credentials from the user or client application. It then uses defined mechanisms to validate credentials and complete the authentication. The authentication methods can vary. For example, passengers travelling on international flights authenticate with passports and visas. In contrast, passengers travelling on domestic flights might authenticate with an identity card or a driver's license. Customers withdrawing cash from an ATM authenticate with a card and a PIN.

OpenAM allows you to configure authentication processes and then customize how they are applied. OpenAM uses *authentication modules* to handle different ways of authenticating. Basically, each authentication module handles one way of obtaining and verifying credentials. You can chain different authentication modules together. In OpenAM, this is called *authentication chaining*. Each authentication module can be configured to specify the continuation and failure semantics with one of the following four flags: required, optional, requisite, or sufficient.<sup>1</sup>

- When a *required* module fails, the rest of the chain is processed, but the authentication fails.

A required module might be used for login with email and password, but then fall through to another module to handle new users who have not yet signed up.

- When an *optional* module fails, authentication continues.

An optional module might be used to permit a higher level of access if the user can present a X.509 certificate for example.

- When a *requisite* module fails, authentication fails and authentication processing stops.

A requisite module might be used with exclusive SSO.

- When a *sufficient* succeeds, authentication is successful and later modules in the chain are skipped.

---

<sup>1</sup>The four flags, required, optional, requisite, and sufficient, come from the standards created for the Java Authentication and Authorization Service (JAAS).

You could set Windows Desktop SSO as sufficient, so authenticated Windows users are let through, whereas web users have to traverse another authentication module such as one requiring an email address and a password.

With OpenAM, you can further set *authentication levels* per module, with higher levels being used typically to allow access to more restricted resources. The OpenAM SPIs also let you develop your own authentication modules, and post-authentication plugins. Client applications can specify the authentication level, module, user, and authentication service to use among those you have configured. As described later in this guide, you can use *realms* to organize which authentication process applies for different applications or different domains, perhaps managed by different people.

When a user successfully authenticates, OpenAM creates a session, which allows OpenAM to manage that user's access to resources. In some deployments you need to limit how many active sessions a user can have at a given time. For example, you might want to prevent a user from using more than two devices at once. See [Section 2.9, “Configuring Session Quotas”](#) for instructions.

OpenAM leaves the authentication process flexible so that you can adapt how it works to your situation. Although at first the number of choices can seem daunting, now that you understand the basic process, you begin to see how choosing authentication modules and arranging them in authentication chains lets you use OpenAM to protect access to a wide range of applications used in your organization.

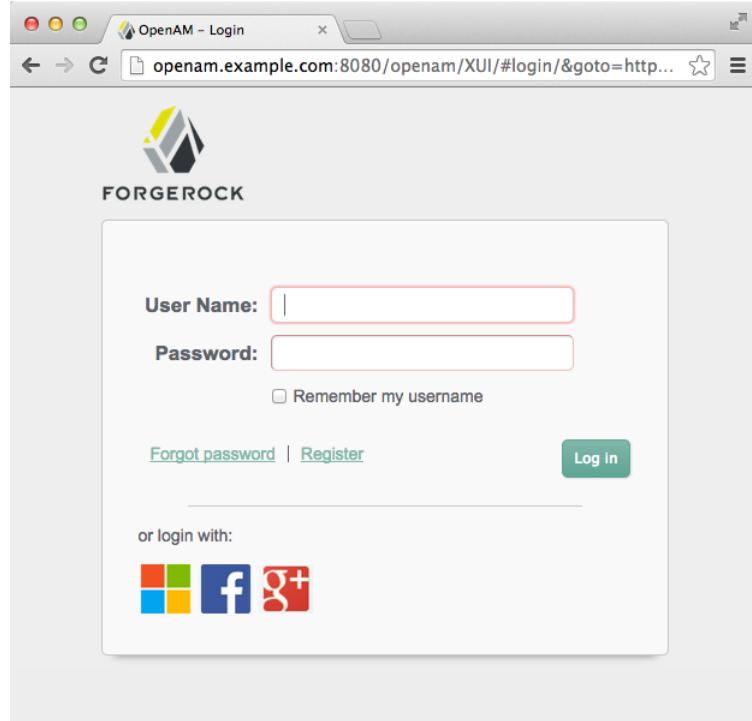
## 2.2 Configuring Social Authentication

OpenAM allows delegation of authentication to any third party OpenID Connect 1.0 server that implements the [OpenID Connect Discovery 1.0 specification](#).

The OpenAM console provides wizards for configuring authentication with selected third parties: Facebook, Google, or Microsoft. An additional wizard provides the ability to configure other third party authentication providers.

The wizards create an Authentication Module and an Authentication Chain containing the correct configuration needed to authenticate with the third party. The wizard also adds configuration data to the realm's *Social Authentication Implementations Service* (and provisions the service if it is not already present) that enables the display of logos of configured third party authentication providers on the OpenAM login screen, as shown below.

**Figure 2.1. Login Screen With Social Authentication Logos**



### 2.2.1 Configuring Pre-Populated Social Authentication Providers

OpenAM provides wizards to quickly enable authentication with Facebook, Google and Microsoft. Most settings are pre-populated, only a *Client ID* and *Client Secret* are required.

To obtain a *Client ID* and *Client Secret* you should register an application with the third party provider, at the following links:

Facebook  
[Facebook App Quickstart](#)

Google  
[Google Developers Console](#)

## Note

You must enable the Google+ API in order to authenticate with Google. To enable the Google+ API, login to the Google Developers Console, select your project, navigate to APIs & auth > APIs, and then set the status of the Google+ API to ON.

Microsoft

[Microsoft account Developer Center](#)

### Procedure 2.1. To Configure Pre-Populated Social Authentication Providers

Once you have registered an application and obtained credentials from the social authentication provider, follow the steps below to configure authentication with the provider.

1. On the Common Tasks tab page of the OpenAM console, click the link for the social authentication provider you want to configure, either *Configure Facebook Authentication*, *Configure Google Authentication*, or *Configure Microsoft Authentication*.
2. On the configure third party authentication page:
  - a. Select the realm in which to enable social authentication.
  - b. Enter the *Client ID* obtained from the third party authentication provider.
  - c. Enter the *Client Secret* obtained from the third party authentication provider, and repeat it in the *Confirm Client Secret* field.
  - d. Leave the default *Redirect URL*, unless you are using an external server as a proxy.
  - e. Click *Create*.

**Figure 2.2. The Configure Google Authentication Wizard**

The screenshot shows a web browser window titled "OpenAM" with the URL "openam.example.com:8080/openam/task/ConfigureSocialAuthN?type=google". The page header includes "VERSION", "User: amAdmin Server: forgerock.example.com", and "LOG OUT". Below the header is the FORGEROCK logo. The main content area is titled "Configure Google Authentication" with a "Create" and "Cancel" button. A note states: "Configure Social Authentication using Google as the identity provider. Use the [Google Developers Console](#) to register your application with Google. Once created, select 'Credentials' in the 'APIs & auth' section and then click the 'Create new Client ID' button under 'OAuth' to be guided through creating an OAuth 2.0 client ID. Once created, copy the CLIENT ID and CLIENT SECRET values into the respective fields below to complete the configuration." A red asterisk indicates required fields. The "Realm" section has a dropdown menu set to "/". The "Client Details" section contains the following fields:

|                          |  |   |
|--------------------------|--|---|
| * Client ID:             | 00000000112233AABCC  | For more information on the OAuth client_id parameter refer to the <a href="#">OAuth IETF draft</a> , chapter 2.1   |
| * Client Secret:         | .....  | For more information on the OAuth client_secret parameter refer to the <a href="#">OAuth IETF draft</a> , chapter 2.1   |
| * Confirm Client Secret: | .....  |   |
| * Redirect URL:          | http://openam.example.com:8080/openam/oauth2c/OAuthProxy.jsp | This URL should only be changed from the default, if an external server is performing the GET to POST proxying. The default is /openam/oauth2c/OAuthProxy.jsp |

On completion, the wizard displays a message confirming the successful creation of a new Authentication Module and an Authentication Chain for the provider, and either the creation of a new Social Authentication Implementations service named `socialAuthNService`, or an update if it already existed.

You can configure the Authentication Module, Authentication Chain, and Social Authentication Implementations service that you created by using the wizards in the same way as manually created versions. For more information, see [Section 2.3, “Configuring Authentication Modules”](#), [Section 2.4, “Configuring Authentication Chains”](#), and [Section 2.2.3, “Configuring the Social Authentication Implementations service”](#).

### 2.2.2 Configuring Custom Social Authentication Providers

OpenAM provides a wizard to quickly enable authentication with any third party provider that supports the [OpenID Connect Discovery 1.0 specification](#).

You must first register an application with the third party provider to obtain a *Client ID*, *Client Secret*, and the *OpenID Discovery URL*.

### Procedure 2.2. To Configure Custom Social Authentication Providers

Once you have registered an application and obtained your credentials from the social authentication provider, follow the steps below to configure authentication with the provider.

1. On the Common Tasks tab page of the OpenAM console, click the [Configure Other Authentication](#) link.
2. On the configure social authentication page:
  - a. Select the realm in which to enable social authentication.
  - b. Enter the *OpenID Discovery URL* obtained from the third party authentication provider.
  - c. Enter a name for the provider in the **Provider Name** field. OpenAM uses this as a label on the login page to identify the provider.
  - d. Enter the URL of an image to be used on the login page in the **Image URL** field. OpenAM places the image on the login page, to enable authentication with the provider.
  - e. Enter the *Client ID* obtained from the third party authentication provider.
  - f. Enter the *Client Secret* obtained from the third party authentication provider, and repeat it in the **Confirm Client Secret** field.
  - g. Leave the default **Redirect URL**, unless you are using an external server as a proxy.
  - h. Click **Create**.

Figure 2.3. The Configure Social Authentication Wizard

The screenshot shows a web browser window for 'OpenAM' at 'openam.example.com:8080/openam/task/ConfigureSocialAuthN?type=other'. The header includes 'VERSION', 'User: amAdmin Server: forgerock.example.com', and 'LOG OUT'. Below the header is the 'FORGEROCK' logo. The main content is titled 'Configure Social Authentication' with a sub-instruction 'Configure a social authentication provider via OpenID Connect.' On the right are 'Create' and 'Cancel' buttons, with a note '\* Indicates required field'. The form has sections for 'Realm' (with a dropdown for 'Realm') and 'Provider Details' (containing fields for 'OpenID Discovery URL' (set to 'https://login.salesforce.com/.well-known/openid-configuration'), 'Provider Name' (set to 'Salesforce'), and 'Image URL' (set to 'XUI/images/dashboard/salesforce.png')). Below that is a 'Client Details' section with fields for 'Client ID' (set to '00112233AABBCC'), 'Client Secret' (set to '\*\*\*\*\*'), 'Confirm Client Secret' (set to '\*\*\*\*\*'), and 'Redirect URL' (set to 'http://openam.example.com:8080/openam/oauth2c/OAuthProxy.jsp'). A note next to the Redirect URL field states: 'This URL should only be changed from the default, if an external server is performing the GET to POST proxying. The default is /openam/oauth2c/OAuthProxy.jsp'.

On completion, the wizard displays a message confirming the successful creation of a new Authentication Module and an Authentication Chain for the provider, and either the creation of a new Social Authentication Implementations service named `socialAuthNService`, or an update if it already existed.

You can configure the Authentication Module, Authentication Chain, and Social Authentication Implementations service that you created by using the wizard in the same way as manually created versions. For more information, see Section 2.3, “Configuring Authentication Modules”, Section 2.4, “Configuring Authentication Chains”, and Section 2.2.3, “Configuring the Social Authentication Implementations service”.

## 2.2.3 Configuring the Social Authentication Implementations service

You can add logos to the login page to allow users to authenticate using configured social authentication providers.

Wizards are provided to configure common social authentication providers, which also configure the Social Authentication Implementations service to add logos to the login page. You can manually add other Authentication Chains that contain an OAuth 2.0 / OpenID Connect authentication module.

To add a social authentication provider to the login screen, you must first configure an OAuth 2.0 / OpenID Connect Authentication Module , and an Authentication Chain that contains it. You can create the by using a wizard, see [Section 2.2.1, “Configuring Pre-Populated Social Authentication Providers”](#) and [Section 2.2.2, “Configuring Custom Social Authentication Providers”](#) or created manually, see [Section 2.3, “Configuring Authentication Modules”](#) and [Section 2.4, “Configuring Authentication Chains”](#).

### Procedure 2.3. To Configure Social Authentication Implementations

Once you have created an Authentication Chain containing an OAuth 2.0 / OpenID Connect Authentication Module, follow the steps below to add a logo for the authentication provider to the login screen.

1. On the Access Control tab page of the OpenAM console, click the realm containing the authentication module and authentication chain to be added to the login screen.
2. On the Services tab page for the realm:
  - If the Social Authentication Implementations service exists, click on it.
  - If the Social Authentication Implementations service does not exist, click New, select Social Authentication Implementations, and then click Next.
3. On the Social Authentication Implementations page:
  - a. In the *Display Names* section, enter a Map Key, enter the text to display as ALT text on the logo in the Corresponding Map Value field, and then click Add.

### Note

OpenAM uses the value in the Map Key fields throughout the configuration to tie the various implementation settings to each other. The value is case-sensitive.

## Configuring the Social Authentication Implementations service

---

- b. In the *Authentication Chains* section, re-enter the Map Key used in the previous step, select the Authentication Chain from the Corresponding Map Value list, and then click Add.
- c. In the *Icons* section, re-enter the Map Key used in the previous steps, enter the path to a logo image to be used on the login screen in the Corresponding Map Value list, and then click Add.
- d. In the *Enabled Implementations* section, re-enter the Map Key used in the previous steps, and then click Add.

### Tip

Removing a Map Key from the Enabled Implementations list removes the associated logo from the login screen. There is no need to delete the Display Name, Authentication Chain or Icon configuration to remove the logo from the login screen.

- e. Click Add or Save.

## Configuring Authentication Modules

**Figure 2.4. Configuring the Social Authentication Implementations service**

The screenshot shows the 'Social Authentication Implementations' configuration page. At the top, there are tabs for 'VERSION' and 'LOG OUT'. Below that, it shows the user 'amAdmin' and server 'forgerock.example.com'. The main area is titled 'Social Authentication Implementations' with buttons for 'Save', 'Reset', and 'Back to Services'.

**Realm Attributes**

**Display Names**

Current Values: [Salesforce]=Salesforce

New Value:  Map Key:  Corresponding Map Value:

[?] The display names for the implementations - this will be used to provide a name for the icon displayed on the login page. The key should be used across all the settings on this page to join them together.

**Authentication Chains**

Current Values: [Salesforce]=SalesforceSocialAuthenticationService

New Value:  Map Key:  Corresponding Map Value:  SalesforceSocialAuthenticationService

[?] The name of the authentication chains that are the entry points to being authenticated by each respective social authentication provider. The key should correspond to a key used to define a Display Name above.

**Icons**

Current Values: [Salesforce]=/images/salesforce.png

New Value:  Map Key:  Corresponding Map Value:

[?] Either a full URL or a path relative to the base of the site/server where the image can be found. The image will be used on the login page to link to the authentication chain defined above. The key should correspond to a key used to define a Display Name above.

**Enabled Implementations**

Current Values: Salesforce

New Value:

[?] Provide a key that has been used to define the settings above to enable that set of settings.

An icon now appears on the OpenAM login screen, allowing users to authenticate with the third party authentication provider.

## 2.3 Configuring Authentication Modules

The OpenAM console provides two places where the OpenAM administrator can configure authentication modules.

1. Under Configuration > Authentication, you configure available modules for use throughout OpenAM. What you set up here is inherited for use elsewhere.
2. Under Access Control > *Realm Name* > Authentication, you configure modules for your realm. What you set up at this level inherits from the global configuration, but you can override what is inherited. You can also add your own modules if necessary.

The configuration of individual modules depend on its function. The configuration of an Active Directory over LDAP user authentication module requires connection information and details about where to search for users. In contrast, the configuration of the HOTP module for OTP authentication requires data about the password length and the mail server or SMS gateway for to send the password during authentication.

### 2.3.1 Hints for the Active Directory Authentication Module

OpenAM connects to Active Directory over Lightweight Directory Access Protocol (LDAP). OpenAM provides separate Active Directory and LDAP modules to support the use of both Active Directory and another directory service in an authentication chain.

**ssoadm** service name: sunAMAuthADService

Primary Active Directory Server

Secondary Active Directory Server

The default port for LDAP is 389. If you are connecting to Active Directory over SSL, the default port for LDAP/SSL is 636.

To allow users to change passwords through OpenAM, Active Directory requires that you connect over SSL.

If you want to use SSL or TLS for security, then scroll down to enable SSL/TLS Access to Active Directory Server. Make sure that OpenAM can trust the Active Directory certificate when using this option.

OpenAM first attempts to contact primary servers. If no primary server is available, then OpenAM attempts to contact secondaries.

When authenticating users from a directory server that is remote from OpenAM, set both the primary and secondary server values.

**ssoadm** attributes: primary is `iplanet-am-auth-ldap-server`; secondary is `iplanet-am-auth-ldap-server2`

DN to Start User Search

LDAP data is organized hierarchically, a bit like a file system on Windows or UNIX. More specific DNs likely result in better performance. When

configuring the module for a particular part of the organization, you can perhaps start searches from a specific organizational unit such as `OU=sales, DC=example, DC=com`.

If multiple entries exist with identical search attribute values, make this value specific enough to return only one entry.

**ssoadm** attribute: `iplanet-am-auth-ldap-base-dn`

#### Bind User DN, Bind User Password

If OpenAM stores attributes in Active Directory, for example to manage account lockout, or if Active Directory requires that OpenAM authenticate in order to read users' attributes, then OpenAM needs the DN and password to authenticate to Active Directory.

The default is `amldapuser`. If the administrator authentication chain (default: `ldapService`) has been configured to include only the Active Directory module, then make sure that the password is correct before you logout. If it is incorrect, you will be locked out. If you do get locked out, you can login with the super user DN, which by default is `uid=amAdmin,ou=People,OpenAM-deploy-base`, where *OpenAM-deploy-base* was set during OpenAM configuration.

**ssoadm** attributes: `iplanet-am-auth-ldap-bind-dn` and `iplanet-am-auth-ldap-bind-passwd`

#### Attributes Used to Retrieve User Profile

#### Attributes Used to Search for a User to be Authenticated

##### User Search Filter

##### Search Scope

LDAP searches for user entries return entries with attribute values matching the filter you provide. For example if you search under `CN=Users,DC=example,DC=com` with a filter "`(MAIL=bjensen@example.com)`", then the directory returns the entry that has `MAIL=bjensen@example.com`. In this example the attribute used to search for a user is `mail`. Multiple attribute values mean the user can authenticate with any one of the values. For example, if you have both `uid` and `mail`, then Barbara Jensen can authenticate with either `bjensen` or `bjensen@example.com`.

The User Search Filter text box provides a more complex filter.

For example, if you search on `mail` and add User Search Filter (`objectClass=inetOrgPerson`), then OpenAM uses the resulting search filter `(&(mail=address) (objectClass=inetOrgPerson))`, where `address` is the mail address provided by the user.

This controls how and the level of the directory that will be searched. You can set the search to run at a high level or against a specific area.

- `OBJECT` will search only for the entry specified as the DN to Start User Search.

- ONELEVEL will search only the entries that are directly children of that object.
- SUBTREE will search the entry specified and every entry under it.

**ssoadm** attributes: `iplanet-am-auth-ldap-user-naming-attribute`, `iplanet-am-auth-ldap-user-search-attributes`, `iplanet-am-auth-ldap-search-filter`, and `iplanet-am-auth-ldap-search-scope`

#### SSL/TLS Access to Active Directory Server

If you enable SSL/TLS, OpenAM must be able to trust Active Directory certificates, either because the Active Directory 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: `iplanet-am-auth-ldap-ssl-enabled`

#### Return User DN to DataStore

When enabled, and OpenAM uses Active Directory as the user store, the module returns the DN rather than the User ID, so the bind for authentication can be completed without a search to retrieve the DN.

**ssoadm** attribute: `iplanet-am-auth-ldap-return-user-dn`

#### User Creation Attributes

This list lets you map (external) attribute names from Active Directory to (internal) attribute names used by OpenAM.

**ssoadm** attribute: `iplanet-am-ldap-user-creation-attr-list`

#### LDAP Connection Heartbeat Interval

Specifies how often OpenAM should 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.

Default: 1

**ssoadm** attribute: `openam-auth-ldap-heartbeat-interval`

#### LDAP Connection Heartbeat Time Unit

Specifies the time unit corresponding to LDAP Connection Heartbeat Interval.

Default: minute

**ssoadm** attribute: `openam-auth-ldap-heartbeat-interval`

#### Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: sunAMAuthADAuthLevel

### 2.3.2 Hints for the Adaptive Risk Authentication Module

The Adaptive Risk module is designed to assess risk during authentication so that OpenAM can determine whether to require the user to complete further authentication steps. After configuring the Adaptive Risk module, insert it in your authentication chain with criteria set to sufficient as shown in the following example.

The screenshot shows a configuration interface titled "AdaptiveRisk - Properties". At the top right are three buttons: "Save", "Reset", and "Back to Authentication". Below the title is a table header "(3 Item(s))" with columns: "Add", "Remove", and "Reorder". The main table has three rows, each representing an authentication module:

| Instance     | Criteria   | Options |
|--------------|------------|---------|
| LDAP         | REQUIRED   |         |
| AdaptiveRisk | SUFFICIENT |         |
| HOTP         | REQUIRED   |         |

In the example authentication chain shown, OpenAM has users authenticate first using the LDAP module providing a user ID and password combination. Upon success, OpenAM calls the Adaptive Risk module. The Adaptive Risk module assesses the risk based on your configured parameters. If the Adaptive Risk module calculates a total score below the threshold you set, the module returns success, and OpenAM finishes authentication processing without requiring further credentials. Otherwise the Adaptive Risk module evaluates the score to be above the risk threshold, and returns failure. OpenAM then calls the HOTP module, requiring the user to authenticate with a one-time password delivered to her by email or by SMS to her mobile phone.

When you configure the Adaptive Risk module to save cookies and profile attributes after successful authentication, OpenAM performs the save as post-authentication processing, only after the entire authentication chain returns success. You must set up OpenAM to save the data as part of post-authentication processing by editing the authentication chain to add `org.forgerock.openam.authentication.modules.adaptive.Adaptive` to the list of post authentication plugins.

When the Adaptive Risk module relies 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 the *Installation Guide* section, [Handling HTTP Request Headers](#).

**ssoadm** service name: sunAMAuthAdaptiveService

## General

### Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: openam-auth-adaptive-auth-level

### Risk Threshold

Risk threshold score. If the sum of the Scores is greater than the threshold, the Adaptive Risk module returns failure. Default: 1

**ssoadm** attribute: openam-auth-adaptive-auth-threshold

## Failed Authentications

### Failed Authentication Check

When enabled, check the user profile for authentication failures since the last successful login. This check therefore requires OpenAM to have access to the user profile, and Account Lockout to be enabled (otherwise OpenAM does not record authentication failures).

**ssoadm** attribute: openam-auth-adaptive-failure-check

### Score

Value to add to the total score if the user fails the Failed Authentication Check. Default: 1

**ssoadm** attribute: openam-auth-adaptive-failure-score

### Invert Result

When selected, add the Score to the total score if the user passes the Failed Authentication Check.

**ssoadm** attribute: openam-auth-adaptive-failure-invert

## IP Address Range

### IP Range Check

When enabled, check whether the client IP address is within one of the specified IP Ranges.

**ssoadm** attribute: openam-auth-adaptive-ip-range-check

IP Range

For IPv4, specifies a list of IP ranges either in CIDR-style notation ( $x.x.x.x/YY$ ) or as a range from one address to another ( $x.x.x.x-y.y.y.y$ , meaning from  $x.x.x.x$  to  $y.y.y.y$ ).

For IPv6, specifies a list of IP ranges either in CIDR-style notation ( $X:X:X:X:X:X:X/YY$ ) or as a range from one address to another ( $X:X:X:X:X:X-X-Y:Y:Y:Y:Y:Y:Y$ , meaning from  $X:X:X:X:X:X:X$  to  $Y:Y:Y:Y:Y:Y:Y:Y$ ).

**ssoadm** attribute: openam-auth-adaptive-ip-range-range

Score

Value to add to the total score if the user fails the IP Range Check. Default: 1

**ssoadm** attribute: openam-auth-adaptive-ip-range-score

Invert Result

When selected, add the Score to the total score if the user passes the IP Range Check.

**ssoadm** attribute: openam-auth-adaptive-ip-range-invert

## IP Address History

IP History Check

When enabled, check whether the client IP address matches one of the known values stored on the profile attribute you specify. This check therefore requires that OpenAM have access to the user profile.

**ssoadm** attribute: openam-auth-adaptive-ip-history-check

History Size

Specifies how many IP address values to retain on the profile attribute you specify. Default: 5

**ssoadm** attribute: openam-auth-ip-adaptive-history-count

Profile Attribute Name

Name of the user profile attribute on which to store known IP addresses.  
Default: iphistory

**ssoadm** attribute: openam-auth-adaptive-ip-history-attribute

Save Successful IP Address

When enabled, save new client IP addresses to the known IP address list following successful authentication.

**ssoadm** attribute: openam-auth-adaptive-ip-history-save

Score

Value to add to the total score if the user fails the IP History Check. Default: 1

**ssoadm** attribute: openam-auth-adaptive-ip-history-score

Invert Result

When selected, add the Score to the total score if the user passes the IP History Check.

**ssoadm** attribute: openam-auth-adaptive-ip-history-invert

## Known Cookie

Cookie Value Check

When enabled, check whether the client browser request has the specified cookie and optional cookie value.

**ssoadm** attribute: openam-auth-adaptive-known-cookie-check

Cookie Name

Specifies the name of the cookie for which OpenAM checks when you enable the Cookie Value Check.

**ssoadm** attribute: openam-auth-adaptive-known-cookie-name

Cookie Value

Specifies the value of the cookie for which OpenAM checks. If no value is specified, OpenAM does not check the cookie value.

**ssoadm** attribute: openam-auth-adaptive-known-cookie-value

Save Cookie Value on Successful Login

When enabled, save the cookie as specified in the client's browser following successful authentication. If no Cookie Value is specified, the value is set to 1.

**ssoadm** attribute: openam-auth-adaptive-known-cookie-save

Score

Value to add to the total score if user passes the Cookie Value Check.  
Default: 1

**ssoadm** attribute: openam-auth-adaptive-known-cookie-score

Invert Result

When selected, add the Score to the total score if the user passes the Cookie Value Check.

**ssoadm** attribute: openam-auth-adaptive-known-cookie-invert

### Device Cookie

#### Device Registration Cookie Check

When enabled, check whether the client browser request has the specified cookie with the correct device registration identifier as the value.

**ssoadm** attribute: openam-auth-adaptive-device-cookie-check

#### Cookie Name

Specifies the name of the cookie for the Device Registration Cookie Check.  
Default: Device

**ssoadm** attribute: openam-auth-adaptive-device-cookie-name

#### Save Device Registration on Successful Login

When enabled, save the specified cookie with a hashed device identifier value in the client's browser following successful authentication.

**ssoadm** attribute: openam-auth-adaptive-device-cookie-save

#### Score

Value to add to the total score if the user fails the Device Registration Cookie Check. Default: 1

**ssoadm** attribute: openam-auth-adaptive-device-cookie-score

#### Invert Result

When selected, add the Score to the total score if the user passes the Device Registration Cookie Check.

**ssoadm** attribute: openam-auth-adaptive-device-cookie-invert

### Time Since Last Login

#### Time Since Last Login Check

When enabled, check whether the client browser request has the specified cookie that holds the encrypted last login time, and check that the last login time is more recent than a maximum number of days you specify.

**ssoadm** attribute: openam-auth-adaptive-time-since-last-login-check

#### Cookie Name

Specifies the name of the cookie holding the encrypted last login time value.

**ssoadm** attribute: openam-auth-adaptive-time-since-last-login-cookie-name

Max Time since Last Login

Specifies a threshold age of the last login time in days. If the client's last login time is more recent than the number of days specified, then the client successfully passes the check.

**ssoadm** attribute: `openam-auth-adaptive-time-since-last-login-value`

Save time of Successful Login

When enabled, save the specified cookie with the current time encrypted as the last login value in the client's browser following successful authentication.

**ssoadm** attribute: `openam-auth-adaptive-time-since-last-login-save`

Score

Value to add to the total score if the user fails the Time Since Last Login Check. Default: 1

**ssoadm** attribute: `openam-auth-adaptive-time-since-last-login-score`

Invert Result

When selected, add the Score to the total score if the user passes the Time Since Last Login Check.

**ssoadm** attribute: `openam-auth-adaptive-time-since-last-login-invert`

## Profile Attribute

Profile Risk Attribute check

When enabled, check whether the user profile contains the specified attribute and value.

**ssoadm** attribute: `openam-auth-adaptive-risk-attribute-check`

Attribute Name

Specifies the attribute to check on the user profile for the specified value.

**ssoadm** attribute: `openam-auth-adaptive-risk-attribute-name`

Attribute Value

Specifies the value to match on the profile attribute. If the attribute is multi-valued, a single match is sufficient to pass the check.

**ssoadm** attribute: `openam-auth-adaptive-risk-attribute-value`

Score

Value to add to the total score if the user fails the Profile Risk Attribute Check. Default: 1

**ssoadm** attribute: `openam-auth-adaptive-risk-attribute-score`

#### Invert Result

When selected, add the Score to the total score if the user passes the Profile Risk Attribute Check.

**ssoadm** attribute: `openam-auth-adaptive-risk-attribute-invert`

### Geo Location

#### Geolocation Country Code Check

When enabled, check whether the client IP address location matches a country specified in the Valid Country Codes list. The

**ssoadm** attribute: `forgerock-am-auth-adaptive-geo-location-check`

#### Geolocation Database location

Path to GeoIP data file used to convert IP addresses to country locations. The geolocation database is not packaged with OpenAM. You can download the GeoIP Country database from [MaxMind](#). Use the binary .dat file format, rather than .csv. You can use the GeoLite Country database for testing.

**ssoadm** attribute: `openam-auth-adaptive-geo-location-database`

#### Valid Country Codes

Specifies the list of country codes to match. Use | to separate multiple values.

**ssoadm** attribute: `openam-auth-adaptive-geo-location-values`.

#### Score

Value to add to the total score if the user fails the Geolocation Country Code Check. Default: 1

**ssoadm** attribute: `openam-auth-adaptive-geo-location-score`

#### Invert Result

When selected, add the Score to the total score if the user passes the Geolocation Country Code Check.

**ssoadm** attribute: `openam-auth-adaptive-geo-location-invert`

### Request Header

#### Request Header Check

When enabled, check whether the client browser request has the specified header with the correct value.

**ssoadm** attribute: `openam-auth-adaptive-req-header-check`

Request Header Name

Specifies the name of the request header for the Request Header Check.

**ssoadm** attribute: openam-auth-adaptive-req-header-name

Request Header Value

Specifies the value of the request header for the Request Header Check.

**ssoadm** attribute: openam-auth-adaptive-req-header-value

Score

Value to add to the total score if the user fails the Request Header Check.

Default: 1

**ssoadm** attribute: openam-auth-adaptive-req-header-score

Invert Result

When selected, add the Score to the total score if the user passes the Request Header Check.

**ssoadm** attribute: openam-auth-adaptive-req-header-invert

### 2.3.3 Hints for the Anonymous Authentication Module

This module lets you configure and track anonymous users, who can log in to your application or web site without login credentials. Typically, you would provide such users with very limited access, for example, an anonymous user may have access to public downloads on your site. When the user attempts to access resources that require more protection, the module can force further authentication for those resources.

You can configure the Anonymous Authentication Module by specifying the **ssoadm** service name and Anonymous Authentication realm attributes: Valid Anonymous Users, Default User Name, Case Sensitive User IDs, and Authentication Level.

**ssoadm** service name: iPlanetAMAuthAnonymousService

Valid Anonymous Users

Specifies the list of valid anonymous user IDs that can log in without submitting a password.

**ssoadm** attribute: iplanet-am-auth-anonymous-users-list

When user accesses the default module instance login URL, then the module prompts the user to enter a valid anonymous user name.

The default module instance login URL is defined as follows:

## Hints for the Anonymous Authentication Module

```
protocol://hostname:port/deploy_URI/UI/Login?module=Anonymous&org=org_name
```

Next, OpenAM checks that the user ID is a valid anonymous user name. If the valid anonymous user ID exists in the list, the user is granted access to the application and the session is assigned to the user. The valid user can bypass the Login page at the following URL:

```
protocol://hostname:port/deploy_URI/UI/Login?module=Anonymous&org=org_name&IDToken1=<valid anonymous username>
```

If the valid anonymous user ID does not exist in the list, the user will be authenticated as the Default Anonymous User Name (see below).

### Default Anonymous User Name

Specifies the user ID assigned by the module if the Valid Anonymous Users list is empty. The default value is anonymous. Note that the anonymous user must be defined in the realm.

```
protocol://hostname:port/deploy_URI/UI/Login?module=Anonymous&org=orgName&IDToken1=<Default Anonymous User Name>
```

For example, the user can bypass the Login page by accessing the URL as the default anonymous user using the value of anonymous:

```
http://openam.example.com:8080/openam/UI/Login?module=Anonymous&org=example.com&IDToken1=anonymous
```

**ssoadm** attribute: iplanet-am-auth-anonymous-default-user-name

### Case Sensitive User IDs

Determines whether case matters for anonymous user IDs.

**ssoadm** attribute: iplanet-am-auth-anonymous-case-sensitive

### Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 (default) to any positive integer and is set for each authentication method. The higher number corresponds to a higher level of authentication. If you configured your authentication levels from a 0 to 5 scale, then an authentication level of 5 will require the highest level of authentication.

After a user has authenticated, OpenAM stores the authentication level in the session token. When the user attempts to access a protected resource, the token is presented to the application. The application uses the token's value to determine if the user has the correct authentication level required to access the resource. If the user does not have the required authentication level, the application can prompt the user to authenticate with a higher authentication level.

**ssoadm** attribute: `iplanet-am-auth-anonymous-auth-level`

## Note

You can configure the Anonymous Authentication Module using the OpenAM Console by clicking Configuration > Authentication > Anonymous.

### 2.3.4

## Hints for the Certificate Authentication Module

X.509 digital certificates can enable secure authentication without the need for user names and passwords or other credentials. Certificate authentication can be handy to manage authentication by applications. If all certificates are signed by a recognized Certificate Authority (CA), then you might get away without additional configuration. If you need to look up public keys of OpenAM clients, this module can also look up public keys in an LDAP directory server.

When you store certificates and certificate revocation lists (CRL) in an LDAP directory service, you must configure both how to access the directory service and also how to look up the certificates and CRLs, based on the fields in the certificates that OpenAM clients present to authenticate.

Access to the LDAP server and how to search for users is similar to LDAP module configuration as in [Section 2.3.13, “Hints for the LDAP Authentication Module”](#). The primary difference is that, unlike for LDAP configuration, OpenAM retrieves the user identifier from a field in the certificate that the client application presents, then uses that identifier to search for the LDAP directory entry that holds the certificate, which should match the certificate presented. For example, if the Subject field of a typical certificate has a DN `C=FR, O=Example Corp, CN=Barbara Jensen`, and Barbara Jensen's entry in the directory has `cn=Barbara Jensen`, then you can use `CN=Barbara Jensen` from the Subject DN to search for the entry with `cn=Barbara Jensen` in the directory.

**ssoadm** service name: `iPlanetAMAuthCertService`

Match Certificate in LDAP

When enabled, OpenAM searches for a match for the user's certificate in the LDAP directory. If a match is found and not revoked according to a CRL or OCSP validation, then authentication succeeds.

**ssoadm** attribute: `iplanet-am-auth-cert-check-cert-in-ldap`

Subject DN Attribute Used to Search LDAP for Certificates

Indicates which attribute and value in the certificate Subject DN is used to find the LDAP entry holding the certificate.

Default: CN

**ssoadm** attribute: `iplanet-am-auth-cert-attr-check-ldap`

Match Certificate to CRL

When enabled, OpenAM checks whether the certificate has been revoked according to a CRL in the LDAP directory.

**ssoadm** attribute: `iplanet-am-auth-cert-check-crl`

Issuer DN Attribute Used to Search LDAP for CRLs

Indicates which attribute and value in the certificate Issuer DN is used to find the CRL in the LDAP directory.

Default: CN

If only one attribute is specified, the LDAP search filter used to find the CRL based on the Subject DN of the CA certificate is (*attr-name=attr-value-in-subject-DN*).

For example, if the subject DN of the issuer certificate is `C=US, CN=Some CA, serialNumber=123456`, and the attribute specified is `CN`, then the LDAP search filter used to find the CRL is (`CN=Some CA`).

In order to distinguish among different CRLs for the same CA issuer, specify multiple attributes separated by commas (,) in the same order they occur in the subject DN. When multiple attribute names are provided in a comma-separated list, the LDAP search filter used is (`cn=attr1=attr1-value-in-subject-DN, attr2=attr2-value-in-subject-DN, ..., attrN=attrN-value-in-subject-DN`).

For example, if the subject DN of the issuer certificate is `C=US, CN=Some CA, serialNumber=123456`, and the attributes specified are `CN, serialNumber`, then the LDAP search filter used to find the CRL is (`cn=CN=Some CA, serialNumber=123456`).

**ssoadm** attribute: `iplanet-am-auth-cert-attr-check-crl`

HTTP Parameters for CRL Update

Your certificate authority should provide the URL to use here, from which OpenAM can get CRL updates.

**ssoadm** attribute: `iplanet-am-auth-cert-param-get-crl`

Match CA Certificate to CRL

When enabled, OpenAM checks the CRL against the CA certificate to ensure it has not been compromised.

**ssoadm** attribute: `sunAMValidateCACert`

Cache CRLs in Memory

When enabled, CRLs will be cached.

**ssoadm** attribute: `openam-am-auth-cert-attr-cache-crl`

Update CA CRLs from CRLDistributionPoint

When enabled, OpenAM updates CRLs from the LDAP directory store.

**ssoadm** attribute: `openam-am-auth-cert-update-crl`

OCSP Validation

Enable this to use Online Certificate Status Protocol (OCSP) instead of CRLs to check certificates' revocation status.

If you enable this, you also must configure OSCP for OpenAM under Configuration > Server and Sites > Default Server Settings, or Configuration > Server and Sites > *Server Name* > Security.

**ssoadm** attribute: `iplanet-am-auth-cert-check-ocsp`

LDAP Server Where Certificates are Stored

Identifies the LDAP server with certificates. Remember to specify URLs with appropriate port numbers (389 for unencrypted LDAP, 636 for LDAP over SSL). When configuring a secure connection, scroll down to enable Use SSL/TLS for LDAP Access.

**ssoadm** attribute: `iplanet-am-auth-cert-ldap-provider-url`

LDAP Search Start or Base DN

Valid base DN for the LDAP search, such as `dc=example,dc=com`.

**ssoadm** attribute: `iplanet-am-auth-cert-start-search-loc`

LDAP Server Authentication User, LDAP Server Authentication Password

If OpenAM stores attributes in the LDAP directory, for example to manage account lockout, or if the LDAP directory requires that OpenAM authenticate in order to read users' attributes, then OpenAM needs the DN and password to authenticate to the LDAP directory.

**ssoadm** attributes: `iplanet-am-auth-cert-principal-user`, and `iplanet-am-auth-cert-principal-passwd`

Use SSL/TLS for LDAP Access

If you use SSL/TLS for LDAP access, OpenAM must be able to trust the LDAP server certificate.

**ssoadm** attribute: `iplanet-am-auth-cert-use-ssl`

Certificate Field Used to Access User Profile

If the user profile is in a different entry from the user certificate, then this can be different from subject DN attribute used to find the entry with the

certificate. When you select other, provide an attribute name in the Other Certificate Field Used to Access User Profile text box.

**ssoadm** attribute: `iplanet-am-auth-cert-user-profile-mapper`

Other Certificate Field Used to Access User Profile

This field is only used if the Certificate Field Used to Access User Profile attribute is set to other. This field allows a custom certificate field to be used as the basis of the user search.

**ssoadm** attribute: `iplanet-am-auth-cert-user-profile-mapper-other`

SubjectAltNameExt Value Type to Access User Profile

Use this if you want to look up the user profile from an RFC 822 style name, or a User Principal Name as used in Active Directory.

**ssoadm** attribute: `iplanet-am-auth-cert-user-profile-mapper-ext`

Trusted Remote Hosts

Hosts trusted to send certificates to OpenAM, such as load balancers doing SSL termination, or OpenAM distributed authentication UI instances.

**ssoadm** attribute: `iplanet-am-auth-cert-gw-cert-auth-enabled`

HTTP Header Name for Client Certificate

If you configure trusted hosts, specify the HTTP header name for the client certificate inserted by the trusted host.

**ssoadm** attribute: `sunAMHttpParamName`

Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: `iplanet-am-auth-cert-auth-level`

## 2.3.5 Hints for the Core Authentication Module

The Core authentication module is a meta-module.

The Core module lets you set up the list of modules available, and specify what types of client applications can authenticate with which modules. It also lets you configure connection pools for access to directory servers, and whether to retain objects used during authentication for use during logout. Furthermore, the Core module lets you set defaults used when configuring authentication in a particular realm.

The Core Authentication module is divided into seven sections:

- [Section 2.3.5.1, “Core - Global Attributes”](#)
- [Section 2.3.5.2, “Core - Realm Attributes”](#)
- [Section 2.3.5.3, “Core - Persistent Cookie \(Legacy\)”](#)
- [Section 2.3.5.4, “Core - Account Lockout”](#)
- [Section 2.3.5.5, “Core - General”](#)
- [Section 2.3.5.6, “Core - Security”](#)
- [Section 2.3.5.7, “Core - Post Authentication Processing”](#)

**ssoadm** service name: `iPlanetAMAuthService`

### **2.3.5.1 Core - Global Attributes**

The Global Attributes includes the list of available modules, LDAP connection settings, authentication process options, and an option to disable the XUI and make the classic UI the default end user interface. The Global Attributes are defined in the [Authentication Configuration](#) section of the [Reference](#).

### **2.3.5.2 Core - Realm Attributes**

This section of the Core Authentication module is on the Realm Attributes section page before options for Persistent Cookies.

#### User Profile

Whether a user profile needs to exist in the user data store, or should be created on successful authentication.

#### Dynamic

Specifies that on successful authentication the Authentication Service creates a user profile if one does not already exist. OpenAM then issues the SSOToken. OpenAM creates the user profile in the user data store configured for the realm.

#### Dynamic with User Alias

Specifies that on successful authentication the Authentication Service creates a user profile that contains the User Alias List attribute which defines one or more aliases that for mapping a user's multiple profiles.

#### Ignored

Specifies that a user profile is not required for the Authentication Service to issue an SSOToken after a successful authentication.

Required

Specifies that on successful authentication the user must have a user profile in the user data store configured for the realm in order for the Authentication Service to issue an SSOToken.

**ssoadm** attribute: `iplanet-am-auth-dynamic-profile-creation`

User Profile Dynamic Creation Default Roles

Specifies the Distinguished Name (DN) of a role to be assigned to a new user whose profile is created when either of the Dynamic options is selected under the User Profile attribute. There are no default values. The role specified must be within the realm for which the authentication process is configured.

This role can be either an OpenAM or Sun DSEE role, but it cannot be a filtered role. If you wish to automatically assign specific services to the user, you have to configure the Required Services attribute in the User Profile.

**ssoadm** attribute: `iplanet-am-auth-default-role`

Alias Search Attribute Name

After a user is successfully authenticated, the user's profile is retrieved. OpenAM first searches for the user based on the data store settings. If that fails to find the user, OpenAM will use the attributes listed here to lookup the user profile. This setting accepts any data store specific attribute name.

**ssoadm** attribute: `iplanet-am-auth-alias-attr-name`

## Note

If the Alias Search Attribute Name property is empty, OpenAM uses the `iplanet-am-auth-user-naming-attr` property from the `iPlanetAmAuthService`. The `iplanet-am-auth-user-naming-attr` property is only configurable through the **ssoadm** command-line tool and not through the OpenAM console.

```
$ ssoadm get-realm-svc-attrs \
--adminid amadmin \
--password-file PATH_TO_PWDFILE \
--realm REALM \
--servicename iPlanetAMAuthService

$ ssoadm set-realm-svc-attrs \
--adminid amadmin \
--password-file PATH_TO_PWDFILE \
--realm REALM \
--servicename iPlanetAMAuthService \
--attributevalues iplanet-am-auth-user-naming-attr=SEARCH_ATTRIBUTE
```

### 2.3.5.3 Core - Persistent Cookie (Legacy)

This section of the Core Authentication module covers the Persistent Cookie options.

#### Note

Two methods are available in OpenAM to configure persistent cookies. The options described here and in [Section 2.6, “Authenticating To OpenAM”](#) specify one method. There is also a new module, described in [Section 2.3.18, “Hints for the Persistent Cookie Module”](#). If you want to set up persistent cookies, you are encouraged to use the new module. The options described here have no effect on that module.

##### Persistent Cookie Mode

Determines whether users can return to their authenticated session after restarting the browser. When enabled, the persistent cookie can be used to reauthenticate until the persistent cookie expires (as specified by the value of the Persistent Cookie Maximum Time attribute), or until the user explicitly logs out. By default, the Authentication Service uses only memory cookies (expires when the browser is closed).

The client must explicitly request a persistent cookie by adding `iPSPCookie=yes` as a parameter to the login URL. OpenAM sets a `DProPCookie` as described in [Section 2.6, “Authenticating To OpenAM”](#).

**ssoadm** attribute: `iplanet-am-auth-persistent-cookie-mode`

##### Persistent Cookie Maximum Time

Specifies the interval after which a persistent cookie expires. The interval begins when the user's session is successfully authenticated. The maximum value is 2147483647 (in seconds, so a bit more than 68 years). The field accepts any integer value less than the maximum.

**ssoadm** attribute: `iplanet-am-auth-persistent-cookie-time`

##### Persistent Cookie Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: `openam-auth-pcookie-auth-level`

#### Note

The distributed authentication service (DAS) and cross-domain single sign-on (CDSSO) do not support the `iPSPCookie/DProPCookie` query string

parameter to set a DProPCookie in the user-agent as a mechanism for cookie persistence. Neither DAS nor CDSSO retains iPSPCookie=yes.

### 2.3.5.4 Core - Account Lockout

This section of the Core Authentication module includes details on how account lockouts can be configured.

#### Login Failure Lockout Mode

Selecting this attribute enables a physical lockout. Physical lockout will inactivate an LDAP attribute (defined in the Lockout Attribute Name property) in the user's profile. This attribute works in conjunction with several other lockout and notification attributes.

**ssoadm** attribute: `iplanet-am-auth-login-failure-lockout-mode`

#### Login Failure Lockout Count

Defines the number of attempts that a user has to authenticate, within the time interval defined in Login Failure Lockout Interval, before being locked out.

**ssoadm** attribute: `iplanet-am-auth-login-failure-count`

#### Login Failure Lockout Interval

Defines the time in minutes during which failed login attempts are counted. If one failed login attempt is followed by a second failed attempt, within this defined lockout interval time, the lockout count starts, and the user is locked out if the number of attempts reaches the number defined in Login Failure Lockout Count. If an attempt within the defined lockout interval time proves successful before the number of attempts reaches the number defined in Login Failure Lockout Count, the lockout count is reset.

**ssoadm** attribute: `iplanet-am-auth-login-failure-duration`

#### Email Address to Send Lockout Notification

Specify one (or more) email address(es) to which notification is sent if a user lockout occurs.

Separate multiple addresses with spaces, and append `|locale|charset` to addresses for recipients in non-English locales.

**ssoadm** attribute: `iplanet-am-auth-lockout-email-address`

#### Warn User After N Failures

The number of authentication failures after which OpenAM displays a warning message that the user will be locked out.

**ssoadm** attribute: `iplanet-am-auth-lockout-warn-user`

#### Login Failure Lockout Duration

Defines how many minutes a user must wait after a lockout before attempting to authenticate again. Entering a value greater than 0 enables memory lockout and disables physical lockout. Memory lockout means the user's account is locked in memory for the number of minutes specified. The account is unlocked after the time period has passed.

**ssoadm** attribute: `iplanet-am-auth-lockout-duration`

#### Lockout Duration Multiplier

Defines a value with which to multiply the value of the Login Failure Lockout Duration attribute for each successive lockout. For example, if Login Failure Lockout Duration is set to 3 minutes, and the Lockout Duration Multiplier is set to 2, the user is locked out of the account for 6 minutes. Once the 6 minutes has elapsed, if the user again provides the wrong credentials, the lockout duration is then 12 minutes. With the Lockout Duration Multiplier, the lockout duration is incrementally increased based on the number of times the user has been locked out.

**ssoadm** attribute: `sunLockoutDurationMultiplier`

#### Lockout Attribute Name

Defines the LDAP attribute used for physical lockout. The default value is `inetuserstatus`, although the field in the OpenAM console is empty. The Lockout Attribute Value field must also contain an appropriate value.

**ssoadm** attribute: `iplanet-am-auth-lockout-attribute-name`

#### Lockout Attribute Value

Specifies the action to take on the attribute defined in Lockout Attribute Name. The default value is `inactive`, although the field in the OpenAM console is empty. The Lockout Attribute Name field must also contain an appropriate value.

**ssoadm** attribute: `iplanet-am-auth-lockout-attribute-value`

#### Invalid Attempts Data Attribute Name

Specifies the LDAP attribute used to hold the number of failed authentication attempts towards Login Failure Lockout Count.

**ssoadm** attribute: `sunAMAuthInvalidAttemptsDataAttrName`

#### Store Invalid Attempts in Data Store

Enables the storage of information regarding failed authentication attempts as the value of the Invalid Attempts Data Attribute Name in the user data store. In order to store data in this attribute, the OpenAM schema has to be loaded. Information stored includes number of invalid attempts, time of last failed attempt, lockout time and lockout duration. Storing this information

in the identity repository allows it to be shared among multiple instances of OpenAM.

**ssoadm** attribute: sunStoreInvalidAttemptsInDS

### 2.3.5.5 Core - General

This section of the Core Authentication module includes general options.

#### Default Authentication Locale

Specifies the default language subtype to be used by the Authentication Service. The default value is en\_US.

**ssoadm** attribute: iplanet-am-auth-locale

#### Identity Types

Lists the type or types of identities used during a profile lookup. You can choose more than one to search on multiple types if you would like OpenAM to conduct a second lookup if the first lookup fails. Default: Agent and User

##### Agent

Searches for identities under your agents.

##### agentgroup

Searches for identities according to your established agent group.

##### agentonly

Searches for identities only under your agents.

##### Group

Searches for identities according to your established groups.

##### User

Searches for identities according to your users.

**ssoadm** attribute: sunAMIdentityType

#### Pluggable User Status Event Classes

Specifies one or more Java classes used to provide a callback mechanism for user status changes during the authentication process. The Java class must implement the com.sun.identity.authentication.spi.AMAuthCallBack OpenAM interface. OpenAM supports account lockout and password changes. OpenAM supports password changes through the LDAP authentication module, and so the feature is only available for the LDAP module.

A .jar containing the user status event class belongs in the WEB-INF/lib directory of the deployed OpenAM instance. If you do not build a .jar, add the class files under WEB-INF/classes.

**ssoadm** attribute: sunAMUserStatusCallbackPlugins

Default Authentication Level

Specifies the default authentication level for authentication modules.

**ssoadm** attribute: iplanet-am-auth-default-auth-level

## 2.3.5.6 Core - Security

This section of the Core Authentication module includes basic security options.

Module Based Authentication

Enables users to authenticate using module-based authentication. Otherwise, all attempts at authentication using the `module=module-name` login parameter result in failure. It is recommended that this be turned off in a production environment.

**ssoadm** attribute: sunEnableModuleBasedAuth

Zero Page Login

If enabled, allow users to authenticate using only GET request parameters without showing a login screen. Enable this with caution as browsers can cache and servers can log credentials when they are part of the URL.

OpenAM always allows HTTP POST requests for zero page login.

Default: false (disabled)

**ssoadm** attribute: openam.auth.zero.page.login.enabled

Zero Page Login Referer Whitelist

List of HTTP Referer URLs for which OpenAM allows zero page login. These URLs are supplied in the HTTP request header, Referer, which is designed to allow the client to specify the web page that provided the link to the requested resource.

If you enable zero page login, include the URLs here for the pages from which to allow zero page login, or leave this list blank to allow zero page login from any Referer.

This setting applies for both HTTP GET and also HTTP POST requests for zero page login.

**ssoadm** attribute: openam.auth.zero.page.login.referer.whitelist

Zero Page Login Allowed without Referer?

If enabled and zero page login is enabled, allow zero page login for requests without an HTTP Referer request header.

Default: true (enabled)

**ssoadm** attribute: openam.auth.zero.page.login.allow.null.referer

### 2.3.5.7 Core - Post Authentication Processing

This section of the Core Authentication module specifies options for post authentication processing.

Default Success Login URL

Accepts a list of values that specifies where users are directed after successful authentication. The format of this attribute is *client-type|URL* although the only value you can specify at this time is a URL which assumes the type HTML. The default value is /openam/console. Values that do not specify HTTP have that appended to the deployment URI.

**ssoadm** attribute: iplanet-am-auth-login-success-url

Default Failure Login URL

Accepts a list of values that specifies where users are directed after authentication has failed. The format of this attribute is *client-type|URL* although the only value you can specify at this time is a URL which assumes the type HTML. Values that do not specify HTTP have that appended to the deployment URI.

**ssoadm** attribute: iplanet-am-auth-login-failure-url

Authentication Post Processing Classes

Specifies one or more Java classes used to customize post authentication processes for successful or unsuccessful logins. The Java class must implement the com.sun.identity.authentication.spi. AMPostAuthProcessInterface OpenAM interface.

A .jar containing the post processing class belongs in the WEB-INF/lib directory of the deployed OpenAM instance. If you do not build a .jar, add the class files under WEB-INF/classes. For deployment, add the .jar or classes into a custom OpenAM .war file.

**ssoadm** attribute: iplanet-am-auth-post-login-process-class

Generate UserID Mode

When enabled, the Membership module generates a list of alternate user identifiers if the one entered by a user during the self-registration process is not valid or already exists. The user identifiers are generated by the class specified in the Pluggable User Name Generator Class property.

**ssoadm** attribute: iplanet-am-auth-username-generator-enabled

**Pluggable User Name Generator Class**

Specifies the name of the class used to generate alternate user identifiers when Generate UserID Mode is enabled. The default value is com.sun.identity.authentication.spi.DefaultUserIDGenerator.

**ssoadm** attribute: iplanet-am-auth-username-generator-class

**User Attribute Mapping to Session Attribute**

Enables the authenticating user's identity attributes (stored in the identity repository) to be set as session properties in the user's SSOToken. The value takes the format *User-Profile-Attribute|Session-Attribute-Name*. If *Session-Attribute-Name* is not specified, the value of *User-Profile-Attribute* is used. All session attributes contain the am.protected prefix to ensure that they cannot be edited by the Client SDK.

For example, if you define the user profile attribute as mail and the user's email address (available in the user session) as user.mail, the entry for this attribute would be mail|user.mail. After a successful authentication, the SSOToken.getProperty(String) method is used to retrieve the user profile attribute set in the session. The user's email address is retrieved from the user's session using the SSOToken.getProperty("am.protected.user.mail") method call.

Properties that are set in the user session using User Attribute Mapping to Session Attributes can not be modified (for example, SSOToken.setProperty(String, String)). This results in an SSOException. Multi-value attributes, such as memberOf, are listed as a single session variable with a | separator.

**ssoadm** attribute: sunAMUserAttributesSessionMapping

## 2.3.6 Hints for the Data Store Authentication Module

The Data Store authentication module allows a login using the Identity Repository of the realm to authenticate users. Using the Data Store module removes the requirement to write an authentication plug-in module, load, and then configure the authentication module if you need to authenticate against the same data store repository. Additionally, you do not need to write a custom authentication module where flat-file authentication is needed for the corresponding repository in that realm.

Yet, the Data Store module is generic. It does not implement data store-specific capabilities such as the password policy and password reset features provided by LDAP modules. Therefore the Data Store module returns failure when such capabilities are invoked.

**ssoadm** service name: sunAMAuthDataStoreService

**Authentication Level**

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: sunAMAuthDataStoreAuthLevel

### 2.3.7 Hints for the Device ID (Match) Authentication Module

The Device ID (Match) module provides device fingerprinting functionality for risk-based authentication. The Device ID (Match) works together with other modules, for example, the DataStore, HMAC One-Time Password (HOTP), and Device ID (Save) modules in an authentication chain.

The Device ID (Match) module collects the unique characteristics of a remote user's computing device and compares them to characteristics on a saved device profile. The module computes any variances between the collected characteristics to those stored on the saved device profile and assigns penalty points for each difference.

For example, when a user who typically logs on to the Internet on a FireFox browser, logs on using Chrome, the Device ID (Match) module notes the difference and assigns penalty points to this change in behavior. If the module detects additional differences in behavior, such as browser fonts, geolocation, etc., then additional points are assessed and calculated. If the total maximum number of penalty points exceeds a pre-configured threshold value, the Device ID (Match) module fails and control is determined by how you configured your authentication chain. For example, if you include the HMAC One-Time Password (HOTP) module in your authentication chain and if the Device ID (Match) module fails after the maximum number of penalty points have been exceeded, then the authentication chain issues a HOTP request to the user, requiring the user to identify himself using two-factor authentication.

Typically, you configure and gather the following device print items:

- User agents, associated with the configuration of a web browser
- Installed fonts
- The plugins installed for the web browser
- The resolution and color depth associated with a display
- The timezone or even the geolocation of a device

The Device ID (Match) module comes out of the box with default client-side and server-side JavaScript code, supplying the logic necessary to fingerprint the user agent and computer. Scripting allows you to customize the code, providing more control over the device fingerprint elements that you would like to collect. OpenAM supports both the JavaScript (default) and Groovy languages. The

## Hints for the Device ID (Match) Authentication Module

client-side code must be in JavaScript. The server-side script can be written in JavaScript or Groovy.

### Important

By default, the maximum penalty points is set to 0, which you can adjust in the server-side script.

To access the Device ID (Match) module page on the console, click Configuration > Device Id (Match).

**Figure 2.5. Device ID (Match) Module**

The screenshot shows the 'Device Id (Match)' configuration page. At the top, there are three buttons: 'Save', 'Reset', and 'Back to Service Configuration'. Below these are sections for 'Realm Attributes' and 'Client-side Script Enabled'. The 'Client-side Script Enabled' checkbox is checked. A detailed description of the client-side script is provided, including its purpose (detecting font availability), author (Lalit Patel), website (http://www.lalit.org/lab/javascript-css-font-detect/), license (Apache Software License 2.0), version history (0.15 to 0.3), and changes made. Below this is a text area for the client-side script, with an 'Upload' button. Under 'Server-side Script Language', 'JavaScript' is selected. A note indicates the language of the server-side script. The 'Server-side Script' section contains a large text area with a CDDL header, copyright information, and a note about distributing the code. An 'Upload' button is available for this script. At the bottom, there is an 'Authentication Level' input field set to '0' and a note about the authentication level. Finally, at the very bottom are the 'Save', 'Reset', and 'Back to Service Configuration' buttons.

Device Id (Match)

Realm Attributes

Client-side Script Enabled:  Enabled

Enable this setting if the client-side script should be executed.

Client-side Script:

```
var fontDetector = (function () {  
    /*  
     * JavaScript code to detect available availability of a  
     * particular font in a browser using JavaScript and CSS.  
     */  
    // Author : Lalit Patel  
    // Website : http://www.lalit.org/lab/javascript-css-font-detect/  
    // License : Apache Software License 2.0  
    // http://www.apache.org/licenses/LICENSE-2.0  
    // Version: 0.15 (21 Sep 2009)  
    // Changed comparison font to default from sans-default-default,  
    // as in FF3.0 font of child element didn't fallback  
    // to parent's font if the font is missing.  
    // Version: 0.2 (04 Mar 2012)  
    // Comparing font against all the 3 generic font families ie,  
    // 'monospace', 'sans-serif' and 'sans'. If it doesn't match all 3  
    // then that font is 100% not available in the system  
    // Version: 0.3 (24 Mar 2012)  
    // Replaced sans with serif in the list of baseFonts  
}/
```

The client-side script.

Upload

Server-side Script Language:  Groovy  JavaScript

The language the server-side script is written in.

Server-side Script:

```
/  
* DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS HEADER.  
* Copyright (c) 2009 Sun Microsystems Inc. All Rights Reserved  
*  
* The contents of this file are subject to the terms  
* of the Common Development and Distribution License  
* (the License). You may not use this file except in  
* compliance with the License.  
*  
* You can obtain a copy of the License at  
* https://openssouserdev.java.net/public/CDDLv1.0.html or  
* opensso/legal/CDDLv1.0.txt  
* See the License for the specific language governing  
* permission and limitations under the License.  
*  
* When distributing Covered Code, include this CDDL  
* Header Notice in each file and include the License file  
* at opensso/legal/CDDLv1.0.txt.  
* If applicable, add the following below the CDDL Header.  
*/
```

The server-side script to execute.

Upload

Validate

Authentication Level: 0

The authentication level associated with the authentication module.

Save Reset Back to Service Configuration

## Caution

The Device ID (Match) module's default JavaScript client-side and server-side scripts are fully functional. If you change the client-side script, you must also make a corresponding change to the server-side script. For a safer option if you want to change the behavior of the module, you can copy the scripts to a custom scripted module, and edit the code to make your own customized version.

The Device ID (Match) does not stand on its own within an authentication chain and requires additional modules. For example, you can have any module that identifies the user (for example, DataStore, Active Directory or others), Device ID (Match), any module that provides two-factor authentication, (for example, HOTP or SecurID), and Device ID (Save) within your authentication chain.

As an example, you can configure the following modules with the specified criteria:

1. **DataStore - Requisite.** The Device ID (Match) module requires user authentication information to validate the username. You can also use other modules that identify the username, such as Active Directory, RADIUS, or Windows NT.
2. **Device ID (Match) - Sufficient.** The Device ID (Match) runs the client-side script, which invokes the device fingerprint collectors, captures the data, and converts it into a JSON string. It then auto-submits the data in a JSP page to the server-side scripting engine.

The server-side script calculates the penalty points based on differences between the client device and stored device profile, and determines if the client device successfully "matches" the stored profile. If a match is successful, OpenAM determines that the client's device has the required attributes for a successful authentication.

If the device does not have a match, then the module fails and falls through to the HMAC One-Time Password (HOTP) module for further processing.

3. **HMAC One-Time Password (HOTP) - Requisite.** If the user's device does not match a stored profile, OpenAM presents the user with a HMAC One-Time Password (HOTP) screen either by SMS or email, prompting the user to enter a password.

You can also use any other module that provides two-factor authentication.

After the HOTP has successfully validated the user, the Device ID (Save) module gathers additional data from the user. For specific information about the HOTP module, see [Section 2.3.10, "Hints for the HOTP Authentication Module"](#).

**4. Device ID (Save) - Required.** The Device ID (Save) module provides configuration options to enable an auto-save feature on the device profile as well as set a maximum number of stored device profiles on the user entry or record. Once the maximum number of stored device profiles is reached, OpenAM deletes the old data from the user record as new ones are added. User records could thus contain both old and new device profiles.

If the auto-save feature is not enabled, OpenAM presents the user with a screen to save the new device profile.

The module also takes the device print and creates a JSON object that includes the ID, name, last selected date, selection counter, and device print. For specific information about the Device ID (Save) module, see [Section 2.3.8, "Hints for the Device ID \(Save\) Module"](#).

## Note

If a user has multiple device profiles, the profile that is the closest match to the current client details is used for the comparison result.

You can create modules instances for a particular realm by clicking Access Control > *Realm Name* > Authentication, and then click New under Module Instances to add Device Id (Match) and Device Id (Save).

Under Authentication Chaining, click New to create a new chain, and then enter a name for the chain. On the *Chain Name* - Properties page, click Add to set up the modules for your authentication chain.

**Figure 2.6. Authentication Chain with Device ID (Match) and Device ID (Save)**

| (4 item(s))                                   |            |         |  |
|---|------------|---------|--|
| Add   | Remove     | Reorder |  |
| Instance                                      | Criteria   | Options |  |
| <input checked="" type="checkbox"/> DataStore | REQUISITE  |         |  |
| <input type="checkbox"/> Device-Id-Match      | SUFFICIENT |         |  |
| <input type="checkbox"/> HOTP                 | REQUISITE  |         |  |
| <input type="checkbox"/> Device-Id-Save       | REQUIRED   |         |  |

The Device ID (Match) module has the following properties:

### Client-side Script Enabled

Enable Device ID (Match) to send JavaScript in an authentication page to the device to collect data about the device by a self-submitting form.

**ssoadm** attribute: `iplanet-am-auth-scripted-client-script-enabled`

#### Client-side Script

You can see default client-side JavaScript code that you can modify if necessary. Note that if you change the client-side script, you must make a corresponding change in the server-side script to account for the specific addition or removal of an element.

**ssoadm** attribute: `iplanet-am-auth-scripted-client-script`

#### Server-side Script Language

Default: JavaScript. Select Groovy if your script is written in that language.

**ssoadm** attribute: `iplanet-am-auth-scripted-script-type`

#### Server-side Script

You can see default server-side JavaScript code that you can modify if necessary. Note that a change in the client-side script requires a corresponding change in the server-side script to account for the specific addition or removal of an element.

**ssoadm** attribute: `iplanet-am-auth-scripted-server-script`

#### Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

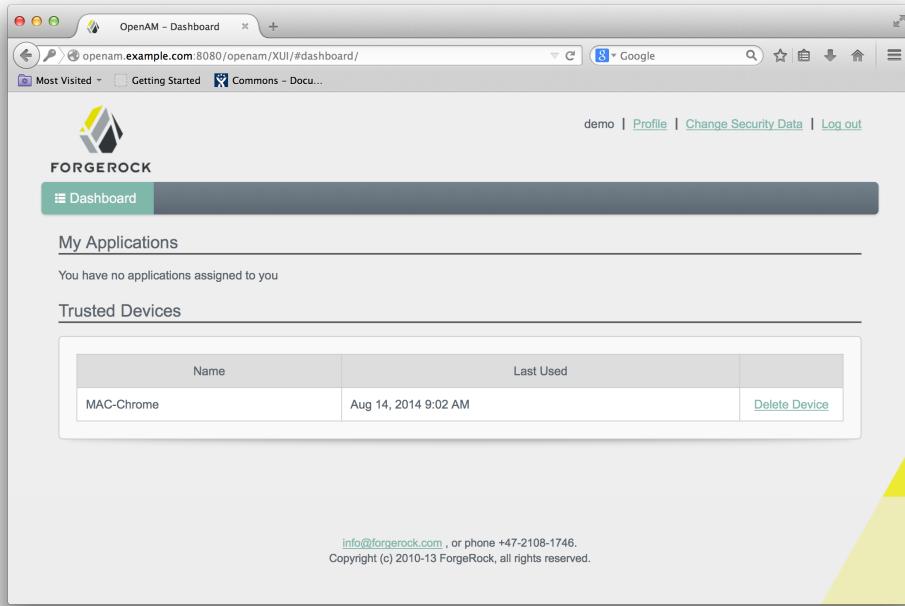
**ssoadm** attribute: `iplanet-am-auth-scripted-auth-level`

When the user logs on to the console, OpenAM determines if the user's device differs from that of the stored profile. If the differences exceed the maximum number of penalty points, OpenAM sends an "Enter OTP" page, requiring the user to enter a one-time password, which is sent to the user via email or SMS. The user also has the option to request a one-time password.

Next, OpenAM presents the user with a "Add to Trusted Devices?" page, asking if the user wants to add the device to the list of trusted device profiles. If the user clicks "Yes", OpenAM prompts the user to enter a descriptive name for the trusted device.

Next, OpenAM presents the user with the User Profile page, where the user can click the Dashboard link in the upper left corner to access the My Applications and Trusted Devices page. Once on the Dashboard, the user can view the list of trusted devices or remove the device by clicking the Delete Device link.

**Figure 2.7. Trusted Device Management on the User Profile Page**



### 2.3.8 Hints for the Device ID (Save) Module

The Device ID (Save) module auto-saves the device profile rather than presenting a Save screen to the user each time a device profile requires update. The module also provides a configuration setting for the maximum number of stored profiles on the user record.

#### Note

If a user has multiple device profiles, the profile that is the closest match to the current client details is used for the comparison result.

Within its configured authentication chain, the Device ID (Save) module also takes the device print and creates a JSON object that consists of the ID, name, last selected date, selection counter, and device print itself.

The Device ID (Save) module has the following properties:

Automatically store new profiles

Select the checkbox to automatically store new profiles. After successful HMAC one-time password (HOTP) confirmation, OpenAM stores the new profile automatically.

**ssoadm** attribute: `iplanet-am-auth-device-id-save-auto-store-profile`

Maximum stored profile quantity

Sets the maximum number of stored profiles on the user's record.

**ssoadm** attribute: `iplanet-am-auth-device-id-save-max-profiles-allowed`

Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: `iplanet-am-auth-device-id-save-auth-level`

### 2.3.9 Hints for the Federation Authentication Module

The Federation authentication module is used by a service provider to create a user session after validating single sign-on protocol messages. This authentication module is used by the SAML, SAMLv2, ID-FF, and WS-Federation protocols.

**ssoadm** service name: `sunAMAuthFederationService`

Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: `sunAMAuthFederationAuthLevel`

### 2.3.10 Hints for the HOTP Authentication Module

The HMAC One-Time Password authentication module works together in an authentication chain with any module that stores the `username` attribute. The module uses the `username` from the `sharedState` set by the previous module in the chain and retrieves the user's email address or telephone number to send a one-time password to the user. The user then enters the password on a Login page and completes the authentication process if successful.

For example, to set up HOTP in an authentication chain, you can configure the Data Store module (or any module that stores the user's `username`) as the requisite first module, and the HOTP module as the second requisite module.

When authentication succeeds against the Data Store module, the HOTP module retrieves the Email Address and Telephone Number attributes from the data store based on the `username` value. For the HOTP module to use either attribute, the Email Address must contain a valid email address, or the Telephone Number must contain a valid SMS telephone number.

You can set the HOTP module to automatically generate a password when users begin logging into the system. You can also setup mobile phone, mobile carrier, and email attributes for tighter controls over where the messages are generated and what provider the messages go through to reach the user.

**ssoadm** service name: `sunAMAuthHOTPSERVICE`

**Authentication Level**

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: `sunAMAuthHOTPAuthLevel`

**SMS Gateway Implementation Class**

Change this if you must customize the SMS gateway implementation. The default class sends an SMS or email, depending on the configuration.

**ssoadm** attribute: `sunAMAuthHOTPSMSGatewayImplClassName`

**Mail Server Host Name**

Host name of the mail server supporting Simple Message Transfer Protocol for electronic mail.

**ssoadm** attribute: `sunAMAuthHOTPSMTPHostName`

**Mail Server Host Port**

The default outgoing mail server port is 25, 465 (when connecting over SSL).

**ssoadm** attribute: `sunAMAuthHOTPSMTPHostPort`

**Mail Server Authentication Username**

User name for OpenAM to connect to the mail server.

**ssoadm** attribute: `sunAMAuthHOTPSMTPUserName`

**Mail Server Authentication Password**

Password for OpenAM to connect to the mail server.

**ssoadm** attribute: `sunAMAuthHOTPSMTPUserPassword`

**Mail Server Secure Connection**

If OpenAM connects to the mail server securely, OpenAM must be able to trust the server certificate.

**ssoadm** attribute: sunAMAuthHOTPSMTPSSLEnabled

Email From Address

The From: address when sending a one-time password by mail.

**ssoadm** attribute: sunAMAuthHOTPSMTPFromAddress

One Time Password Validity Length (in minutes)

One-time passwords are valid for 5 minutes after they are generated by default.

**ssoadm** attribute: sunAMAuthHOTPPasswordValidityDuration

One Time Password Length (in digits)

Set the length of the one-time password to 6 or 8 digits.

**ssoadm** attribute: sunAMAuthHOTPPasswordLength

One Time Password Delivery

Send the one-time password by SMS, by mail, or both.

**ssoadm** attribute: sunAMAuthHOTPPasswordDelivery

Mobile Phone Number Attribute Names

Provides the attribute name used for the text message. The default value is telephoneNumber.

**ssoadm** attribute: openamTelephoneAttribute

Mobile Carrier Attribute Name

Provides the name of the carrier that will send the text message.

Every carrier has their own attribute name ending, for example Verizon uses @vtext.com or vtext.com. Contact your mobile carrier to find out what their attribute name is. If you will be sending international texts, ask your carrier if a country code is required.

**ssoadm** attribute: openamSMSCarrierAttribute

Email Attribute Name

Provides the attribute name used to email the OTP. The default value is mail (email).

**ssoadm** attribute: openamEmailAttribute

Auto Send OTP Code

Setup the HOTP module to automatically generate an email or text message when users begin the login process.

**ssoadm** attribute: sunAMAuthHOTPAutoClicking

### 2.3.11 Hints for the HTTP Basic Authentication Module

HTTP basic authentication takes a user name and password from HTTP authentication and tries authentication against the backend module in OpenAM, depending on what you configure as the Backend Module Name.

**ssoadm** service name: `iPlanetAMAuthHTTPBasicService`

Backend Module Name

Specifies the module that checks the user credentials. The credentials are then supplied to either a Data Store or other identity repository module for authentication.

**ssoadm** attribute: `iplanet-am-auth-http-basic-module-configured`

Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: `iplanet-am-auth-httpbasic-auth-level`

### 2.3.12 Hints for the JDBC Authentication Module

The Java Database Connectivity (JDBC) module lets OpenAM connect to a database such as MySQL or Oracle DB to authenticate users.

**ssoadm** service name: `sunAMAuthJDBCSERVICE`

Connection Type

Choose Connection pool is retrieved via JNDI to connect using the Java Naming and Directory Interface connection pool supported by the web container in which OpenAM runs. Choose Non-persistent JDBC connection to connect directly through the JDBC driver.

**ssoadm** attribute: `sunAMAuthJDBCConnectionType`

Connection Pool JNDI Name

When using Connection pool is retrieved via JNDI, this specifies the pool. How you configure connection pooling depends on the web container where you run OpenAM. Refer to the documentation for your web container for instructions on setting up connection pooling.

**ssoadm** attribute: `sunAMAuthJDBCJndiName`

JDBC Driver

When using Non-persistent JDBC connection, this specifies the JDBC driver provided by the database.

The jar containing the JDBC driver belongs in the WEB-INF/lib directory of the deployed OpenAM instance, and so you should add it to a custom OpenAM .war file that you deploy.

**ssoadm** attribute: sunAMAuthJDBCDriver

JDBC URL

When using Non-persistent JDBC connection, this specifies the URL to connect to the database.

**ssoadm** attribute: sunAMAuthJDBCUrl

Database Username

Specify the user name to open the database connection.

**ssoadm** attribute: sunAMAuthJDBCDbuser

Database Password

Specify the password for the user opening the database connection.

**ssoadm** attribute: sunAMAuthJDBCDbpassword

Password Column Name

Specify the database column name where passwords are stored.

**ssoadm** attribute: sunAMAuthJDBCPasswordColumn

Prepared Statement

Specify the SQL query to return the password corresponding to the user to authenticate.

**ssoadm** attribute: sunAMAuthJDBCStatement

Class to Transform Password Syntax

Specify the class that transforms the password retrieved to the same format as provided by the user.

The default class expects the password in clear text. Custom classes must implement the JDBCPasswordSyntaxTransform interface.

**ssoadm** attribute: sunAMAuthJDBCPasswordSyntaxTransformPlugin

Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: sunAMAuthJDBCAuthLevel

## 2.3.13 Hints for the LDAP Authentication Module

OpenAM connects to directory servers using Lightweight Directory Access Protocol (LDAP). To build an easy-to-manage, high performance, pure Java, open source directory service, try [OpenDJ](#) directory services.

**ssoadm** service name: iPlanetAMAuthLDAPService

Primary LDAP Server

Secondary LDAP Server

Directory servers generally use built-in data replication for high availability. Thus a directory service likely consists of a pool of replicas to which OpenAM can connect to retrieve and update directory data. You set up primary and secondary servers in case a replica is down due to maintenance or to a problem with a particular server.

Set one primary and optionally one secondary directory server for each OpenAM server. For the current OpenAM server, specify each directory server as a *host:port* combination. For other OpenAM servers in the deployment, you can specify each directory server as *server-name|host:port*, where *server-name* is the FQDN portion of the OpenAM server from the list under Configuration > Servers and Sites, and *host:port* identifies the directory server.

For example, if the *server-name* that is listed is `http://openam.example.com:8080/openam` and the directory server is accessible at `opendj.example.com:1389`, you would enter `openam.example.com|opendj.example.com:1389`.

When authenticating users from a directory service that is remote from OpenAM, set both the primary and secondary server values.

If you want to use SSL or TLS for security, then scroll down to enable SSL/TLS Access to LDAP Server. Make sure that OpenAM can trust the servers' certificates when using this option.

**ssoadm** attributes: primary is `iplanet-am-auth-ldap-server`, secondary is `iplanet-am-auth-ldap-server2`, and `iplanet-am-auth-ldap-ssl-enabled`

DN to Start User Search

LDAP data is organized hierarchically, a bit like a file system on Windows or UNIX. More specific DNs likely result in better search performance. When configuring the module for a particular part of the organization, you can perhaps start searches from a specific organizational unit such as `ou=sales,dc=example,dc=com`.

If multiple entries exist with identical search attribute values, make this value specific enough to return only one entry.

**ssoadm** attribute: `iplanet-am-auth-ldap-base-dn`

### Bind User DN, Bind User Password

If OpenAM stores attributes in the directory, for example to manage account lockout, or if the directory requires that OpenAM authenticate in order to read users' attributes, then OpenAM needs the DN and password to authenticate to the directory.

The default is `cn=Directory Manager`. Make sure that password is correct before you logout. If it is incorrect, you will be locked out. If this should occur, you can login with the super user DN, which by default is `uid=amAdmin, ou=People,OpenAM-deploy-base`, where `OpenAM-deploy-base` you set during OpenAM configuration.

**ssoadm** attributes: `iplanet-am-auth-ldap-bind-dn`, `iplanet-am-auth-ldap-bind-passwd`

### Attributes Used to Retrieve User Profile

#### Attributes Used to Search for a User to be Authenticated

##### User Search Filter

##### Search Scope

LDAP searches for user entries return entries with attribute values matching the filter you provide. For example if you search under `ou=people,dc=example,dc=com` with a filter "`(mail=bjensen@example.com)`", then the directory returns the entry that has `mail=bjensen@example.com`. In this example the attribute used to search for a user is `mail`. Multiple attribute values mean the user can authenticate with any one of the values. For example, if you have both `uid` and `mail`, then Barbara Jensen can authenticate with either `bjensen` or `bjensen@example.com`.

Should you require a more complex filter for performance, you add that to the User Search Filter text box. For example, if you search on `mail` and add User Search Filter (`objectClass/inetOrgPerson`), then OpenAM uses the resulting search filter `(&(mail=address)(objectClass=inetOrgPerson))`, where `address` is the mail address provided by the user.

Scope OBJECT means search only the entry specified as the DN to Start User Search, whereas ONELEVEL means search only the entries that are directly children of that object. SUBTREE means search the entry specified and every entry under it.

**ssoadm** attributes: `iplanet-am-auth-ldap-user-naming-attribute`, `iplanet-am-auth-ldap-user-search-attributes`, `iplanet-am-auth-ldap-search-filter`, and `iplanet-am-auth-ldap-search-scope`

### SSL/TLS Access to LDAP Server

If you enable SSL/TLS, OpenAM must be able to trust LDAP certificates, either because the 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: `iplanet-am-auth-ldap-ssl-enabled`

Return User DN to Authenticate

When enabled, and OpenAM uses the directory service as the user store, the module returns the DN rather than the User ID, so the bind for authentication can be completed without a search to retrieve the DN.

**ssoadm** attribute: `iplanet-am-auth-ldap-return-user-dn`

User Creation Attributes

This list lets you map (external) attribute names from the LDAP directory server to (internal) attribute names used by OpenAM.

**ssoadm** attribute: `iplanet-am-ldap-user-creation-attr-list`

Minimum Password Length

Specify the minimum acceptable password length.

**ssoadm** attribute: `iplanet-am-auth-ldap-min-password-length`

LDAP Behera Password Policy Support

When enabled, support interoperability with servers that implement the Internet-Draft, [Password Policy for LDAP Directories](#).

Support for this Internet-Draft is limited to the LDAP authentication module. Other components of OpenAM, such as the password change functionality in the /idm/EndUser page, do not support the Internet-Draft. In general, outside of the LDAP authentication module, OpenAM binds to the directory server as an administrator, such as Directory Manager. When OpenAM binds to the directory server as an administrator rather than as an end user, many features of the Internet-Draft password policies do not apply.

**ssoadm** attribute: `iplanet-am-auth-ldap-behera-password-policy-enabled`

Trust All Server Certificates

When enabled, blindly trust server certificates, including self-signed test certificates.

**ssoadm** attribute: `iplanet-am-auth-ldap-ssl-trust-all`

LDAP Connection Heartbeat Interval

Specifies how often OpenAM should 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.

Default: 1

**ssoadm** attribute: openam-auth-ldap-heartbeat-interval

LDAP Connection Heartbeat Time Unit

Specifies the time unit corresponding to LDAP Connection Heartbeat Interval.

Default: minute

**ssoadm** attribute: openam-auth-ldap-heartbeat-interval

Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: iplanet-am-auth-ldap-auth-level

### 2.3.14 Hints for the MSISDN Authentication Module

The Mobile Station Integrated Services Digital Network (MSISDN) authentication module enables non-interactive authentication using a mobile subscriber ISDN associated with a terminal such as a mobile phone. The module checks the subscriber ISDN against the value found on a user's entry in an LDAP directory service.

**ssoadm** service name: sunAMAuthMSISDNService

Trusted Gateway IP Address

Specifies a list of IP addresses of trusted clients that can access MSISDN modules. Either restrict the clients allowed to access the MSISDN module by adding each IPv4 or IPv6 address here, or leave the list empty to allow all clients to access the module. If you specify the value none, no clients are allowed access.

**ssoadm** attribute: sunAMAuthMSISDNTrustedGatewayList

MSISDN Number Search Parameter Name

Specifies a list of parameter names that identify which parameters to search in the request header or cookie header for the MSISDN number. For example, if you define x-Cookie-Param, AM\_NUMBER, and COOKIE-ID, the MSISDN authentication service checks those parameters for the MSISDN number.

**ssoadm** attribute: sunAMAuthMSISDNParameterNameList

LDAP Server and Port

If you want to use SSL or TLS for security, then scroll down to enable SSL/TLS Access to LDAP. Make sure that OpenAM can trust the servers' certificates when using this option.

**ssoadm** attribute: sunAMAuthMSISDNLdapProviderUrl

LDAP Start Search DN

Specify the DN of the entry where the search for the user's MSISDN number should start.

**ssoadm** attribute: sunAMAuthMSISDNBaseDn

Attribute To Use To Search LDAP

Specify the name of the attribute in the user's profile that contains the MSISDN number to search for the user. The default is sunIdentityMSISDNNumber.

**ssoadm** attribute: sunAMAuthMSISDNUserSearchAttribute

LDAP Server Authentication User

If OpenAM must authenticate to the directory server in order to search, then specify the bind DN. The default is cn=amldapuser,ou=DSAME Users,dc=example,dc=com.

**ssoadm** attribute: sunAMAuthMSISDNPrincipalUser

LDAP Server Authentication Password

Specify the password corresponding to the bind DN.

**ssoadm** attribute: sunAMAuthMSISDNPrincipalPasswd

SSL/TLS for LDAP Access

If you choose to enable SSL or TLS, then make sure that OpenAM can trust the servers' certificates.

**ssoadm** attribute: sunAMAuthMSISDNUseSsl

MSISDN Header Search Attribute

Specify the headers to use for searching the request for the MSISDN number.

- Cookie Header tells OpenAM to search the cookie.
- Request Header tells OpenAM to search the request header.
- Request Parameter tells OpenAM to search the request parameters.

**ssoadm** attribute: sunAMAuthMSISDNHeaderSearch

LDAP Attribute Used to Retrieve User Profile

Specify the LDAP attribute that is used during a search to return the user profile for MSISDN authentication service. The default is uid.

**ssoadm** attribute: sunAMAuthMSISDNUserNamingAttribute

Return User DN to DataStore

Enable this option only when the OpenAM directory is the same as the directory configured for MSISDN searches. When enabled, this option allows the authentication module to return the DN instead of the User ID. OpenAM thus does not need to perform an additional search with the user ID to find the user's entry.

**ssoadm** attribute: sunAMAuthMSISDNReturnUserDN

Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: sunAMAuthMSISDNAuthLevel

### 2.3.15 Hints for the OATH Module

The Open Authentication (OATH) module provides a more secure method for users to access their accounts with the help of a device, such as their mobile phone or Yubikey. Users can log into OpenAM and update their information more securely from a one-time password (OTP) displayed on their device. The OATH module includes the [OATH standard protocols \(RFC 4226 and RFC 6238\)](#). The OATH module has several enhancements to the HMAC One-Time Password (HOTP) Authentication Module, but does not replace the original module for those already using HOTP prior to the 10.1.0 release. The OATH module includes HOTP authentication and Time-Based One-Time Password (TOTP) authentication. Both types of authentication require an OATH compliant device that can provide the OTP.

HOTP authentication generates the OTP every time the user requests a new OTP on their device. The device tracks the number of times the user requests a new OTP, called the counter. The OTP displays for a period of time you designate in the setup, so the user may be further in the counter on their device than on their account. OpenAM will resynchronize the counter when the user finally logs in. To accommodate this, you set the number of passwords a user can generate before their device cannot be resynchronized. For example, if you set the number of HOTP Window Size to 50 and someone presses the button 30 on the user's device to generate a new OTP, the counter in OpenAM will review the OTPs until it reaches the OTP entered by the user. If someone presses the button 51 times, you will need to reset the counter to match the number on the device's counter before the user can login to OpenAM. HOTP authentication does not check earlier passwords, so if the user attempts to reset the counter on their device, they will not be able to login until you reset the counter in OpenAM to match their device.

TOTP authentication constantly generates a new OTP based on a time interval you specify. The device tracks the last two passwords generated and the current password. The Last Login Time monitors the time when a user logs in to make sure that user is not logged in several times within the present time period. Once a user log into OpenAM, they must wait for the time it takes TOTP to generate the next two passwords and display them. This prevents others from being able to access the users account using the OTP they entered. The user's account can be accessed again after the generation of the third new OTP is generated and displayed on their device. For this reason, the TOTP Time-Step Interval should not be so long as to lock users out, with a recommended time of 30 seconds.

An authentication chain can be created to generate an OTP from either HOTP or TOTP.

**ssoadm** service name: iPlanetAMAuthOATHService

Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: iplanet-am-auth-oauth-auth-level

One Time Password Length (in digits)

Set the length of the OTP between 6 and 9 digits long. The default value is 6 digits.

**ssoadm** attribute: iplanet-am-auth-oauth-password-length

Minimum Secret Key Length

The minimal number of characters required to set the Secret Key.

**ssoadm** attribute: iplanet-am-auth-oauth-min-secret-key-length

Secret Key Attribute Name

The name of the attribute where the key will be stored in the user profile.

**ssoadm** attribute: iplanet-am-auth-oauth-secret-key-attribute

OATH Algorithm to Use

Select whether to use HOTP or TOTP. You can create an authentication chain to allow for a greater variety of devices. The default value is HOTP.

**ssoadm** attribute: iplanet-am-auth-oauth-algorithm

HOTP Window Size

The number of requests that the system and the device can be off to resynchronize the password. If a user passes this number of requests before logging into the system, the password will not work. The default value is 100.

**ssoadm** attribute: `iplanet-am-auth-oauth-hotp-window-size`

Counter Attribute Name

The name of the HOTP attribute where the counter will be stored in the user profile.

**ssoadm** attribute: `iplanet-am-auth-oauth-hotp-counter-attribute`

Add Checksum Digit

Adds a checksum digit at the end of the HOTP password to verify it was entered correctly. The default value is No.

**ssoadm** attribute: `iplanet-am-auth-oauth-add-checksum`

Truncation Offset

Advanced feature that is device specific. Any value below 0 or above 15 will turn off the functionality. The default value is -1. If not required by the device, leave at the default setting.

**ssoadm** attribute: `iplanet-am-auth-oauth-truncation-offset`

TOTP Time Step Interval

Defines how long the password will appear on the user's device (in seconds). We recommend keeping this number low, for example 30 seconds, because once a user logs out, they will not be able to login again until two full time cycles have passed. The default value is 30 seconds.

**ssoadm** attribute: `iplanet-am-auth-oauth-size-of-time-step`

TOTP Time Steps

The number of requests that the system and the device can be off to resynchronize the password. If a user passes this number of requests before logging into the system, the password will not work. The default value is 2.

**ssoadm** attribute: `iplanet-am-auth-oauth-steps-in-window`

Last Login Time Attribute

The name of the attribute where both HOTP and TOTP authentication will store information on when a person last logged in.

**ssoadm** attribute: `iplanet-am-auth-oauth-last-login-time-attribute-name`

If you plan to use Yubikey for your OATH module, you will need to take some time to set it up. Go to the [YubiKey website](#) to configure your YubiKey device. If you do not have a YubiKey device, you can purchase them from this page as well.

Select the appropriate instructions, keeping in mind that the cross-platform personalization tool is recommended unless you have specific need for one of the other types. You have the greatest selection of platforms for this choice. Each device will need to be setup before use.

## 2.3.16 Hints for the OAuth 2.0/OpenID Connect Authentication Module

The OAuth 2.0 / OpenID Connect authentication module lets OpenAM authenticate clients of OAuth resource servers. References in this section are to RFC 6749, [The OAuth 2.0 Authorization Framework](#).

### Tip

OpenAM provides a wizard for configuring common OAuth 2.0 / OpenID Connect authentication providers such as Facebook, Google and Microsoft. For more information, see [Section 2.2.1, “Configuring Pre-Populated Social Authentication Providers”](#).

If the module is configured to create an account if none exists, then you must provide valid SMTP settings. As part of account creation, the OAuth 2.0 / OpenID Connect client authentication module sends the resource owner an email with an account activation code. To send email, OpenAM uses the SMTP settings from the configuration for the OAuth 2.0 / OpenID Connect authentication module.

### Note

The default settings are for Facebook.

**ssoadm** service name: sunAMAuthOAuthService

Client ID

OAuth `client_id` as described in [section 2.2 of RFC 6749](#).

**ssoadm** attribute: `iplanet-am-auth-oauth-client-id`

Client Secret

OAuth `client_secret` as described in [section 2.3 of RFC 6749](#).

**ssoadm** attribute: `iplanet-am-auth-oauth-client-secret`

Authentication Endpoint URL

URL to the end point handling OAuth authentication as described in [section 3.1 of RFC 6749](#). The default value is `https://www.facebook.com/dialog/oauth`.

**ssoadm** attribute: `iplanet-am-auth-oauth-auth-service`

Access Token Endpoint URL

URL to the end point handling access tokens as described in [section 3.2 of RFC 6749](#). The default value is `https://graph.facebook.com/oauth/access_token`.

**ssoadm** attribute: `iplanet-am-auth-oauth-token-service`

## Hints for the OAuth 2.0/OpenID Connect Authentication Module

---

### User Profile Service URL

User profile URL that returns profile information in JSON format. The default value is `https://graph.facebook.com/me`.

**ssoadm** attribute: `iplanet-am-auth-oauth-user-profile-service`

### Scope

According to [The OAuth 2.0 Authorization Framework](#), a space-separated list of user profile attributes that the client application requires. The list depends on the permissions that the resource owner, such as the end user, grants to the client application.

Some authorization servers use non-standard separators for scopes. Facebook, for example, takes a comma-separated list.

Default: `email,read_stream` (Facebook example)

**ssoadm** attribute: `iplanet-am-auth-oauth-scope`

### Proxy URL

URL to the `/oauth2c/0AuthProxy.jsp` file, part of OpenAM.

**ssoadm** attribute: `iplanet-am-auth-oauth-sso-proxy-url`

### Account Mapper

Class implementing account mapping. The default value is `org.forgerock.openam.authentication.modules.common.mapping.AttributeMapper`.

**ssoadm** attribute: `org-forgerock-auth-oauth-account-mapper`

### Account Mapper Configuration

Map of OAuth Provider user account attributes used to find the local profile of the authenticated user, with values in the form `provider-attr=local-attr`. Default values `email=mail` and `id=facebook-id`.

**ssoadm** attribute: `org-forgerock-auth-oauth-account-mapper-configuration`

### Account Provider

An account provider. Default: `org.forgerock.openam.authentication.modules.common.mapping.AccountProvider`

**ssoadm** attribute: `org-forgerock-auth-oauth-account-provider`

### Attribute Mapper

List of fully qualified class names for implementations that map attributes from the OAuth 2.0 authorization server or OpenID Connect provider to OpenAM profile attributes. Default: `org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper`

Provided implementations are:

## Hints for the OAuth 2.0/OpenID Connect Authentication Module

`org.forgerock.openam.authentication.modules.common.mapping.JsonAttributeMapper`  
`org.forgerock.openam.authentication.modules.oidc.JwtAttributeMapper` (can only be used when using the openid scope)

### Tip

You can provide string constructor parameters by appending pipe (|) separated values.

For example, the `org.forgerock.openam.authentication.modules.oidc.JwtAttributeMapper` class can take two constructor parameters: a comma-separated list of attributes and a prefix to apply to their values. Specify these as follows:

```
org.forgerock.openam.authentication.modules.oidc.  
JwtAttributeMapper|uid,employeeNumber|myPrefix-
```

**ssoadm** attribute: `org-forgerock-auth-oauth-attribute-mapper`

Attribute Mapper Configuration

Map of OAuth Provider user account attributes to local user profile attributes, with values in the form *provider-attr=local-attr*.

**ssoadm** attribute: `org-forgerock-auth-oauth-attribute-mapper-configuration`

Save attributes in the session

When enabled, add the mapped attributes to the session saved. The default mode is Enabled.

**ssoadm** attribute: `org-forgerock-auth-oauth-save-attributes-to-session-flag`

Email attribute in OAuth2 Response

Specifies the attribute identifying email address in the response from the profile service in the OAuth provider. This setting is used to send an email address with an activation code for accounts created dynamically.

**ssoadm** attribute: `org-forgerock-auth-oauth-mail-attribute`

Create account if it does not exist

When enabled, if the user profile does not exist, optionally retrieve a password and activation code from the user, and then create the profile. The default mode is Enabled.

When the OAuth 2.0 / OpenID Connect client is configured to create new accounts, the SMTP settings must also be valid. As part of account creation, the OAuth 2.0 / OpenID Connect client authentication module sends the resource owner an email with an account activation code. To send the mail,

---

Hints for the OAuth  
2.0/OpenID Connect  
Authentication Module

---

OpenAM uses the SMTP settings you provide here in the OAuth 2.0 / OpenID Connect client configuration.

**ssoadm** attribute: `org-forgerock-auth-oauth-createaccount-flag`

Map to anonymous user

When enabled, the user sets a password, receives an activation code by email. The user must correctly set both in order for the account to be created. The default mode is Enabled.

**ssoadm** attribute: `org-forgerock-auth-oauth-prompt-password-flag`

Anonymous User

Specifies an anonymous user that exists in the current realm. The default is anonymous.

**ssoadm** attribute: `org-forgerock-auth-oauth-anonymous-user`

OAuth 2.0 Provider logout service

Specifies the optional URL of the OAuth Provider.

**ssoadm** attribute: `org-forgerock-auth-oauth-logout-service-url`

Logout options

Specifies whether not to log the user out without prompting from the OAuth Provider on logout, to log the user out without prompting, or to prompt the user regarding whether to logout from the OAuth provider.

**ssoadm** attribute: `org-forgerock-auth-oauth-logout-behaviour`

Mail Server Gateway implementation class

Class to interact with the mail server. Default: `org.forgerock.openam.authentication.modules.oauth2.DefaultEmailGatewayImpl`

**ssoadm** attribute: `org-forgerock-auth-oauth-email-gwy-impl`

SMTP host

Host name of the mail server. The default is `localhost`.

**ssoadm** attribute: `org-forgerock-auth-oauth-smtp-hostname`

SMTP port

SMTP port number for the mail server. The default value is 25.

**ssoadm** attribute: `org-forgerock-auth-oauth-smtp-port`

## Hints for the OAuth 2.0/OpenID Connect Authentication Module

---

### SMTP User Name

If the mail server requires authentication to send mail, specifies the user name.

**ssoadm** attribute: `org-forgerock-auth-oauth-smtp-username`

### SMTP User Password

If the mail server requires authentication to send mail, specifies the password.

**ssoadm** attribute: `org-forgerock-auth-oauth-smtp-password`

### SMTP SSL Enabled

When enabled, connect to the mail server over SSL. OpenAM must be able to trust the SMTP server certificate.

**ssoadm** attribute: `org-forgerock-auth-oauth-smtp-ssl_enabled`

### SMTP From address

Specifies the message sender address, such as `no-reply@example.com`. The default value is `info@forgerock.com`.

**ssoadm** attribute: `org-forgerock-auth-oauth-smtp-email-from`

### Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: `iplanet-am-auth-oauth-auth-level`

## Note

Old uses of `DefaultAccountMapper` are automatically upgraded to the equivalent default implementations.

The following tables show endpoint URLs for OpenAM when [configured as an OAuth 2.0 provider](#). The default endpoints are for Facebook as the OAuth 2.0 provider.

In addition to the endpoint URLs you can set other fields, like scope and attribute mapping, depending on the provider you use.

**Table 2.1. Endpoint URLs for OpenAM**

---

| OpenAM Field               | Details   |
|----------------------------|---|
| Authorization Endpoint URL | <code>/oauth2/authorize</code> under the deployment URL. <sup>a</sup> |

---

| OpenAM Field              | Details   |
|---------------------------|---|
| Access Token Endpoint URL | Example: <code>https://openam.example.com:8443/openam/oauth2/authorize</code> .<br><code>/oauth2/access_token</code> under the deployment URL. <sup>a</sup> |
| User Profile Service URL  | Example: <code>https://openam.example.com:8443/openam/oauth2/access_token</code> .<br><code>/oauth2/tokeninfo</code> under the deployment URL.              |
|                           | Example: <code>https://openam.example.com:8443/openam/oauth2/tokeninfo</code> .   |

<sup>a</sup>This OpenAM endpoint can take additional parameters. In particular you must specify the realm if the OpenAM OAuth 2.0 provider is configured for a subrealm rather than / (Top-Level Realm).

For example, if the OAuth 2.0 provider is configured for the realm `/customers`, then the authentication endpoint URL is as follows: `https://openam.example.com:8443/openam/oauth2/authorize?realm=/customers`

The `/oauth2/authorize` endpoint can also take `module` and `service` parameters. Use either as described in [Authenticating To OpenAM](#), where `module` specifies the authentication module instance to use or `service` specifies the authentication chain to use when authenticating the resource owner.

### 2.3.17 Hints for the OpenID Connect id\_token bearer Module

The OpenID Connect id\_token bearer module lets OpenAM rely on an [OpenID Connect 1.0](#) provider's ID Token to authenticate an end user.

#### Note

This module validates an OpenID Connect ID token and matches it with a user profile. You should not use this module if you want OpenAM to act as a client in the full OpenID Connect authentication flow.

To provision OpenAM as an OpenID Connect client, you should instead configure an OAuth 2.0 / OpenID Connect module. OpenAM provides a wizard to configure an OAuth 2.0 / OpenID Connect module that will authenticate against an OpenID Connect 1.0 provider. For more information, see [Section 2.2.2, “Configuring Custom Social Authentication Providers”](#).

The OpenID Connect id\_token bearer module expects an OpenID Connect ID Token in an HTTP request header. It validates the ID Token, and if successful, looks up the OpenAM user profile corresponding to the end user for whom the

ID Token was issued. Assuming the ID Token is valid and the profile is found, the module authenticates the OpenAM user.

You configure the OpenID Connect id\_token bearer module to specify how OpenAM gets the information needed to validate the ID Token, which request header contains the ID Token, the issuer identifier for the provider who issued the ID Token, and how to map the ID Token claims to an OpenAM user profile.

## Note

The default settings are for Google's provider.

**ssoadm** service name: amAuthOpenIdConnect

Configuration type

In order to validate the ID Token from the OpenID Connect provider, the module needs either a URL to get the public keys for the provider, or the symmetric key for an ID Token signed with a HMAC-based algorithm.

By default, the configuration type is `.well-known/openid-configuration_url`. This means the module should retrieve the keys based on information in the OpenID Connect Provider Configuration Document.

You can instead configure the authentication module to validate the ID Token signature with the client secret key you provide, or to validate the ID token with the keys retrieved from the URL to the OpenID Connect provider's JSON Web Key Set.

`.well-known/openid-configuration_url` (Default)

Retrieve the provider keys based on the information provided in the OpenID Connect Provider Configuration Document.

Specify the URL to the document as the discovery URL.

`client_secret`

Use the client secret that you specify as the key to validate the ID Token signature according to the HMAC, using the client secret to decrypt the hash and then checking that the hash matches the hash of the ID Token JWT.

`jwk_url`

Retrieve the provider's JSON Web Key Set at the URL that you specify.

**ssoadm** attribute: `openam-auth-openidconnect-crypto-context-type`

Discovery url, jwk url, or client\_secret

Edit this field depending on the Configuration type you specified.

Default: `https://accounts.google.com/.well-known/openid-configuration`

**ssoadm** attribute: `openam-auth-openidconnect-crypto-context-value`

Name of header referencing the ID Token

The module looks for the ID Token in this HTTP request header.

Default: `oidc_id_token`

**ssoadm** attribute: `openam-auth-openidconnect-header-name`

Name of OpenID Connect ID Token Issuer

This corresponds to the expected issue identifier value in the `iss` of the ID Token.

Default: `accounts.google.com`

**ssoadm** attribute: `openam-auth-openidconnect-issuer-name`

Mapping of local LDAP attributes to jwt attributes

This setting maps local user profile attributes to OpenID Connect ID Token claims, allowing the module to retrieve the user profile based on the ID Token.

In OpenID Connect, an ID Token is represented as a JSON Web Token (JWT). The [ID Token](#) section of the OpenID Connect Core 1.0 specification defines a number of claims included in the ID Token for all flows. Additional claims depend on the scopes requested of the OpenID Connect provider.

For each item in the map, the key is the ID Token field name and the value is the local user profile attribute name.

Default: `mail=email, uid=sub`

**ssoadm** attribute: `openam-auth-openidconnect-local-to-jwt-attribute-mappings`

Principal Mapper class

The principal mapper matches the OpenID Connect end user with an OpenAM account. The default principal mapper uses the mapping of local attributes to ID Token attributes to find a user profile.

Default: `org.forgerock.openam.authentication.common.mapping.AttributeMapper`

**ssoadm** attribute: `openam-auth-openidconnect-principal-mapper-class`

Principal Account Provider class

The principal account provider provides the means to search for and create OpenID Connect users given a set of attributes.

Default: `org.forgerock.openam.authentication.common.mapping.AccountProvider`

**ssoadm** attribute: `openam-auth-openidconnect-account-provider-class`

## 2.3.18 Hints for the Persistent Cookie Module

The Persistent Cookie module supports the configuration of cookie lifetimes, based on requests and a maximum time. It is the preferred method for creating a persistent cookie. If you choose this method, be aware that it does not interact with the persistent cookie options associated with the Core Authentication module. Note that, by default, the persistent cookie is called `session-jwt`.

### Important

If Secure Cookie is enabled (Configuration > Servers and Sites > Servers > *URL of your server* > Security > Cookies), the Persistent Cookie Module only works over HTTPS.

**ssoadm** service name: `iPlanetAMAuthPersistentCookieService`

Before you begin, make sure a public key alias is defined in OpenAM. The Persistent Cookie module encrypts a JSON Web Token (JWT) using a public key from the OpenAM keystore. The keystore must be configured under Authentication > All Core Settings > Security. If the keystore changes and the default test key is no longer present, the public key alias must be updated to reflect the change, otherwise the module will fail. Similarly, in multi-instance deployments, the keypair must be available on all OpenAM instances.

To configure the Persistent Cookie module globally in the console, navigate to Configuration > Authentication > Persistent Cookie. In the window that appears you should see the following attributes:

#### Idle Timeout

Specify the maximum idle time between requests, in hours. If that time is exceeded, the cookie is no longer valid.

**ssoadm** attribute: `openam-auth-persistent-cookie-idle-time`

#### Max Life

Specify the maximum life of the cookie in hours.

**ssoadm** attribute: `openam-auth-persistent-cookie-max-life`

#### Enforce Client IP

When enabled, enforces that the persistent cookie can only be used from the same client IP to which the cookie was issued.

## Hints for the Persistent Cookie Module

**ssoadm** attribute: openam-auth-persistent-cookie-enforce-ip

Use secure cookie

When enabled, adds the "Secure" attribute to the persistent cookie.

**ssoadm** attribute: openam-auth-persistent-cookie-secure-cookie

Use HTTP only cookie

When enabled, adds the "HttpOnly" attribute to the persistent cookie.

**ssoadm** attribute: openam-auth-persistent-cookie-http-only-cookie

When the Persistent Cookie module enforces 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 the *Installation Guide* section, [Handling HTTP Request Headers](#).

The Persistent Cookie module belongs with a second module in an authentication chain. To see how this works, navigate to Access Control > *Realm Name* > Authentication > New. Enter a name for the chain and add modules as shown in the figure. The following example shows how a Persistent Cookie module is sufficient; if that cookie does not yet exist, authentication in this case relies on a data store module such as LDAP.

**Persistent Cookie Chain - Properties**

[Save](#) [Reset](#) [Back to Authentication](#)

| (2 Item(s))                                     |            |         |
|---|------------|---------|
| Add   | Remove     | Reorder |
| Instance  | Criteria   | Options |
| <input checked="" type="checkbox"/> Pers_Cookie | SUFFICIENT |         |
| <input type="checkbox"/> LDAP                   | REQUIRED   |         |

Scroll down in the properties page for the chain. Set the Post Authentication Processing Class to org.forgerock.openam.authentication.modules.persistentcookie.PersistentCookieAuthModule as shown in the following figure:

**Post Authentication Processing Class**

Current Values  [Remove](#)

New Value  [Add](#)

A list of post authentication processing classes for all users in this realm.

You should now be able to authenticate automatically, as long as the cookie exists for the associated domain.

## Note

Unlike the legacy Core Authentication module, the Persistent Cookie module does not support the iPSPCookie option described in [Section 2.6, "Authenticating To OpenAM"](#)

### 2.3.19 Hints for the RADIUS Authentication Module

The Remote Authentication Dial-In User Service (RADIUS) module lets OpenAM authenticate users against RADIUS servers.

**ssoadm** service name: `iPlanetAMAuthRadiusService`

Primary Radius Servers, Secondary Radius Servers

Specify the IP address or fully qualified domain name of the primary RADIUS server. The default is `127.0.0.1` (localhost loopback).

**ssoadm** attribute: `primary is iplanet-am-auth-radius-server1; secondary is iplanet-am-auth-radius-server2`

Shared Secret

Specify the shared secret for RADIUS authentication. The shared secret should be as secure as a well-chosen password.

**ssoadm** attribute: `iplanet-am-auth-radius-secret`

Port Number

Specify the RADIUS server port. Default is `1645`.

**ssoadm** attribute: `iplanet-am-auth-radius-server-port`

Timeout

Specify how many seconds to wait for the RADIUS server to respond. The default value is `3` seconds.

**ssoadm** attribute: `iplanet-am-auth-radius-timeout`

Health check interval

Used for failover. Specify how often OpenAM performs a health check on a previously unavailable RADIUS server by sending an invalid authentication request. Default: `5 minutes`

**ssoadm** attribute: `openam-auth-radius-healthcheck-interval`

Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: `iplanet-am-auth-radius-auth-level`

## 2.3.20 Hints for the SAE Authentication Module

The Secure Attribute Exchange (SAE) module lets OpenAM authenticate a user who has already authenticated with an entity that can vouch for the user to OpenAM, so that OpenAM creates a session for the user. This module is useful in virtual federation, where an existing entity instructs the local OpenAM instance to use federation protocols to transfer authentication and attribute information to a partner application.

**ssoadm** attribute: `sunAMAuthSAEService`

Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: `sunAMAuthSAEAuthLevel`

## 2.3.21 Hints for Scripted Authentication Modules

This section covers what to configure for scripted authentication modules.

A scripted authentication module runs scripts to authenticate a user. The configuration for the module can hold two scripts, one to include in the web page run on the client user-agent, another to run in OpenAM on the server side.

The client-side script is intended to retrieve data from the user-agent. This must be in a language the user-agent, such as JavaScript, even if the server-side script is written in Groovy.

The server-side script is intended to handle authentication.

Scripts are stored not as files, but instead as OpenAM configuration data. This makes it easy to update a script on one OpenAM server, and then to allow replication to copy it to other servers. You can manage the scripts through OpenAM console, where you can write them in the text boxes provided or upload them from files.

You can also upload scripts using the **ssoadm** command. To perform the command-line equivalent of the Upload button in the console, use the **ssoadm** attributes, `iplanet-am-auth-scripted-client-script-file` or `iplanet-am-auth-`

`scripted-server-script-file`. The following example shows how to create a scripted authentication module, sets up a basic configuration with scripts as strings in the configuration, and then uploads a server-side script file.

```
#  
# Create a scripted authentication module.  
#  
ssoadm \  
  create-auth-instance \  
  --adminid amadmin \  
  --password-file /tmp/pwd.txt \  
  --authtype Scripted \  
  --realm / \  
  --name ScriptedModule  
  
#  
# Add basic configuration, including minimal client and server-side scripts.  
#  
ssoadm \  
  update-auth-instance \  
  --adminid amadmin \  
  --password-file /tmp/pwd.txt \  
  --authtype Scripted \  
  --realm / \  
  --name ScriptedModule \  
  --attributevalues \  
    iplanet-am-auth-scripted-auth-level=0 \  
    iplanet-am-auth-scripted-client-script-enabled=false \  
    iplanet-am-auth-scripted-script-type='JavaScript' \  
    iplanet-am-auth-scripted-client-script='alert("Hello");' \  
    iplanet-am-auth-scripted-server-script='authState=SUCCESS;'  
  
#  
# Upload a server-side script from a script file, /tmp/server-side.js.  
#  
ssoadm \  
  update-auth-instance \  
  --adminid amadmin \  
  --password-file /tmp/pwd.txt \  
  --authtype Scripted \  
  --realm / \  
  --name ScriptedModule \  
  --attributevalues \  
    iplanet-am-auth-scripted-server-script-file=/tmp/server-side.js
```

If you have multiple separate sets of client-side and server-side scripts, then configure multiple modules, one for each set of scripts.

For details on writing authentication module scripts, see the *Developer Guide* chapter, [Scripting Authentication](#).

**ssoadm** service name: iPlanetAMAuthScriptedService

Use the following settings at the realm level when configuring an individual scripted authentication module, in OpenAM Console under Access Control > *Realm Name* > Authentication > Module Instances.

**Server-Side Script Language**

The language of the server-side script.

**ssoadm** attribute: `iplanet-am-auth-scripted-script-type`

**Client Side Script Enabled**

When selected, include the client-side script in the login page to be executed on the user-agent prior to the server-side script.

**ssoadm** attribute: `iplanet-am-auth-scripted-client-script-enabled`

**Client Side Script**

The script to include in the login page. This script is run on the user-agent prior to the server-side script.

This script must be written in a language the user-agent can interpret, such as JavaScript, even if the server-side script is written in Groovy.

**ssoadm** attribute: `iplanet-am-auth-scripted-client-script`

When uploading the script from a file, use the **ssoadm** attribute: `iplanet-am-auth-scripted-client-script-file`

**Server Side Script**

The script to run in OpenAM after the client-side script has completed.

**ssoadm** attribute: `iplanet-am-auth-scripted-server-script`

When uploading the script from a file, use the **ssoadm** attribute: `iplanet-am-auth-scripted-server-script-file`

**Authentication Level**

Sets the authentication level used to indicate the level of security associated with the scripted authentication module.

The value can range from 0 to any positive integer.

**ssoadm** attribute: `iplanet-am-auth-scripted-auth-level`

Use the following global configuration settings for all scripted authentication modules, in OpenAM Console under Configuration > Authentication > Scripted Module.

**Server Side Script Timeout**

Sets the maximum execution time to run a single server-side script, in seconds.

Default: 0 (meaning no timeout)

If your server-side scripts send external HTTP requests, consider setting an appropriate value for this timeout.

**ssoadm** attribute: `iplanet-am-auth-scripted-server-timeout`

Core thread pool size

Core size for the pool of threads dedicated to running server-side scripts.

Default: 10

This setting is not hot-swappable. Instead, you must restart OpenAM or the container in which it runs for the changes to take effect.

**ssoadm** attribute: `iplanet-am-auth-scripted-core-threads`

Maximum thread pool size

Maximum size for the pool of threads dedicated to running server-side scripts.

Default: 50

This setting is not hot-swappable. Instead, you must restart OpenAM or the container in which it runs for the changes to take effect.

**ssoadm** attribute: `iplanet-am-auth-scripted-core-threads`

### 2.3.22 Hints For the SecurID Authentication Module

The SecurID module lets OpenAM authenticate users with RSA Authentication Manager software and RSA SecurID authenticators.

In order to use the SecurID authentication module, you must first build an OpenAM war file that includes the supporting library, authapi-2005-08-12.jar.

1. Unpack the OpenAM .war file.

```
$ mkdir /tmp/openam
$ cd /tmp/openam/
$ jar -xf ~/Downloads/openam/OpenAM-12.0.0.war
```

2. Copy authapi-2005-08-12.jar into the WEB-INF/lib directory.

```
$ cp /path/to/authapi-2005-08-12.jar WEB-INF/lib/
```

3. Pack up the OpenAM .war file to deploy.

```
$ jar -cf ../openam.war *
```

4. Deploy the new .war file.

In this example the .war file to deploy is /tmp/openam.war.

**ssoadm** service name: iPlanetAMAuthSecurIDService

**ACE/Server Configuration Path**

Specify the directory where the SecurID ACE/Server sdconf.rec file is located, which by default is expected under the OpenAM configuration directory, such as \$HOME/openam/openam/auth/ace/data. The directory must exist before OpenAM can use SecurID authentication.

**ssoadm** attribute: iplanet-am-auth-securid-server-config-path

**Authentication Level**

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: iplanet-am-auth-securid-auth-level

### 2.3.23 Hints for the Windows Desktop SSO Authentication Module

The Windows Desktop SSO module uses Kerberos authentication. The user presents a Kerberos token to OpenAM through the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) protocol. The Windows Desktop SSO authentication module enables desktop single sign on such that a user who has already authenticated with a Kerberos Key Distribution Center can authenticate to OpenAM without having to provide the login information again. Users might need to set up Integrated Windows Authentication in Internet Explorer to benefit from single sign on when logged on to a Windows desktop.

**ssoadm** service name: iPlanetAMAuthWindowsDesktopSSOService

**Service Principal**

Specify the Kerberos principal for authentication in the following format.

```
HTTP/host.domain@dc-domain-name
```

Here, *host* and *domain* correspond to the host and domain names of the OpenAM instance, and *dc-domain-name* is the domain name of the Windows Kerberos domain controller server. The *dc-domain-name* can differ from the domain name for OpenAM.

You set up the account on the Windows domain controller, creating a computer account for OpenAM and associating the new account with a service provider name.

**ssoadm** attribute: `iplanet-am-auth-windowsdesktopsso-principal-name`

**Keytab File Name**

Specify the full path of the keytab file for the Service Principal. You generate the keytab file using the Windows **ktpass** utility.

**ssoadm** attribute: `iplanet-am-auth-windowsdesktopsso-keytab-file`

**Kerberos Realm**

Specify the Kerberos Key Distribution Center realm. For the Windows Kerberos service this is the domain controller server domain name.

**ssoadm** attribute: `iplanet-am-auth-windowsdesktopsso-kerberos-realm`

**Kerberos Server Name**

Specify the fully qualified domain name of the Kerberos Key Distribution Center server, such as that of the domain controller server.

**ssoadm** attribute: `iplanet-am-auth-windowsdesktopsso-kdc`

**Return Principal with Domain Name**

When enabled, OpenAM automatically returns the Kerberos principal with the domain controller's domain name during authentication.

**ssoadm** attribute: `iplanet-am-auth-windowsdesktopsso-returnRealm`

**Authentication Level**

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: `iplanet-am-auth-windowsdesktopsso-auth-level`

**Search for the user in the realm**

Validates the user against the configured data stores. If the user from the Kerberos token is not found, authentication will fail. If an authentication chain is set, the user is able to authenticate through another module. This search uses the Alias Search Attribute Name from the core realm attributes. See [Core - Realm Attributes](#) for more information.

**ssoadm** attribute: `iplanet-am-auth-windowsdesktopsso-lookupUserInRealm`

## Note

For Windows 7 and later, you will need to turn off the "Enable Integrated Windows Authentication" option in Internet Explorer. In addition, you will

need to add and activate the DisableNTMLPreAuth key to the Windows Registry. For detailed instructions, see the Microsoft KB article on when [\*You cannot post data to a non-NTLM-authenticated Web site.\*](#)

### 2.3.24 Hints for the Windows NT Authentication Module

The Windows NT module lets OpenAM authenticate against a Microsoft Windows NT server.

This module requires that you install a Samba client in a bin directory under the OpenAM configuration directory such as \$HOME/openam/openam/bin.

**ssoadm** service name: iPlanetAMAuthNTService

Authentication Domain

Specify the Windows domain name to which users belong.

**ssoadm** attribute: iplanet-am-auth-nt-domain

Authentication Host

Specify the NetBIOS name of the Windows NT host to which to authenticate users.

**ssoadm** attribute: iplanet-am-auth-nt-host

Samba Configuration File Name

Specify the full path to the Samba configuration file.

**ssoadm** attribute: iplanet-am-auth-samba-config-file-name

Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: iplanet-am-auth-nt-auth-level

### 2.3.25 Hints for the WSSAuth Authentication Module

The Web Service Security (WSSAuth) module lets OpenAM validate a user name, password combination received as an authentication token in a request from a Web Service Client to a Web Service Provider.

#### Note

The current implementation of the WSSAuth Authentication module, along with the Web Service Provider, Web Service Client and STS

Client Policy agents, were developed for versions 1.0 and 1.3 of the WS-Trust standard. ForgeRock now includes configuration options for the RESTful Security Token Service (STS), compatible with version 1.4 of the WS-Trust standard, which uses client certificate and Kerberos JASPI authentication modules.

For more information on the new implementation of STS, see [The RESTful Security Token Service](#).

**ssoadm** service name: sunAMAuthWSSAuthModuleService

User search attribute

Specify a user attribute to search for a user. Default is uid.

**ssoadm** attribute: sunWebservicesUserSearchAttribute

User realm

Specify the realm to which users belong. For the OpenAM STS, this is /.

**ssoadm** attribute: sunWebServicesUserRealm

User password attribute

Specify the password attribute or that of the password equivalent. The default is userPassword.

**ssoadm** attribute: sunWebservicesUserPasswordAttribute

Authentication Level

Sets the authentication level used to indicate the level of security associated with the module. The value can range from 0 to any positive integer.

**ssoadm** attribute: sunWebservicesAuthenticationLevel

## 2.4

## Configuring Authentication Chains

Once you have configured authentication modules, and added the modules to the list of module instances, you can configure authentication chains. Authentication chains let you handle situations where alternative modules are needed, or where a single set of credentials is not sufficient.

### Tip

OpenAM provides a wizard for configuring authentication providers, including Facebook, Google and Microsoft. The wizard creates a relevant Authentication Chain as part of the process. For more information, see [Section 2.2, “Configuring Social Authentication”](#).

#### Procedure 2.4. To Create an Authentication Chain

1. On the Access Control tab page of the OpenAM console, click the realm for which to create the authentication chain.
2. On the Authentication tab page for the realm, scroll to the bottom of the page, and click the New button in the Authentication Chaining table.
3. Give the new authentication chain a name, and add instances of the modules to use in the chain.
4. Assign appropriate criteria (optional, required, requisite, sufficient) as described above in [Section 2.1, “About Authentication in OpenAM”](#). You can also configure where OpenAM redirects the user upon successful and failed authentication, and plug in your post-authentication processing classes as necessary.
5. If you need modules in the chain to share user credentials, then set options for the module.

`iplanet-am-auth-shared-state-enabled`

Set `iplanet-am-auth-shared-state-enabled=true` to allow subsequent modules in the authentication chain to use the credentials, such as user name and password, captured by this module. (Default: `true`)

`iplanet-am-auth-store-shared-state-enabled`

Set `iplanet-am-auth-store-shared-state-enabled=true` to store the captured credentials. Shared state is cleared when the user successfully authenticates, quits the chain, or logs out. (Default: `false`)

`iplanet-am-auth-shared-state-behavior-pattern`

Set `iplanet-am-auth-shared-state-behavior-pattern=tryFirstPass` (the default) to try authenticating with the captured password. If authentication fails, then OpenAM prompts the user for the credentials again.

Set `iplanet-am-auth-shared-state-behavior-pattern=useFirstPass` to authenticate with the captured password. If authentication fails, then the module fails.

For example, consider a chain with two modules sharing credentials according to the default settings. The first module in the chain has the option `iplanet-am-auth-shared-state-enabled=true`, and criteria `REQUIRED`. The second module in the chain has options `iplanet-am-auth-shared-state-enabled=true`, `iplanet-am-auth-shared-state-behavior-pattern=tryFirstPass`, and criteria `REQUIRED`. A successful authentication sequence happens as

follows. The user enters her credentials for the first module, successfully authenticating. The first module shares the credentials with the second module, successfully authenticating the user without prompting again for her credentials, unless the credentials for the first module do not successfully authenticate here to the second module. Just be sure to separate the options with a space and not a comma.

6. Save your work.

### **Procedure 2.5. To Select the Default Chain**

Before you select the default chain for users, and especially for administrators, test the authentication chain first. For example, <http://openam.example.com:8080/openam/UI/Login?service>NewChain>. If you cannot log in, then go back and fix the authentication chain's configuration before making it the default.

1. On the Access Control tab page of the OpenAM console, click the realm for which to set the default authentication chain.
2. If necessary, on the Authentication tab page for the realm, adjust the drop-down lists for Organization Authentication Configuration and Administrator Authentication Configuration to the appropriate authentication chains.

The Organization Authentication Configuration serves when users access /openam/UI/Login.

The Administrator Authentication Configuration serves when users access /openam/console.

You can set these independently to separate administrative login from user login. For example, you can change the default user chain, but leave the default administrator chain as is to avoid locking yourself out as administrator. By default, amadmin can login at /openam/UI/Login. You can change that for your deployment.

3. Save your work.

## **2.5 Post Authentication Plugins**

Post authentication plugins include custom processing at the end of the authentication process, immediately before the subject is authenticated. Common uses of post authentication plugins include setting cookies and session variables. Post authentication plugins are often used in conjunction with policy agents. The post authentication plugin sets custom session properties, and then the policy agent injects the custom properties into the request header to the protected application.

In the OpenAM console, you add post authentication plugins to an authentication chain. Navigate to Access Control > *Realm Name* > Authentication > Authentication Chaining > *Auth Chain Name*. Scroll down to the Post Authentication Processing Class list.

### **Standard Post Authentication Plugins**

OpenAM provides some post authentication plugins as part of the standard product delivery.

Class name: `org.forgerock.openam.authentication.modules.adaptive.Adaptive`  
The adaptive authentication plugin serves to save cookies and profile attributes after successful authentication.

Add it to your authentication chains that use the adaptive authentication module configured to save cookies and profile attributes.

Class name: `org.forgerock.openam.authentication.modules.oauth2.OAuth2PostAuthnPlugin`

The OAuth 2.0 post authentication plugin builds a global logout URL used by /oauth2c/OAuthLogout.jsp after successful OAuth 2.0 client authentication. This logs the resource owner out with the OAuth 2.0 provider when logging out of OpenAM.

Before using this plugin, configure the OAuth 2.0 authentication module with the correct OAuth 2.0 Provider logout service URL, and set the Logout options to Log out or Prompt. This plugin cannot succeed unless those parameters are correctly set.

Sometimes OAuth 2.0 providers change their endpoints, including their logout URLs. When using a provider like Facebook, Google, or MSN make sure you are aware when they change their endpoint locations so that you can change your client configuration accordingly.

Class name: `org.forgerock.openam.authentication.plugins.AccountExpirePlugin`  
The account expiration post authentication plugin sets an account expiration date after successful authentication. OpenAM uses this to prevent expired accounts from being used to authenticate.

The default of 30 days can be changed using the advanced OpenAM server property, `org.forgerock.openam.authentication.accountExpire.days`.

If necessary, you can also write your own custom post authentication plugin as described in the *Developer's Guide* chapter on [Creating a Post Authentication Plugin](#).

## **2.6      Authenticating To OpenAM**

This section explains how to connect to OpenAM for user authentication by adding parameters to the login URL when testing your configuration.

The base URL to authenticate to OpenAM points to /UI/Login under the deployment URL, such as `http://openam.example.com:8080/openam/UI/Login`.<sup>2</sup> You can, however, specify parameters in the query string of the URL to request a specific authentication configuration. For example, `http://openam.example.com:8080/openam/UI/Login?module=LDAP` requests that OpenAM use the LDAP authentication module.

OpenAM accepts the following parameters in the query string. With the exception of IDToken parameters, use no more than one occurrence of each.

## Note

The way you enter parameters depends on whether you are using the XUI or the classic UI. For example, if you want to request that OpenAM end the user's current session and start a new session in the classic UI, you might enter a URL similar to: `http://openam.example.com:8080/openam/UI/Login?locale=fr`. Alternatively, for the JavaScript-based XUI, the corresponding URL would be subtly different: `http://openam.example.com:8080/openam/XUI/#login/&arg=newsession`.

**arg=newsession**

Request that OpenAM end the user's current session and start a new session.

**authlevel**

Request that OpenAM authenticate the user using a module with at least the specified authentication level that you have configured.

As this parameter determines authentication module selection, do not use it with `module`, `service`, or `user`.

**ForceAuth**

If `ForceAuth=true`, request that OpenAM force the user to authenticate even if she already has a valid session. On successful authentication, OpenAM updates the session token.

**goto**

On successful authentication, or successful logout, request that OpenAM redirect the user to the specified location. Values must be URL encoded.

**gotoOnFail**

On authentication failure, request that OpenAM redirect the user to the specified location. Values must be URL encoded.

---

<sup>2</sup>The base URL to logout is similar, for example `http://openam.example.com:8080/openam/UI/Logout`.

IDToken1, IDToken2, ..., IDTokenN

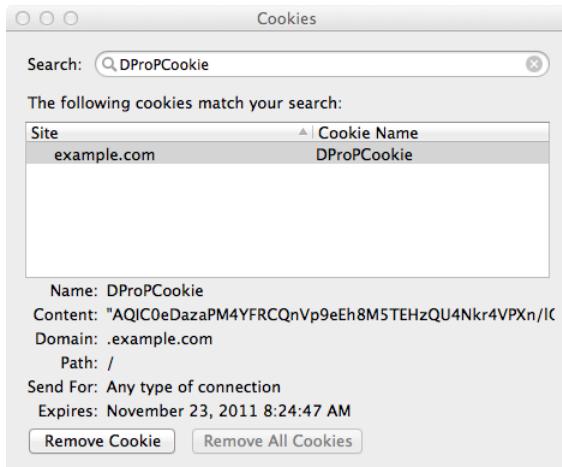
Pass the specified credentials as IDToken parameters in the URL. The IDToken credentials map to the fields in the login page for the authentication module, such as IDToken1 as user ID and IDToken2 as password for basic user name, password authentication. The order depends on the callbacks in login page for the module; IDTokenN represents the N<sup>th</sup> callback of the login page.

iPSPCookie=yes

Applicable only if you configure persistent cookies based on [Section 2.3.5.3, "Core - Persistent Cookie \(Legacy\)"](#)

Request that OpenAM return a persistent cookie that remains in the browser after the browser is closed, allowing the user to login again without being prompted for credentials. This only works if you have configured persistent cookie mode for the realm where the user logs in.

OpenAM sets an DProPCookie that persists until expiry. The following screen shot shows an example.



An alternative persistent cookie mechanism extends the lifetime of the normal iPlanetDirectoryPro using the advanced server settings, `openam.session.persist_am_cookie` or `openam.session.allow_persist_am_cookie`, and `com.iplanet.am.cookie.timeToLive`.

To set the mechanism globally for the server, browse in the OpenAM console to Configuration > Servers and Sites > *Server Name* > Advanced, and then set `openam.session.persist_am_cookie` to true and `com.iplanet.am.cookie.timeToLive` to the cookie lifetime in seconds.

To allow users to use this mechanism on a per-session basis, browse in the OpenAM console to Configuration > Servers and Sites > *Server*

*Name* > Advanced, and then set `openam.session.allow_persist_am_cookie` to true and `com.ipplanet.am.cookie.timeToLive` to the cookie lifetime in seconds. (If the OpenAM.war deployed does not include the console, set these properties in the .properties configuration file.) Also configure the session properties either globally under Configuration > Global > Session > Dynamic Attributes, or per realm under Access Control > *Realm Name* > Services > Session. Then, to request the cookie, use `openam.session.persist_am_cookie=Yes` as one of the query string parameters in the login URL.

## Note

Neither the XUI, Persistent Cookie module, distributed authentication service (DAS), nor cross-domain single sign-on (CDSSO) support the deprecated `iPSPCookie`. Similarly, DAS and CDSSO do not support `DProPCookie`.

### locale

Request that OpenAM display the user interface in the specified, supported locale. Locale can also be set in the user's profile, in the HTTP header from her browser, configured in OpenAM, and so on.

### module

Request that OpenAM use the authentication module instance as configured for the realm where the user is authenticating.

As this parameter determines authentication module selection, do not use it with `authlevel`, `service`, or `user`.

### realm

Request that OpenAM authenticate the user to the specified realm.

### service

Request that OpenAM authenticate the user with the specified authentication chain.

As this parameter determines authentication module selection, do not use it with `authlevel`, `module`, or `user`.

### user

Request that the user, specified by her OpenAM universal ID, authenticate according to the chain specified in her profile.

As this parameter determines authentication module selection, do not use it with `authlevel`, `module`, or `service`.

## 2.7 Authentication Levels & Session Upgrade

As shown in [Section 2.3, “Configuring Authentication Modules”](#), authentication modules are configured with an authentication level. This configuration sets the level of security associated with the module. Stronger forms of authentication are assigned higher authentication levels. (Or lower authentication level numbers if the deployment defines stronger authentication with lower authentication level numbers.) Upon successful authentication, a user's session includes information about the authentication level achieved.

Authorization policies can require a particular authentication level for access to sensitive resources (or at most or at least a specified authentication level). When a user who is already authenticated in the realm tries to access a sensitive resource with a valid session but that does not have the requisite authentication level, OpenAM denies access to the resource. However, OpenAM also returns *advices* with the authorization decision. The advices indicate the need for the required authentication level. The policy agent or policy enforcement point can then send the user back to OpenAM for *session upgrade*.

During session upgrade the user authenticates with a stronger authentication module. The stronger module is typically part of the same authentication chain that handled the original authentication, though not required for access to less sensitive resources. Upon successful stronger authentication, the user session is upgraded to the new authentication level and modified to include any settings related to the stronger authentication.

If unsuccessful, session upgrade leaves the user session as it was before the attempt at stronger authentication. If session upgrade failed because the login page times out, OpenAM redirects the user's browser to the success URL from the last successful authentication.

OpenAM policy agents generally handle session upgrade without additional configuration, as policy agents are built to handle OpenAM's advices. If you build your own policy enforcement point (PEP), however, take advices and session upgrade into consideration. For RESTful PEPs, see the *Developer's Guide* sections, [Requesting Policy Decisions](#) and [Authentication & Logout](#), for indications on how to handle advices and session upgrade.

## 2.8 Configuring Account Lockout

OpenAM supports two different approaches to *account lockout*, where OpenAM locks an account after repeated authentication failures. Lockout works with modules for which users can enter a password incorrectly.

- Memory lockout locks the user account, keeping track of the locked state only in memory, and then unlocking the account after a specified delay. Memory lockout is also released when OpenAM restarts.

- Persistent (physical) lockout sets the user account status to `inactive` in the user profile. For persistent lockout, OpenAM tracks failed authentication attempts by writing to the user repository.

Persistent account lockout works independently of account lockout mechanisms in the underlying directory server that serves as the user data store.

You configure account lockout by editing settings for the [core authentication module](#). Access the settings in OpenAM console under Access Control > *Realm Name* > Authentication > All Core Settings..., and then scroll down to the Account Lockout section. The inline help explains the settings in detail.

- Enable lockout by checking Login Failure Lockout Mode, setting the number of attempts, and setting the lockout interval and duration.

You can also opt to warn users after several consecutive failures, or to multiply the lockout duration on each successive lockout.

- You can set up email notification upon lockout to an administrator if OpenAM is configured to send mail. (Configuration > Servers and Sites > Default Server Settings > General > Mail Server.)
- For persistent lockout, OpenAM sets the value of the user's `inetuserstatus` profile attribute to `inactive`. You can also specify another attribute to update on lockout. You can further set a non-default attribute on which to store the number of failed authentication attempts. When you do store the number of failed attempts in the data store, other OpenAM servers accessing the user data store can also see the number.

If you need to unlock a user's account, find the user under Access Control > *Realm Name* > Subjects > User, set the user's User Status to Active, and click Save.

## 2.9 Configuring Session Quotas

OpenAM lets you limit the number of active sessions for a user by setting session quotas. You also configure session quota exhaustion actions so that when a user goes beyond the session quota, OpenAM takes the appropriate action.

### Important

To enforce session quotas across multiple servers in a site, configure session failover as described in the *Installation Guide* chapter, [Setting Up OpenAM Session Failover](#).

### **Procedure 2.6. To Configure Session Quotas & Exhaustion Actions**

The session quota applies to all sessions opened for the same user (as represented by the user's universal identifier).

1. Log in to OpenAM Console as administrator, and then browse to Configuration > Global > Session.
2. Set Enable Quota Constraints to ON.
3. Set Resulting behavior if session quota exhausted.

The following settings are available by default.

**DENY\_ACCESS**

Deny access, preventing the user from creating an additional session.

**DESTROY\_NEXT\_EXPIRING**

Remove the next session to expire, and create a new session for the user. The next session to expire is the session with the minimum time left until expiration.

This is the default setting.

**DESTROY\_OLDEST\_SESSION**

Remove the oldest session, and create a new session for the user.

**DESTROY\_OLD\_SESSIONS**

Remove all existing sessions, and create a new session for the user.

If none of these session quota exhaustion actions fit your deployment, you can implement a custom session quota exhaustion action. See the *Developer's Guide* chapter on [Customizing Session Quota Exhaustion Actions](#) for an example.

4. Set Active User Sessions to the session quota.

The default is 5 sessions.

5. Save your work.
6. If you have multiple servers but session failover is not configured, configure multi-server mode as described below.
  - If you have only a single OpenAM server, skip this step. OpenAM enforces the session quota you set for the server.

- If you have multiple servers with session failover configured, then also skip this step. In this case OpenAM uses the session store to enforce session quotas globally across your deployment. In other words when the Set Active User Sessions is 5, a user can have a maximum of 5 active sessions.
- If you have multiple OpenAM servers but session failover is not configured, configure multi-server mode for session quotas. Browse to Configuration > Servers and Sites > Default Server Settings or Configuration > Servers and Sites > *Server Name*, and then use the Advanced tab page to set the following advanced server property.

```
openam.session.useLocalSessionsInMultiServerMode = true
```

When you set this property to `true` for your OpenAM servers, users can potentially reach the session quota for each individual server before all session quotas are exhausted. In other words if you have 4 OpenAM servers and Set Active User Sessions is 5, then the user can have a maximum of 20 ( $5 * 4$ ) sessions.

## 2.10 Configuring Valid goto URL Resources

By default, OpenAM redirects the user to the URL specified in the `goto` and `gotoOnFail` query string parameters supplied to the authentication interface in the login URL. You can increase security against possible phishing attacks through open redirect by specifying a list of valid URL resources using the Valid goto URL Resource service.

OpenAM only redirects a user if the `goto` and `gotoOnFail` URL matches any of the resources specified in this setting. If no setting is present, it is assumed that the `goto` or `gotoOnFail` URL is valid.

The URL whitelisting and pattern matching follow the wildcard rules as specified in [To Configure a Policy for a Web Site](#). There is one main difference: when creating Applications, the URL pattern matching for `ApplicationTypes` is different from the way Applications are matched when creating policies.

- **Creating an Application.** When creating an application, the default pattern is the asterisk ("`*`") wildcard. There is one exception: the default application, `iPlanetAMWebAgentService`, uses a different pattern matching rule: `*:///*:/*/*` and `*:///*:/*/?*` as opposed to just `*`. This rule is specified due to the application resource expecting a URI.
- **Creating a Policy.** When creating a policy, any rule with a trailing asterisk ("`*`") matches anything, including a question mark ("`?`").

Here are some general examples of URL pattern matching:

- If no port is specified, `http://www.example.com` canonicalizes to `http://www.example.com:80` and `https://www.example.com` canonicalizes to `http://www.example.com:443`.

- A wildcard before "://" only matches up to "://"

For example, `http*://*.com/*` matches `http://www.example.com/hello/world` and `https://www.example.com/hello`.

- A wildcard between "://" and ":" matches up to ":"

For example, `http://*:85` matches `http://www.example.com:85`.

- A wildcard between ":" and "/" only matches up to the first "/"

For example, `http://www.*:*/` matches `http://www.example.com:80`. In another example, `http://www.example.com:*` matches `http://www.example.com:[any port]` and `http://www.example.com:[any port]/` but nothing more.

- A wildcard after "/" matches anything depending on whether it is single-level or a wildcard appropriately.

For example, `https://www.example.com/*` matches `https://www.example.com:443/foo/bar/baz/me`

- If you do not use any wildcards, OpenAM exactly matches the string, so `http://www.example.com` only matches `http://www.example.com` but NOT `http://www.example.com/` (trailing slash).

If you put the wildcard after the path, OpenAM expects a path (even if it is blank), so `http://www.example.com/*` matches `http://www.example.com/` and `http://www.example.com/foo/bar/baz.html` but NOT `http://www.example.com`.

- `http://www.example.com:*/` matches `http://www.example.com/`, which also canonicalizes to `http://www.example.com:80/`.
- `https://www.example.com:*/` matches `https://www.example.com/`, which also canonicalizes to `https://www.example.com:443/`.



---

## **Chapter 3**

# Defining Authorization Policies

*Authorization* is determining whether to grant or to deny a user access to a resource. *Policies* define how to determine whether to grant or deny access. This chapter describes how to configure authorization policies in OpenAM.

## **3.1 About Authorization in OpenAM**

Applications rely on OpenAM for access management, which breaks down into authentication, or determining who is trying to access a resource, and authorization, or determining whether to grant or deny access. This is because whether access is granted generally depends on what the policies about access are, who is trying to gain access, and perhaps some other conditions, such as whether the access itself needs to happen over a secure channel or what time of day it is.

To return to the international airport example from the discussion on authentication the policy might be that passengers with valid passports and visas presenting valid plane tickets are allowed through to the gate where the plane is waiting to take off, but only under the condition that the plane is going to leave soon. (You cannot expect to get to the gate today with a scheduled departure for three months from now.)

### **3.1.1 OpenAM Policies and Applications**

To allow OpenAM to determine whether to grant access, you define authorization *policies*. A policy defines the following.

*resources*

The resource definitions constrain which resources, such as web pages or access to the boarding area, the policy applies to.

*actions*

The actions are verbs that describe what the policy allows users to do to the resources, such as read a web page, submit a web form, or access the boarding area.

*subject conditions*

The subject conditions constrain who the policy applies to, such as all authenticated users, only administrators, or only passengers with valid tickets for planes leaving soon.

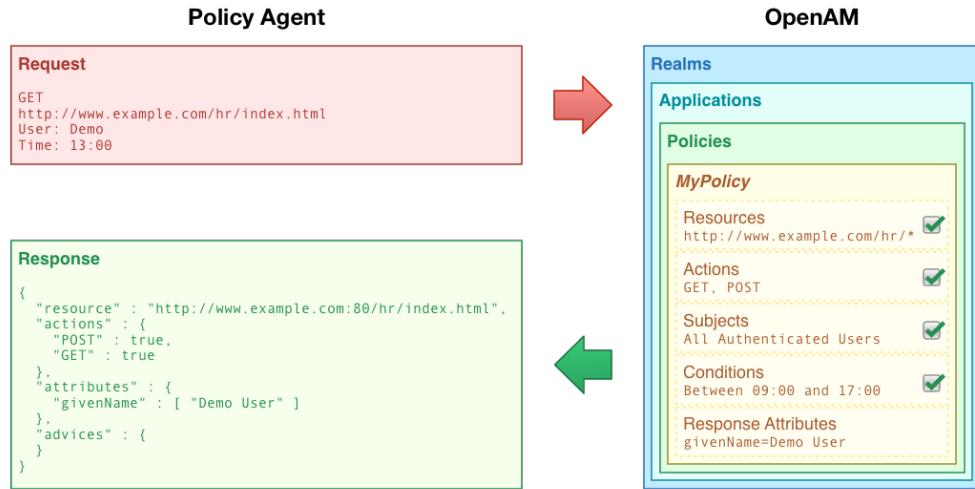
*environment conditions*

The environment conditions set the circumstances under which the policy applies, such as only during work hours, only when accessing from a specific IP address, or only when the flight is scheduled to leave within the next four hours.

*response attributes*

The response attributes define information that OpenAM attaches to a response following a policy decision, such as a name, email address, or frequent flyer status.

When queried about whether to let a user through to a protected resource, OpenAM decides to authorize access or not based on applicable policies as described below in [Section 3.1.2, “OpenAM Policy Decisions”](#). OpenAM communicates its decision to the application using OpenAM for access management. In the common case, this is a policy agent installed on the server where the application runs. The agent then enforces the authorization decision from OpenAM.



To act as a template for the policies, OpenAM uses *applications*. An application can constrain the resources that its policies apply to. For example, an application for Example.com's HR service might constrain all policies to apply to resources under `http://example.com/hr*` and `http://example.com/hr?*`. The list of applications is the first thing that you see when editing policies through OpenAM console. You create or select an application, and then configure the policies for that application.

Applications have associated types. The default application type that represents web resources is `iPlanetAMWebAgentService`, which defines resources as URL patterns and actions as HTTP methods. OpenAM policy agents use a default application type based on this type, which is called `iPlanetAMWebAgentService`. This is the application type for policies that you edit through OpenAM console. OpenAM supports other application types as well that you can manage over the policy REST APIs.

For applications that are not of type `iPlanetAMWebAgentService`, such as CREST applications, you must manage these applications over REST. OpenAM policy agents work with applications of type `iPlanetAMWebAgentService`, so your policy evaluation points (PEPs) should use the REST endpoints for policy decision requests.

## Tip

When you configure a policy agent, if the application for its policies is not named `iPlanetAMWebAgentService`, then you must edit the policy agent configuration, setting the application name to match your application.

The application you specify must exist in the evaluation realm that you specify for the policy agent.

When policies for an application are administered in multiple realms, you set up a *referral* between applications. When finding all policies that apply for a given request, OpenAM follows the referrals to locate the application's policies in other realms. This is useful, for example, when the same policy agents protect multiple applications. The OpenAM policy editor does allow you to manage referrals as well, though the referral editor is not enabled by default.

### 3.1.2 OpenAM Policy Decisions

OpenAM relies on policies to reach authorization decisions, such as whether to grant or to deny access to a resource. OpenAM acts as the *policy decision point* (PDP), whereas OpenAM policy agents act as *policy enforcement points* (PEP). In other words, a policy agent or other PEP takes responsibility only for enforcing a policy decision rendered by OpenAM. When you configured applications and their policies in OpenAM, you used OpenAM as a *policy administration point* (PAP).

Concretely speaking, when a PEP requests a policy decision from OpenAM it specifies the target resource(s), the application (default: iPlanetAMWebAgentService), and information about the subject and the environment. OpenAM as the PDP retrieves policies within the specified application that apply to the target resource(s). OpenAM then evaluates those policies to make a decision based on the conditions matching those of the subject and environment. When multiple policies apply for a particular resource, the default logic for combining decisions is that the first evaluation resulting in a decision to deny access takes precedence over all other evaluations. OpenAM only allows access if all applicable policies evaluate to a decision to allow access.

OpenAM communicates the policy decision to the PEP. The concrete decision, applying policy for a subject under the specified conditions, is called an *entitlement*.

The entitlement indicates the resource(s) it applies to, the actions permitted and denied for each resource, and optionally response attributes and *advice*.

When OpenAM denies a request due to a failed condition, OpenAM can send advice to the PEP, and the PEP can then take remedial action. For instance, suppose a user comes to a web site having authenticated with an email address and password, which is configured as authentication level 0. Had the user authenticated using a one-time password, the user would have had authentication level 1 in their session. Yet, because they have authentication level 0, they currently cannot access the desired page, as the policy governing access requires authentication level 1. OpenAM sends advice, prompting the PEP to have the user re-authenticate using a one-time password module, gaining authentication level 1, and thus having OpenAM grant access to the protected page.

### 3.1.3 Example Authorization

Consider the case where OpenAM protects a user profile web page. An OpenAM policy agent installed in the web server intercepts client requests to enforce policy. The policy says that only authenticated users can access the page to view and to update their profiles.

When a user browses to the profile page, the OpenAM policy agent intercepts the request. The policy agent notices that the request is to access a protected resource, but the request is coming from a user who has not yet logged in and consequently has no authorization to visit the page. The policy agent therefore redirects the user's browser to OpenAM to authenticate.

OpenAM receives the redirected user, serving a login page that collects the user's email and password. With the email and password credentials, OpenAM authenticates the user, and creates a session for the user. OpenAM then redirects the user to the policy agent, which gets the policy decision from OpenAM for the page to access, and grants access to the page.

OpenAM and the policy agent use cookies set in the user's browser to share an opaque reference to the session with OpenAM. While the user has a valid session with OpenAM, the user can go away to another page in the browser, come back to the profile page, and gain access without having to enter their email and password again.

Notice how OpenAM and the policy agent handle the access in the example. The web site developer can offer a profile page, but the web site developer never has to manage login, or handle who can access a page. As OpenAM administrator, you can change authentication and authorization independently of updates to the web site. You might need to agree with web site developers on how OpenAM identifies users so web developers can identify users by their own names when they log in. By using OpenAM and policy agents for authentication and authorization, your organization no longer needs to update web applications when you want to add external access to your Intranet for roaming users, open some of your sites to partners, only let managers access certain pages of your HR web site, or allow users already logged in to their desktops to visit protected sites without having to type their credentials again.

## 3.2 How OpenAM Reaches Policy Decisions

OpenAM has to match policies to resources to take policy decisions. For a policy to match, the resource has to match one of the resource patterns defined in the policy. The user making the request has to match a subject. Furthermore, at least one condition for each condition type has to be satisfied.

If more than one policy matches, OpenAM has to reconcile differences. When multiple policies match, the order in which OpenAM uses them to make a

policy decision is not deterministic. However, a deny decision overrides an allow decision, and so by default once OpenAM reaches a deny decision it stops checking further policies. (If you want OpenAM to continue checking despite the deny, see Configuration > Global > Policy Configuration > Continue Evaluation on Deny Decision.)

### 3.3 Configuring Applications, Policies, and Referrals

You can configure applications, policies and referrals by using the policy editor in the OpenAM console, by using the REST interface, or by using the **ssoadm** command.

This section explains how to use the OpenAM console to configure applications, policies and referrals to protect a web site or web application.

To configure applications and policies to protect other types of resource, you must use the REST API or the **ssoadm** command. For more information on the REST API, see the *Developer's Guide* chapters [Defining Applications](#) and [Managing Policies](#). For more information on the **ssoadm** command, see the *OpenAM Reference Guide* chapter [ssoadm](#).

#### 3.3.1 Configuring Applications by Using the OpenAM Console

This section describes the process of creating applications, which are used as templates for policies protecting a web site or web application, using the OpenAM console.

##### Procedure 3.1. To Configure an Application by Using the Policy Editor

1. In the OpenAM console, select Access Control > *Realm Name* > Policies.
  - a. To create a new application, click Add New Application.
  - b. To modify an existing application, in the row containing the application, click the Edit Application icon.
  - c. To discard any changes and return to the previous page, click Cancel.
2. Provide a name for the application, and optionally a description, and then click Next.

Do not use special characters within policy, application or referral names (for example, "my+referral") using the Policy Editor or REST endpoints as OpenAM returns a 400 Bad Request error. The special characters are: double quotes ("), plus sign (+), comma (,), less than (<), equals (=), greater than (>), backslash (\), and null (\u0000).

3. Define resource patterns that policies within this application use as the basis for their rules, by following the steps below:

- a. In the Available patterns section, click the row containing an asterisk (\*) to move the pattern to the Create your resources section.
- b. Optionally, in the Create your resources section, replace the asterisk with a pattern that the policies in the application use as a template for specifying resources.

For information on specifying patterns that the policies use for matching resources, see [Specifying Resource Patterns with Wildcards](#).

- c. Click the Add icon to move the pattern into the Resources section.

## Tip

To remove a resource pattern, click the Delete icon.

Repeat these steps to add all the resource patterns your policies may require, and then click Next.

4. Review your configuration.

## Tip

If the configuration is not visible, click the maximized link to switch to maximized view.

To make changes to the configuration, either click the step in the list, or click the item to amend in the Review Configuration and Finish section to jump to the relevant step, and make your changes.

When the configuration is complete, click Finish.

To make use of the new application and any policies it contains, you must perform one of the following procedures:

- Configure a policy agent to use the application for policy decisions.

For details see the procedure [To Specify the Realm and Application for Policy Decisions](#).

- Create a referral to the realm the application is in.

For details see the procedure [To Enable Referrals in the Policy Editor](#).

### 3.3.2 Configuring Policies by Using the OpenAM Console

This section describes the process of configuring policies to protect a web site or web application by using the OpenAM console.

#### Procedure 3.2. To Configure a Policy by Using the OpenAM Console

1. In the OpenAM console, select Access Control > *Realm Name* > Policies, and then click the name of the application to configure a policy in.
  - a. To create a new policy, click Add New Policy.
  - b. To modify an existing policy, click the name of the policy.
  - c. To discard any changes and return to the previous page, click Cancel.
2. Provide a name for the policy, and optionally a description, and then click Next.

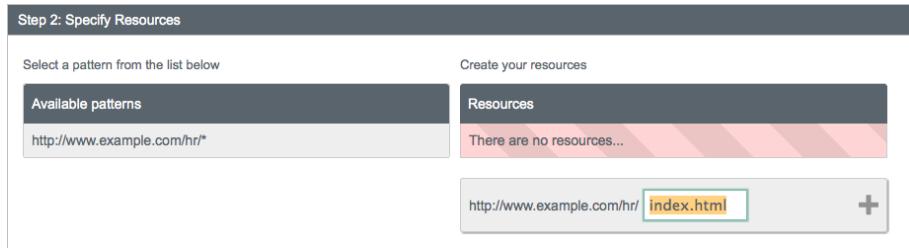
Do not use special characters within policy, application or referral names (for example, "my+referral") using the Policy Editor or REST endpoints as OpenAM returns a 400 Bad Request error. The special characters are: double quotes ("), plus sign (+), comma (,), less than (<), equals (=), greater than (>), backslash (\), and null (\u0000).

3. To define resources that the policy applies to, follow the steps below:
  - a. In the Available patterns section, click the row containing the pattern to use as the template for a resource pattern to move it to the Create your resources section.



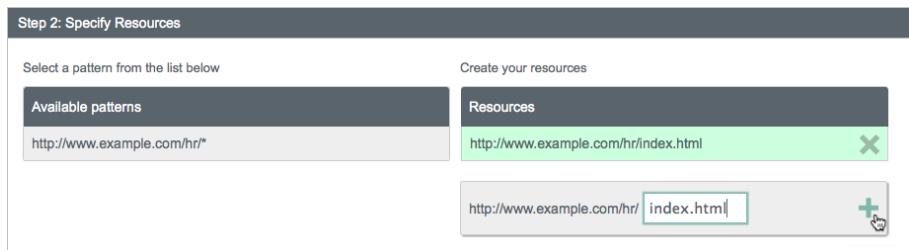
- b. Optionally, in the Create your resources section, replace the asterisks with values to define the resources that the policy applies to.

## Configuring Policies by Using the OpenAM Console



For information on specifying patterns for matching resources, see [Specifying Resource Patterns with Wildcards](#).

- c. Click the Add icon to move the resource into the Resources section.



### Tip

To remove a resource pattern, click the Delete icon.

Repeat these steps to add all the resources your policy applies to, and then click Next.

4. Select the HTTP 1.1 methods that the policy applies to, and whether to Allow or Deny those methods on the resources specified earlier, and then click Next.

For more information on HTTP 1.1 methods, see [Method Definitions](#) and [PATCH Method for HTTP](#).

5. Define conditions in the OpenAM console by combining logical operators with blocks of configured parameters to create a rule set that the policy uses to filter requests for resources. You can nest logical operators at multiple levels to create complex rule sets.

To define the subjects that the policy applies to, complete the following steps:

- a. To add a subject condition, click the Subject Condition button, choose the type from the drop-down menu, specify any required Subject Values, and then drag the block into a drop-point in a logical block above.

## Tip

Valid drop-points in which to drop a block are displayed with a striped horizontal bar.



The available Subject Condition Types are:

### Authenticated Users

Any user that has successfully authenticated with OpenAM.

### Users & Groups

A user or group as defined in the Subjects tab of the realm the policy is created in.

Select one or more users or groups from the Subject Values drop-down menu, which displays the subjects available within the realm.

To remove an entry, click the value, and then press **Delete** (Windows/GNU/Linux) or **Backspace** (Mac OS X).

### Jwt Claim

Validate a claim within a JSON Web Token (JWT).

Type the name of the claim to validate in Claim Name, for example sub, and the required value in Claim Value, for example UserA.

## Note

This condition type only supports string equality comparisons, and is case-sensitive.

### Never Match

Never match any subject, with the result not that access is denied, but instead that the policy itself does not match and therefore cannot be evaluated in order to allow access.

If you do not set a subject condition, "Never Match" is the default. In other words, you must set a subject condition for the policy to apply.

To match regardless of the subject, configure a subject condition that is NOT "Never Match".

- b. To add a logical operator, click the Logical button, choose between AND, NOT, and OR from the drop-down menu, and then drag the block into a valid drop point in the rule set above.
  - c. Continue combining logical operators and subject conditions, and then click Next.
6. To configure environment conditions in the policy, complete the following steps:
  - a. To add an environment condition, click the Environment Condition button, choose the type from the drop-down menu, specify any required parameters, and then drag the block into a drop-point in a logical block above.

## Tip

Valid drop-points in which to drop a block are displayed with a striped horizontal bar.



The available environment condition types are:

### Active Session Time

Make the policy depend on how long the user's session has been active, as specified in Max Session Time. To terminate the session if it has been active for longer than the specified time, set Terminate Sessions to True. The user will need to re-authenticate.

### Authentication by Module Chain

Make the policy depend on the service that was used to authenticate the user.

### Authentication by Module Instance

Make the policy depend on the authentication module used to authenticate, specified in Authentication Scheme. Specify a timeout for application authentication in Application Idle Timeout Scheme and the name of the application in Application Name.

### Authentication Level (greater than or equal to)

Make the policy depend on the minimum acceptable authentication level specified in Authentication Level.

**Authentication to a Realm**

Make the policy depend on the realm to which the user authenticated.

**Current Session Properties**

Make the policy depend on properties set in the user's session.

Type the value to search for in Properties, and set Ignore Value Case to True to make the search case-insensitive.

**Identity Membership**

Make the policy apply if the UUID of the invocator is a member of at least one of the AMIdentity objects specified in AM Identity Name.

Often used to filter requests on the identity of a Web Service Client (WSC).

**IPv4 Address/DNS Name**

Make the policy depend on the IP version 4 address that the request originated from.

**Note**

The IP address is taken from the `requestIp` value of policy decision requests. If this is not provided, the IP address stored in the SSO token is used instead.

Specify a range of addresses to test against by entering four sets of up to three digits, separated by full stops (.) in both Start IP and End IP.

**Note**

If only one of these values is provided, it is used as a single IP address to match.

Optionally, specify a DNS name in DNS Name to filter requests to that domain.

**IPv6 Address/DNS Name**

Make the policy depend on the IP version 6 address that the request originated from.

## Note

The IP address is taken from the `requestIp` value of policy decision requests. If this is not provided, the IP address stored in the SSO token is used instead.

Specify a range of addresses to test against by entering eight sets of four hexadecimal characters, separated by a colon (:) in both Start IP and End IP.

## Note

If only one of these values is provided, it is used as a single IP address to match.

Optionally, specify a DNS name in DNS Name to filter requests to those coming from the specified domain.

## Tip

Use an asterisk (\*) in the DNS name to match multiple subdomains. For example `*.example.com` applies to requests coming from `www.example.com`, `secure.example.com`, or any other subdomain of `example.com`.

### LDAP Filter Condition

Make the policy depend on whether the user's entry can be found using the LDAP search filter you specify in the directory configured for the policy service, which by default is the identity repository. See Configuration > Global > Policy Configuration > Realm Attributes > Primary LDAP Server.

Alternatively you can set this for the realm under Access Control > *Realm Name* > Services > Policy Configuration.

### OAuth2 Scope

Make the policy depend on whether an authorization request includes all of the specified OAuth 2.0 scopes.

Scope names must follow OAuth 2.0 scope syntax described in RFC 6749, [Access Token Scope](#). As described in that section, separate multiple scope strings with spaces, such as `openid profile`.

The scope strings match regardless of order in which they occur, so `openid profile` is equivalent to `profile openid`.

The condition is also met when additional scope strings are provided beyond those required to match the specified list. For example, if the condition specifies `openid profile`, then `openid profile email` also matches.

#### Resource/Environment/IP Address

Make the policy apply to a complex condition such as whether the user is making a request from the localhost and has also authenticated with the LDAP authentication module.

Entries must take the form of an `IF...ELSE` statement. The `IF` statement can specify either IP to match the user's IP address, or `dnsName` to match their DNS name.

If the `IF` statement is true, the `THEN` statement must also be true for the condition to be fulfilled. If not, relevant advice is returned in the policy evaluation request.

The available parameters for the `THEN` statement are as follows:

`module`

The module that was used to authenticate the user, for example `DataStore`.

`service`

The service that was used to authenticate the user.

`authlevel`

The minimum required authentication level.

`role`

The role of the authenticated user.

`user`

The name of the authenticated user.

`redirectURL`

The URL the user was redirected from.

`realm`

The realm that was used to authenticate the user.

The IP address can be IPv4, IPv6, or a hybrid of the two.

Example: `IF IP=[127.0.0.1] THEN role=admins.`

## Configuring Policies by Using the OpenAM Console

### Time (day, date, time, and timezone)

Make the policy depend on when the policy is evaluated.

The values for day, date and time must be set in pairs, a start and an end. Incomplete pairings are highlighted in red.

- b. To add a logical operator, click the Logical button, choose between AND, NOT, and OR from the drop-down menu, and then drag the block into a valid drop point in the rule set above.
  - c. Continue combining logical operators and environment conditions, and when finished, click Next.
7. Add policy response attributes, retrieved from the user entry in the identity repository, into the headers of the request at policy decision time. The policy agent for the protected resources/applications or the protected resources/applications themselves retrieve the policy response attributes to customize or personalize the application. Policy response attributes come in two formats: subject attributes and static attributes.
- a. To add subject attributes, click an empty space in the Subject attributes field, and select the subject attribute to add to the response.

To remove an entry, click the value, and then press **Delete** (Windows/GNU/Linux) or **Backspace** (Mac OS X)
  - b. To add a static attribute, type in a Key and a Value in the fields, and then click the Add icon.

To remove an entry, click the Delete icon in the row containing the Key: Value pair.
  - c. Continue adding subject and static attributes, and when finished, click Next.
8. Review your configuration.

## Tip

If the configuration is not visible, click the maximized link to switch to maximized view.

To make changes to the configuration, either click the relevant step, or click the item to jump to the relevant step and make amendments to the configuration.

When the configuration is completed, click Finish.

### 3.3.3 Configuring Referrals by Using the OpenAM Console

This section describes the process of creating referrals within an application by using the OpenAM Console.

By default, referrals are not displayed in the policy editor within the OpenAM console.

#### Procedure 3.3. To Enable Referrals in the Policy Editor

- In the OpenAM console, select Configuration > Global > Policy Configuration, set Activate Referrals to Enabled, and then click Save.

Referrals enable applications and policies to exist in different realms, and redirect incoming authorization requests to those realms.

#### Procedure 3.4. To Configure Referrals by Using the Policy Editor

1. In the OpenAM console, select Access Control > *Realm Name* > Policies, and then click the name of the application to create a referral in.
  - a. To create a new referral, click Add New Referral.
  - b. To modify an existing referral, click the name of the referral.
  - c. To discard any changes and return to the previous page, click Cancel.
2. Provide a name for the referral, and then click Next.

Do not use special characters within policy, application or referral names (for example, "my+referral") using the Policy Editor or REST endpoints as

## Configuring Referrals by Using the OpenAM Console

OpenAM returns a 400 Bad Request error. The special characters are: double quotes ("), plus sign (+), comma (,), less than (<), equals (=), greater than (>), backslash (\), and null (\u0000).

- To define resources that the referral applies to, follow the steps below:

- In the Available patterns section, click the row containing the pattern to use as the template for a resource pattern to move it to the Create your resources section.

The screenshot shows the 'Step 2: Specify Resources' dialog. On the left, under 'Available patterns', there is a single row: 'http://www.example.com/hr/\*'. On the right, under 'Resources', there is a message: 'There are no resources...'. A green arrow points from the 'Available patterns' section to the 'Resources' section, indicating the movement of the selected pattern.

- Optionally, in the Create your resources section, replace the asterisks with values to define the resources that the referral applies to.

The screenshot shows the 'Step 2: Specify Resources' dialog. Under 'Available patterns', the row 'http://www.example.com/hr/\*' is still present. In the 'Resources' section, a new resource 'http://www.example.com/hr/index.html' has been added, indicated by a green border around the input field and a green '+' icon. The message 'There are no resources...' is still visible.

For information on specifying patterns for matching resources, see [Specifying Resource Patterns with Wildcards](#).

- Click the Add icon to move the resource into the Resources section.

The screenshot shows the 'Step 2: Specify Resources' dialog. The resource 'http://www.example.com/hr/index.html' is now listed in the 'Resources' section with a green checkmark icon next to it, indicating it has been successfully moved. The 'Available patterns' section still contains the row 'http://www.example.com/hr/\*'. The green '+' icon is still visible at the bottom right of the 'Resources' section.

### Tip

To remove a resource, click the Delete icon.

Repeat these steps to add all the resources your referral applies to, and then click Next.

4. In the Available realms drop-down, select a realm to which the referral redirects requests, and then click the Add icon. You can add multiple target realms.

### Note

A referral can only redirect requests to a child, or sibling of the realm it is created in. The Available realms drop-down only displays valid realms.

To remove an entry, click the Delete icon in the row containing the realm to remove.

5. Review your configuration.

### Tip

If the configuration is not visible, click the maximized link to switch to maximized view.

To make changes to the configuration, either click the relevant step, or click the item to jump to the relevant step and make amendments to the configuration.

When the configuration is completed, click Finish.

#### 3.3.4 Specifying Resource Patterns with Wildcards

Resource patterns can specify an individual URL or resource name to protect. Alternatively, a resource pattern can match URLs or resource names by using wildcards.

- The wildcards you can use are \* and -\*.

These wildcards can be used throughout resource patterns to match URLs or resource names. For a resource pattern used to match URLs, wildcards can be employed to match the scheme, host, port, path, and query string of the resource.

- The wildcard \* matches multiple levels in a path.

For example, `http://www.example.com/*` matches `http://www.example.com/`, `http://www.example.com/index.html`, and also `http://www.example.com/company/images/logo.png`.

- The wildcard `-*` matches a single level in a path.

For example, `http://www.example.com/-*` matches `http://www.example.com/index.html` but does not match `http://www.example.com/company/images/logo.png`.

- Wildcards do not match `?`. You must explicitly add patterns to match URLs with query strings.

When used at the end of a pattern after a `?` character, `*` matches one or more characters, not zero or more characters.

For example, `http://www.example.com/*?*` matches `http://www.example.com/users?_action=create`, but not `http://www.example.com/users?.`

For example, to match everything under `http://www.example.com/` specify three patterns, one for `http://www.example.com/*`, one for `http://www.example.com/*?`, and one for `http://www.example.com/*?*`.

When defining patterns to match URLs with query strings, OpenAM sorts the query string field-value pairs alphabetically by field name when normalizing URLs before checking whether a policy matches. Therefore the query string `?subject=SPBnfm+t5PlP+ISyQhVlplE22A8=&action=get` is equivalent to the query string `?action=get&subject=SPBnfm+t5PlP+ISyQhVlplE22A8=`.

- Duplicate slashes (/) are not considered part of the resource name to match. A trailing slash is considered by OpenAM as part of the resource name.

For example, `http://www.example.com//path/`, and `http://www.example.com/path//` are treated in the same way.

`http://www.example.com/path`, and `http://www.example.com/path/` are considered two distinct resources.

- Wildcards can be used to match protocols, host names, and port numbers.

For example, `*://*:/*` matches `http://www.example.com:80/index.html`, `https://www.example.com:443/index.html`, and `http://www.example.net:8080/index.html`.

When a port number is not explicitly specified, then the default port number is implied. Therefore `http://www.example.com/*` is the same as `http://www.example.com:80/*`, and `https://www.example.com/*` is the same as `https://www.example.com:443/*`.

- Wildcards cannot be escaped.
- Do not mix \* and -\* in the same pattern.
- By default, comparisons are not case sensitive. The delimiter, wildcards and case sensitivity are configurable. To see examples of other configurations, browse in the OpenAM Console to Configuration > Global > Policy Configuration > Resource Comparator.

### 3.3.5 Importing and Exporting Policies

You can import and export policies to and from files.

You can use these files to backup policies, transfer policies between OpenAM instances, or store policy configuration in a version control system such as Git or Subversion.

The default, preferred format for importing and exporting OpenAM policies is *eXtensible Access Control Markup Language (XACML) Version 3.0*.

#### Note

OpenAM can only import XACML 3.0 files that were either created by an OpenAM instance, or that have had minor manual modifications, due to the reuse of some XACML 3.0 parameters for non-standard information.

You can import and export policies by using the policy editor in the OpenAM console, using the REST API, or with the **ssoadm** command.

- [Procedure 3.5, “To Export Policies in XACML Format \(OpenAM Console\)”](#)
- [Procedure 3.6, “To Import Policies in XACML Format \(OpenAM Console\)”](#)
- [Procedure 3.7, “To Export Policies in XACML Format \(Command Line\)”](#)
- [Procedure 3.8, “To Import Policies in XACML Format \(Command Line\)”](#)
- [Procedure 3.9, “To Import Policies in XML Format \(Command Line\)”](#)

#### Procedure 3.5. To Export Policies in XACML Format (OpenAM Console)

- In the OpenAM console, select Access Control > *Realm Name* > Policies, and then click Export All Policies.

### **Procedure 3.6. To Import Policies in XACML Format (OpenAM Console)**

1. In the OpenAM console, select Access Control > *Realm Name* > Policies, and then click Import Policies.
  2. Browse to the XACML format file, select it, and then click Open.

### **Procedure 3.7. To Export Policies in XACML Format (Command Line)**

- Use the **ssoadm list-xacml** command.

```
$ ssoadm \
list-xacml \
--realm "/" \
--adminid amadmin \
--password-file /tmp/pwd.txt

<
?xml version="1.0" encoding="UTF-8"
?>
<PolicySet
xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
PolicyCombiningAlgId="urn...rule-combining-algorithm:deny-overrides"
Version="2014.11.25.17.41.15.597"
PolicySetId="/:2014.11.25.17.41.15.597">
<Target />
<Policy
RuleCombiningAlgId="urn...rule-combining-algorithm:deny-overrides"
Version="2014.11.25.17.40.08.067"
PolicyId="myPolicy">
<Description />
<Target>
<AnyOf>
<AllOf>
<Match
MatchId="urn...entitlement:json-subject-match">
<AttributeValue
 DataType="urn...entitlement.conditions.subject.AuthenticatedUsers">
{}
</AttributeValue>
<AttributeDesignator
 MustBePresent="true"
 DataType="urn...entitlement.conditions.subject.AuthenticatedUsers"
 AttributeId="urn...entitlement:json-subject"
 Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" />
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match
MatchId="urn...entitlement:resource-match:application:iPlanetAMWebAgentService">
<AttributeValue
 DataType="http://www.w3.org/2001/XMLSchema#string">
http://www.example.com:8000/*
</AttributeValue>
```

## Importing and Exporting Policies

```
?*
  </AttributeValue>
<AttributeDesignator
  MustBePresent="true"
  DataType="http://www.w3.org/2001/XMLSchema#string"
  AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
  Category="urn...attribute-category:resource" />
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match
  MatchId="urn...application-match">
<AttributeValue
  DataType="http://www.w3.org/2001/XMLSchema#string">
  iPlanetAMWebAgentService
</AttributeValue>
<AttributeDesignator
  MustBePresent="false"
  DataType="http://www.w3.org/2001/XMLSchema#string"
  AttributeId="urn...application-id"
  Category="urn...application-category" />
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match
  MatchId="urn...entitlement:action-match:application:iPlanetAMWebAgentService">
<AttributeValue
  DataType="http://www.w3.org/2001/XMLSchema#string">
  POST
</AttributeValue>
<AttributeDesignator
  MustBePresent="true"
  DataType="http://www.w3.org/2001/XMLSchema#string"
  AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
  Category="urn...attribute-category:action" />
</Match>
</AllOf>
<AllOf>
<Match
  MatchId="urn...entitlement:action-match:application:iPlanetAMWebAgentService">
<AttributeValue
  DataType="http://www.w3.org/2001/XMLSchema#string">
  GET
</AttributeValue>
<AttributeDesignator
  MustBePresent="true"
  DataType="http://www.w3.org/2001/XMLSchema#string"
  AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
  Category="urn...attribute-category:action" />
</Match>
</AllOf>
</AnyOf>
</Target>
<VariableDefinition
  VariableId="....entitlement.applicationName">
<AttributeValue
  DataType="http://www.w3.org/2001/XMLSchema#string">
  iPlanetAMWebAgentService
</AttributeValue>
```

## Importing and Exporting Policies

```
</AttributeValue>
</VariableDefinition>
<VariableDefinition
  VariableId="...privilege.createdBy">
  <AttributeValue
    DataType="http://www.w3.org/2001/XMLSchema#string">
      id=amadmin,ou=user,dc=openam,dc=forgerock,dc=org
    </AttributeValue>
</VariableDefinition>
<VariableDefinition
  VariableId="...privilege.lastModifiedBy">
  <AttributeValue
    DataType="http://www.w3.org/2001/XMLSchema#string">
      id=amadmin,ou=user,dc=openam,dc=forgerock,dc=org
    </AttributeValue>
</VariableDefinition>
<VariableDefinition
  VariableId="...privilege.creationDate">
  <AttributeValue
    DataType="http://www.w3.org/2001/XMLSchema#dateTime">
      2014-11-25T17:40:08.067
    </AttributeValue>
</VariableDefinition>
<VariableDefinition
  VariableId="...privilege.lastModifiedDate">
  <AttributeValue
    DataType="http://www.w3.org/2001/XMLSchema#dateTime">
      2014-11-25T17:40:08.067
    </AttributeValue>
</VariableDefinition>
<Rule
  Effect="Permit"
  RuleId="null:permit-rule">
  <Description>Permit Rule</Description>
  <Targets>
    <AnyOf>
      <AllOf>
        <Match
          MatchId="urn...entitlement:action-match:application:iPlanetAMWebAgentService">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">
              POST
            </AttributeValue>
          <AttributeDesignator
            MustBePresent="true"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            AttributeId="urn: oasis:names:tc:xacml:1.0:action:action-id"
            Category="urn...attribute-category:action" />
        </Match>
      </AllOf>
      <AllOf>
        <Match
          MatchId="urn...entitlement:action-match:application:iPlanetAMWebAgentService">
          <AttributeValue
            DataType="http://www.w3.org/2001/XMLSchema#string">
              GET
            </AttributeValue>
          <AttributeDesignator
            MustBePresent="true"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            AttributeId="urn: oasis:names:tc:xacml:1.0:action:action-id"
            Category="urn...attribute-category:action" />
        </Match>
      </AllOf>
    </AnyOf>
  </Targets>
</Rule>
```

## Importing and Exporting Policies

```
</Match>
</AllOf>
</AnyOf>
</Target>
<Condition>
<Apply
  FunctionId="urn...entitlement:json-subject-and-condition-satisfied">
<AttributeValue
  DataType="urn...entitlement.conditions.subject.AuthenticatedUsers"
  privilegeComponent="entitlementSubject">
  {}
</AttributeValue>
</Apply>
</Condition>
</Rule>
</Policy>
</PolicySet>

Policy definitions were returned under realm, /.
```

### Procedure 3.8. To Import Policies in XACML Format (Command Line)

- Use the **ssoadm create-xacml** command.

```
$ ssoadm \
  create-xacml \
  --realm "/" \
  --adminid amadmin \
  --password-file /tmp/pwd.txt \
  --xmlfile policy.xml

Policies were created under realm, /.
```

### Procedure 3.9. To Import Policies in XML Format (Command Line)

You can import policies stored in non-XACML XML format. Once imported, the policy can then be exported in the XACML format using either the OpenAM console or the **ssoadm list-xacml** command.

- To import policies stored in non-XACML XML format, use the **ssoadm create-policies** command.

```
$ ssoadm \
  create-policies \
  --realm "/" \
  --adminid amadmin \
  --password-file /tmp/pwd.txt \
  --xmlfile policy.xml

Policies were created under realm, /.
```

### **3.4 Delegating Policy Management**

To delegate policy management and other administrative tasks, use privileges. You set privileges in OpenAM console on the Privileges page for a realm.

For details see the procedure, [\*To Delegate Administration\*](#).



---

## Chapter 4

# Configuring Realms

This chapter shows how to configure OpenAM *realms*, which 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 login 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), containing OpenAM configuration data, and allowing 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.

## 4.1 Managing Realms

You create and configure realms through the console, starting from the Access Control tab > Realms table. You delegate administration for a realm by setting privileges in the realm.

- [Procedure 4.1, “To Create a New Realm”](#)
- [Procedure 4.2, “To Delegate Administration”](#)

### Procedure 4.1. To Create a New Realm

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

1. Login to the OpenAM console as OpenAM Administrator, `amadmin`.
2. On the Access Control tab > Realms table, click New to open the New Realm page, where you configure the realm.

#### Note

Do not use the names of OpenAM REST endpoints as the name of a realm. The OpenAM REST endpoint names that should not be used includes: "users", "groups", "realms", "policies" and "applications".

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

Realm/DNS aliases must follow standard FQDN conventions, such as `hr`.<sup>1</sup> `example.com` or `pay.example.com`.

3. Save your work after defining the configuration for the new realm.

### Procedure 4.2. To Delegate Administration

You can delegate administration in a realm. OpenAM grants administrative capabilities to members of groups having administrative privileges.

You can grant privileges through the OpenAM console as described below, or by using the **ssoadm add-privileges** command.

---

<sup>1</sup> The Realm/DNS Alias option refers to an FQDN that can be used to represent the realm. It is not related to the CNAME record used in DNS database zones. In other words, the Realm/DNS Alias option shown in the console does not conform to the definition of DNS Aliases described in [RFC 2219](#).

1. On the Access Control tab > Realms table, click the realm for which you want to delegate administration to view the realm configuration.
2. On the Privileges tab, click the name of the group to whom you intend to grant access.
3. Select the administrative privileges to delegate for the realm. See [Table 4.1, "OpenAM Privileges"](#) for information about OpenAM privileges.

Unless your delegation model requires a high degree of granularity, assign realm administrators the `RealmAdmin` privilege, and assign policy administrators the `PolicyAdmin` privilege.
4. Save your work.

The following table describes privileges that you can assign in the OpenAM console or by using the `ssoadm add-privileges` command:

**Table 4.1. OpenAM Privileges**

| Privilege  | Privilege Name to Use With the <code>ssoadm add-privileges</code> Command | Notes   |
|--|---|---|
| Read and write access to all realm and policy properties | <code>RealmAdmin</code>   | 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. |
| Read and write access to all log files                   | <code>LogAdmin</code>   | Subset of the <code>RealmAdmin</code> privilege.  |
| Read access to all log files                             | <code>LogRead</code>  | Subset of the <code>RealmAdmin</code> privilege.  |
| Write access to all log files                            | <code>LogWrite</code>   | Subset of the <code>RealmAdmin</code> privilege.  |
| Read and write access to all configured agents           | <code>AgentAdmin</code>   | Provides access to centralized agent configuration; subset of the <code>RealmAdmin</code> privilege.  |
| Read and write access to all federation                  | <code>FederationAdmin</code>  | Subset of the <code>RealmAdmin</code> privilege.  |

| Privilege  | Privilege Name to Use With<br>the <code>ssoadm add-privileges</code><br>Command | Notes  |
|--|---|--|
| metadata configurations  |   |  |
| REST calls for reading realms  | RealmReadAccess   | Subset of the <code>RealmAdmin</code> privilege.   |
| Read and write access only for policy properties, including REST calls | PolicyAdmin   | 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, referrals, subject types, condition types, subject attributes, and decision combiners. All other OpenAM privileges that affect policy components are included with this privilege. Subset of the <code>RealmAdmin</code> privilege. |
| REST calls for policy evaluation                                       | EntitlementRestAccess   | Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.   |
| REST calls for reading policies  | PrivilegeRestReadAccess   | Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.   |
| REST calls for managing policies                                       | PrivilegeRestAccess   | Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.   |
| REST calls for reading policy referrals                                | ReferralsReadAccess   | Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.   |
| REST calls for modifying policy referrals                              | ReferralsModifyAccess   | Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.   |
| REST calls for reading policy applications                             | ApplicationReadAccess   | Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.   |
| REST calls for modifying policy applications                           | ApplicationModifyAccess   | Subset of the <code>RealmAdmin</code> and <code>PolicyAdmin</code> privileges.   |

| Privilege                                       | Privilege Name to Use With<br>the <code>ssoadm add-privileges</code><br>Command | Notes  |
|---|---|--|
| 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. |

---

## 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 4.3, "To Specify the Realm and Application for Policy Decisions"](#)
- [Procedure 4.4, "To Configure a Web or J2EE Agent for Login to a Realm"](#)

### Procedure 4.3. To Specify the Realm and Application for Policy Decisions

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

1. In the OpenAM console, browse to Access Control > *Realm Name* > Agents > *Web or Java EE Agent Type* > *Agent Name* > OpenAM Services > Policy Client Service.
2. Set the Realm and Application.

For example, if the realm is /hr and the application is myHRApp:

- Realm: /hr
  - Application: myHRApp
3. Save your work.

### **Procedure 4.4. To Configure a Web or J2EE Agent for Login 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 OpenAM console, browse to Access Control > *Realm Name* > Agents > *Web or Java EE Agent Type* > *Agent Name* > OpenAM Services.
2. Add login and logout URLs, including the realm 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/UI/Login?realm=hr`
  - Logout URL: `http://openam.example.com:8080/openam/UI/Logout?realm=hr`
3. Save your work.

## **4.3 Configuring Data Stores**

When you first set up a realm, the new realm inherits the data store from the parent realm. 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.

### **Procedure 4.5. To Configure a Data Store**

1. In OpenAM console, browse to Access Control > *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.

- [\*Hints for Configuring Active Directory Data Stores\*](#)
- [\*Hints for Configuring Active Directory Application Mode \(ADAM\) Data Stores\*](#)
- [\*Hints for Configuring Database Repository \(Early Access\) Data Stores\*](#)
- [\*Hints for Configuring Generic LDAPv3 Data Stores\*](#)
- [\*Hints for Configuring OpenDJ Data Stores\*](#)
- [\*Hints for Configuring Sun/Oracle DSEE Data Stores\*](#)
- [\*Hints for Configuring Tivoli Directory Server Data Stores\*](#)

5. 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.

6. 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](#) as described in the chapter on authentication to use your realm data store, rather than the inherited settings.

#### **4.3.1 Hints for Configuring Active Directory Data Stores**

Use these hints 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.

**ssoadm** attribute: idRepoLoadSchema

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 SSL/TLS Enabled

Whether to use LDAPS or StartTLS to connect to the directory server. If you enable SSL/TLS, 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-ssl-enabled

Default: false

#### 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 Plug-in 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 Plug-in 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 Plug-in 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, devicePrintProfiles, displayName, distinguishedName, dn, employeeNumber, givenName, 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-

## Hints for Configuring Active Directory Data Stores

---

```
reset-force-reset, iplanet-am-user-password-reset-options, iplanet-am-user-password-reset-question-answer, iplanet-am-user-success-url, mail, name, objectclass, objectGUID, postalAddress, preferredlanguage, preferredLocale, preferreddatetimezone, 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, sunIdentityServerPPPEncryPTKey, sunIdentityServerPPFacadegreeGreetmesound, sunIdentityServerPPFacadeGreetSound, sunIdentityServerPPFacadeMugShot, sunIdentityServerPPFacadeNamePronounced, sunIdentityServerPPFacadeWebSite, sunIdentityServerPPInformalName, sunIdentityServerPPLegalIdentityAltIdType, sunIdentityServerPPLegalIdentityAltIdValue, sunIdentityServerPPLegalIdentityDOB, sunIdentityServerPPLegalIdentityGender, sunIdentityServerPPLegalIdentityLegalName, sunIdentityServerPPLegalIdentityMaritalStatus, sunIdentityServerPPLegalIdentityVATIdType, sunIdentityServerPPLegalIdentityVATIdValue, sunIdentityServerPPMsgContact, sunIdentityServerPPSignKey, telephoneNumber, unicodePwd, userAccountControl, userpassword, userPrincipalname
```

### 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: 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.

---

Hints for Configuring  
Active Directory Application  
Mode (ADAM) Data Stores

---

**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: false

DN Cache Size

Maximum number of DNs cached when caching is enabled

**ssoadm** attribute: sun-idrepo-ldapv3-dncache-size

Default: 1500 items

#### 4.3.2 Hints for Configuring Active Directory Application Mode (ADAM) Data Stores

Use these hints 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.

**ssoadm** attribute: idRepoLoadSchema

## Hints for Configuring Active Directory Application Mode (ADAM) Data Stores

---

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 SSL/TLS Enabled

Whether to use LDAPS or StartTLS to connect to the directory server. If you enable SSL/TLS, OpenAM must be able to trust server certificates, either because the server certificates were signed by a CA whose certificate is

## Hints for Configuring Active Directory Application Mode (ADAM) Data Stores

---

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-ssl-enabled

Default: false

### 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

Hints for Configuring  
Active Directory Application  
Mode (ADAM) Data Stores

---

Default: 10

LDAPv3 Plug-in 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 Plug-in 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 Plug-in 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

## Hints for Configuring Active Directory Application Mode (ADAM) Data Stores

---

**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, iplanet-am-auth-configuration-service, iplanet-am-managed-person, iplanet-am-user-service, iPlanetPreferences, organizationalPerson, person, sunAMAuthAccountLockout, sunFederationManagerDataStore, sunFMSAML2NameIdentifier, 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.

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-attributes

Default: assignedDashboard, cn, devicePrintProfiles, displayName, distinguishedName, dn, employeeNumber, givenName, iplanet-am-auth-configuration, iplanet-am-session-add-session-listener-on-all-sessions,

## Hints for Configuring Active Directory Application Mode (ADAM) Data Stores

---

```
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, name, objectclass, objectGUID, postalAddress, preferredlanguage,
preferredLocale, preferreddatetimezone, 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, sunIdentityServerPPFacadeGreetmesound,
sunIdentityServerPPFacadeGreetSound, sunIdentityServerPPFacadeMugShot,
sunIdentityServerPPFacadeNamePronounced,
sunIdentityServerPPFacadeWebSite, sunIdentityServerPPIinformalName,
sunIdentityServerPPLegalIdentityAltIdType,
sunIdentityServerPPLegalIdentityAltIdValue,
sunIdentityServerPPLegalIdentityDOB,
sunIdentityServerPPLegalIdentityGender,
sunIdentityServerPPLegalIdentityLegalName,
sunIdentityServerPPLegalIdentityMaritalStatus,
sunIdentityServerPPLegalIdentityVATIdType,
sunIdentityServerPPLegalIdentityVATIdValue, sunIdentityServerPPMsgContact,
sunIdentityServerPPSignKey, telephoneNumber, unicodePwd, userAccountControl,
userpassword, userPrincipalname
```

### 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

Hints for Configuring  
Active Directory Application  
Mode (ADAM) Data Stores

---

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

Hints for Configuring  
Active Directory Application  
Mode (ADAM) Data Stores

---

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*

## Hints for Configuring Database Repository (Early Access) Data Stores

---

### 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.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: false

### DN Cache Size

Maximum number of DNs cached when caching is enabled

**ssoadm** attribute: sun-idrepo-ldapv3-dncache-size

Default: 1500 items

### 4.3.3 Hints for Configuring Database Repository (Early Access) Data Stores

Use these hints 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

## Hints for Configuring Database Repository (Early Access) Data Stores

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.

**ssoadm** attribute: idRepoLoadSchema

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/openssousersdb

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

Hints for Configuring  
Database Repository  
(Early Access) Data Stores

---

Default: `jdbc:mysql://127.0.0.1:3306/test`

Connect This User to Database  
Username used when connecting directly

**ssoadm** attribute: `sun-openssodata-JDBCDBuser`

Default: `root`

Password for Connecting to Database  
Password used when connecting directly

**ssoadm** attribute: `sun-openssodata-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-openssodata-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 Plug-in Supported Types and Operations  
Map of OpenAM operations that can be performed in the specified OpenAM contexts

Hints for Configuring  
Database Repository  
(Early Access) Data Stores

---

**ssoadm** attribute: sun-openssodata-base-sunIdRepoSupportedOperations

Default: group=read,create,edit,delete, user=read,create,edit,delete,  
service

Database User Table Name

Table to store user profiles

**ssoadm** attribute: sun-openssodata-base-UserTableName

Default: opensso\_users

List of User Attributes Names in Database

Columns for user profile attributes

**ssoadm** attribute: sun-openssodata-base-UserAttrs

Default: ChangePassword, cn, employeenumber, 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,  
mail, manager, postaladdress, preferredlocale, sn, sunIdentityMSISDNNumber,  
telephonenumber, uid, userpassword

User Password Attribute Name

Column for user passwords

**ssoadm** attribute: sun-openssodata-base-UserPasswordAttr

Default: userpassword

User ID Attribute Name

Column for user IDs

**ssoadm** attribute: sun-openssodata-base-UserIDAttr

Default: uid

Attribute Name of User Status

Column to check/set user status

**ssoadm** attribute: sun-openssodata-base-UserStatusAttr

Default: inetuserstatus

User Status Active Value

Active users have the user status set to this value.

**ssoadm** attribute: sun-openssodata-base-activeValue

Default: Active

User Status Inactive Value

Inactive users have the user status set to this value.

**ssoadm** attribute: sun-openssodata-base-inactiveValue

Default: Inactive

Users Search Attribute in Database

Key for looking up user profiles by name

**ssoadm** attribute: sun-openssodata-base-config-users-search-attribute

Default: cn

Database Membership table name

Table to store group profiles

**ssoadm** attribute: sun-openssodata-base-MembershipTableName

Default: groups

Membership ID Attribute Name

Column for group IDs

**ssoadm** attribute: sun-openssodata-base-MembershipIDAttr

Default: group\_name

Membership Search Attribute in Database

Key for looking up group profiles by name

**ssoadm** attribute: sun-openssodata-base-membership-search-attribute

Default: cn

#### 4.3.4 Hints for Configuring Generic LDAPv3 Data Stores

Use these hints 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.

**ssoadm** attribute: `idRepoLoadSchema`

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 SSL/TLS Enabled

Whether to use LDAPS or StartTLS to connect to the directory server. If you enable SSL/TLS, 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-ssl-enabled

Default: false

#### 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 Plug-in 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 Plug-in 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 Plug-in 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, 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.

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-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.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: false

DN Cache Size  
Maximum number of DNs cached when caching is enabled

**ssoadm** attribute: sun-idrepo-ldapv3-dncache-size

Default: 1500 items

#### 4.3.5 Hints for Configuring OpenDJ Data Stores

Use these hints 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.

**ssoadm** attribute: idRepoLoadSchema

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 SSL/TLS Enabled

Whether to use LDAPS or StartTLS to connect to the directory server. If you enable SSL/TLS, 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-ssl-enabled

Default: false

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 Plug-in 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 Plug-in 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 Plug-in 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, 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, 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.

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-attributes

Default: `sunIdentityServerPPDemographicsBirthDay`, `uid`,  
`sunIdentityServerPPLegalIdentityLegalName`, `manager`, `assignedDashboard`,  
`sunIdentityServerPPCommonNameSN`, `userPassword`, `iplanet-am-session-get-valid-sessions`, `sunIdentityServerPPEmploymentIdentityJobTitle`, `iplanet-am-user-password-reset-question-answer`, `sunIdentityServerPPLegalIdentityDOB`,  
`sunIdentityServerPPEmergencyContact`, `sunIdentityServerPPCommonNameCN`,  
`iplanet-am-user-success-url`, `iplanet-am-user-admin-start-dn`, `iplanet-am-user-federation-info`, `userCertificate`, `sunIdentityServerPPFacadeGreetSound`,  
`sunAMAuthInvalidAttemptsData`, `sunIdentityServerPPFacadeNamePronounced`,  
`distinguishedName`, `sunIdentityServerPPDemographicsTimeZone`,  
`sunIdentityMSISDNNumber`, `iplanet-am-session-max-caching-time`, `sn`,  
`iplanet-am-session-quota-limit`, `iplanet-am-session-max-session-time`,  
`adminRole`, `sunIdentityServerPPEmploymentIdentityAlt0`, `objectClass`, `sun-fm-saml2-nameid-info`, `sunIdentityServerPPLegalIdentityMaritalStatus`,  
`iplanet-am-user-login-status`, `sunIdentityServerPPLegalIdentityAltIdType`,  
`devicePrintProfiles`, `iplanet-am-session-max-idle-time`,  
`sunIdentityServerPPFacadeGreetmesound`, `cn`, `iplanet-am-user-password-reset-options`, `telephoneNumber`, `preferredLanguage`, `iplanet-am-user-federation-info-key`, `sunIdentityServerPPMsgContact`,  
`sunIdentityServerPPLegalIdentityGender`, `iplanet-am-user-alias-list`, `sunIdentityServerPPCommonNameFN`, `caCertificate`, `inetUserStatus`,  
`sunIdentityServerPPCommonNameMN`, `sunIdentityServerPPEncryPTKey`,  
`givenName`, `memberOf`, `sunIdentityServerPPLegalIdentityVATIdValue`,  
`preferredLocale`, `iplanet-am-session-service-status`, `sun-fm-saml2-nameid-infokey`, `sunIdentityServerPPDemographicsAge`,  
`sunIdentityServerDiscoEntries`, `sunIdentityServerPPLegalIdentityVATIdType`,  
`iplanet-am-user-auth-config`, `iplanet-am-user-failure-url`,  
`sunIdentityServerPPAddressCard`, `sunIdentityServerPPCommonNamePT`,  
`dn`, `iplanet-am-session-add-session-listener-on-all-sessions`, `mail`,  
`authorityRevocationList`, `iplanet-am-user-password-reset-force-reset`, `inetUserHttpURL`, `sunIdentityServerPPLegalIdentityAltIdValue`,

```
sunIdentityServerPPCommonNameAltCN, preferredtimezone,  
sunIdentityServerPPIinformalName, sunIdentityServerPPSignKey,  
sunIdentityServerPPEmploymentIdentityOrg,  
iplanet-am-session-destroy-sessions,  
sunIdentityServerPPFacadeMugShot, sunIdentityServerPPFacadeWebSite,  
sunIdentityServerPPDemographicsDisplayLanguage, postalAddress, iplanet-am-  
auth-configuration, employeeNumber, iplanet-am-user-account-life, iplanet-  
am-user-auth-modules, sunIdentityServerPPDemographicsLanguage
```

#### 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.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 DNs cached when caching is enabled

**ssoadm** attribute: sun-idrepo-ldapv3-dncache-size

Default: 1500 items

#### 4.3.6 Hints for Configuring Sun/Oracle DSEE Data Stores

Use these hints 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.

**ssoadm** attribute: idRepoLoadSchema

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 SSL/TLS Enabled**

Whether to use LDAPS or StartTLS to connect to the directory server. If you enable SSL/TLS, 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-ssl-enabled

Default: false

**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 Plug-in 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 Plug-in 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 Plug-in 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, inetadmin, 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.

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-attributes`

Default: `sunIdentityServerPPDemographicsBirthDay, uid, sunIdentityServerPPLegalIdentityLegalName, manager, assignedDashboard, sunIdentityServerPPCommonNameSN, userPassword, iplanet-am-session-get-valid-sessions, sunIdentityServerPPEmploymentIdentityJobTitle, iplanet-am-user-password-reset-question-answer, sunIdentityServerPPLegalIdentityDOB, sunIdentityServerPPEmergencyContact, sunIdentityServerPPCommonNameCN, iplanet-am-user-success-url, iplanet-am-user-admin-start-dn, iplanet-am-user-federation-info, userCertificate, sunIdentityServerPPFacadeGreetSound, sunAMAuthInvalidAttemptsData, sunIdentityServerPPFacadeNamePronounced, distinguishedName, sunIdentityServerPPDemographicsTimeZone, sunIdentityMSISDNNumber, iplanet-am-session-max-caching-time, sn, iplanet-am-session-quota-limit, iplanet-am-session-max-session-time, adminRole, sunIdentityServerPPEmploymentIdentityAlt0, objectClass, sun-fm-saml2-nameid-info, sunIdentityServerPPLegalIdentityMaritalStatus, iplanet-am-user-login-status, sunIdentityServerPPLegalIdentityAltIdType, devicePrintProfiles, iplanet-am-session-max-idle-time, sunIdentityServerPPFacadeGreetmesound, cn, iplanet-am-user-password-reset-options, telephoneNumber, preferredlanguage, iplanet-am-user-federation-info-key, sunIdentityServerPPMsgContact, sunIdentityServerPPLegalIdentityGender, iplanet-am-user-`

```
alias-list, sunIdentityServerPPCommonNameFN, caCertificate,  
inetUserStatus, sunIdentityServerPPCommonNameMN,  
sunIdentityServerPPEncryPTKey, givenName, memberOf, iplanet-am-  
static-group-dn, sunIdentityServerPPLegalIdentityVATIdValue,  
preferredLocale, iplanet-am-session-service-status, sun-fm-  
saml2-nameid-infokey, sunIdentityServerPPDemographicsAge,  
sunIdentityServerDiscoEntries, sunIdentityServerPPLegalIdentityVATIdType,  
iplanet-am-user-auth-config, iplanet-am-user-failure-url,  
sunIdentityServerPPAddressCard, sunIdentityServerPPCommonNamePT,  
dn, iplanet-am-session-add-session-listener-on-all-sessions, mail,  
authorityRevocationList, iplanet-am-user-password-reset-force-  
reset, inetUserHttpURL, sunIdentityServerPPLegalIdentityAltIdValue,  
sunIdentityServerPPCommonNameAltCN, preferreddatetimezone,  
sunIdentityServerPPIformalName, sunIdentityServerPPSignKey,  
sunIdentityServerPPEmploymentIdentityOrg,  
iplanet-am-session-destroy-sessions,  
sunIdentityServerPPFacadeMugShot, sunIdentityServerPPFacadeWebSite,  
sunIdentityServerPPDemographicsDisplayLanguage, postalAddress, iplanet-am-  
auth-configuration, employeeNumber, iplanet-am-user-auth-modules, iplanet-  
am-user-account-life, sunIdentityServerPPDemographicsLanguage
```

#### 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 DNs cached when caching is enabled

**ssoadm** attribute: sun-idrepo-ldapv3-dncache-size

Default: 1500 items

#### 4.3.7 Hints for Configuring Tivoli Directory Server Data Stores

Use these hints 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.

**ssoadm** attribute: idRepoLoadSchema

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 SSL/TLS Enabled

Whether to use LDAPS or StartTLS to connect to the directory server. If you enable SSL/TLS, 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-ssl-enabled

Default: false

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 Plug-in 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 Plug-in 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 Plug-in 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.

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-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`, `sunIdentityServerPPIinformalName`, `sunIdentityServerPPLegalIdentityAltIdType`, `sunIdentityServerPPLegalIdentityAltIdValue`, `sunIdentityServerPPLegalIdentityDOB`, `sunIdentityServerPPLegalIdentityGender`,

```
sunIdentityServerPPLegalIdentityLegalName,  
sunIdentityServerPPLegalIdentityMaritalStatus,  
sunIdentityServerPPLegalIdentityVATIdType,  
sunIdentityServerPPLegalIdentityVATIdValue, sunIdentityServerPPMsgContact,  
sunIdentityServerPPSSignKey, 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 DNs cached when caching is enabled

**ssoadm** attribute: sun-idrepo-ldapv3-dncache-size

Default: 1500 items



---

## **Chapter 5**

# **Configuring 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 in 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, but usually 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 such that you manage all agent configuration changes centrally. This chapter describes how to set up policy agent profiles in OpenAM for centralized configuration.

## **5.1**

### **Open Identity Gateway or Policy Agent?**

OpenAM includes both the [Open Identity Gateway \(OpenIG\)](#) and also a variety of policy agents. Both OpenIG and also the policy agents enforce policy, redirecting users to authenticate when necessary, and controlling access to protected resources. Yet, the Identity Gateway 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.

The OpenIG allows you 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, where you can install them, 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.

Of course, for organizations with both servers where 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 the OpenIG for the latter.

## 5.2 Kinds of Agent Profiles

When you open the OpenAM console to configure agents for the top level realm, you can choose from a number of different types of agents. Web and J2EE policy agents are the most common, requiring the least integration effort.

### 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.

### Web Service Provider

WSP agents are for use with Web Services Security.

### Web Service Client

WSC agents are for use with Web Services Security.

### STS Client

The Security Token Service client agent is for securing requests to the Security Token Service.

### 2.2 Agents

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

### OAuth 2.0 Client Agent

OAuth 2.0 clients are registered using this type of policy agent 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.

## 5.3 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 5.1. To Create an Agent Profile

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

1. On the Access Control tab page of the OpenAM console, click the link for the realm in which you manage agents.
2. Click the Agents tab, click the tab page for the kind of agent you want to create, and then click the New... button in the Agent table.
3. Provide a name for the agent, and also the URLs to OpenAM and to the application to protect, then click Create. Note that for Java EE policy agents, an example URL must include the agentapp context: `http://www.example.com:8080/agentapp`.
4. After creating the agent profile, you can click the link to the new profile to adjust and export the configuration.

### Procedure 5.2. To Create an Agent Profile Group & Inherit Settings

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

1. On the Access Control tab page of the OpenAM console, click the link for the realm in which you manage agents.
2. Click the Agents tab, click the tab page for the kind of agent you want to create, and then click the New... button in the Group table.

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

3. 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 5.3. 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 as described in the *Installation Guide* procedure, [To Set Up Administration Tools](#).
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.sun.identity.agents.config.agenturi.prefix=http://www.example.com:80/amagent
com.sun.identity.agents.config.cdsso.cdc servlet.url[0]=https://openam.example.com:8443/
    openam/cdc servlet
com.sun.identity.agents.config.fqdn.default=www.example.com
com.sun.identity.agents.config.login.url[0]=http://openam.example.com:8443/
    openam/UI/Login
com.sun.identity.agents.config.logout.url[0]=http://openam.example.com:8443/
    openam/UI/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
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. Set up a password file used when authenticating to OpenAM. The password file must be read-only for the user who creates the policy agent profile, and must not be accessible to other users.

```
$ echo password > /tmp/pwd.txt
$ chmod 400 /tmp/pwd.txt
```

4. Create the profile in OpenAM.

```
$ ssoadm \
create-agent \
--realm / \
--agentname myAgent \
--agenttype J2EE \
--adminid amadmin \
--password-file /tmp/pwd.txt \
--datafile myAgent.properties

Agent configuration was created.
```

At this point you can view the profile in OpenAM Console under Access Control > *Realm Name* > Agents to make sure the configuration is what you expect.

## 5.4 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 however you delegate policy agent installation, then you might not want to share OpenAM administrator credentials with everyone who installs policy agents.

### Procedure 5.4. To Create Agent Administrators for a Realm

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

1. In OpenAM console, browse to Access Control > *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.

## 5.5 Configuring Web Policy Agents

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 OpenAM console. Alternatively, you can choose to store the agent configuration locally and configure the agent by changing values in the properties file. This section covers centralized configuration, indicating the corresponding properties for use in a local configuration file where applicable.<sup>1</sup>

### Tip

To show the agent properties in configuration file format that correspond to what you see in the console, click Export Configuration after editing agent properties.

This corresponds to the local Java properties configuration file that is set up when you install an agent, for example in `Agent_001/config/OpenSSOAgentConfiguration.properties`.

After changing properties specified as "Hot swap: no" you must restart the agent's container for the changes to take effect.

### 5.5.1 Configuring Web Policy Agent Global Properties

This section covers global web agent properties. After creating the agent profile, you access these properties in the OpenAM console under `Access Control > Realm Name > Agents > Web > Agent Name > Global`.

---

<sup>1</sup>The configuration file syntax is that of a standard Java properties file. See `java.util.Properties.load` for a description of the format. The value of a property specified multiple times is not defined.

## Profile properties

### Group

For assigning the agent to a previously configured web agent group in order to inherit selected properties from the group.

### Password

Agent password used when creating the password file and when installing the agent.

### Status

Status of the agent configuration.

### Location of Agent Configuration Repository

Indicates agent's configuration located either on agent's host or centrally on OpenAM server.

If you change this to a local configuration, you can no longer manage the policy agent configuration through OpenAM console.

Property: `com.sun.identity.agents.config.repository.location`

### Agent Configuration Change Notification

Enable agent to receive notification messages from OpenAM server for configuration changes.

Property: `com.sun.identity.agents.config.change.notification.enable`

### Enable Notifications

If enabled, the agent receives policy updates from the OpenAM notification mechanism to maintain its internal cache. If disabled, the agent must poll OpenAM for changes.

Property: `com.sun.identity.agents.config.notification.enable`

Hot swap: no

### Agent Notification URL

URL used by agent to register notification listeners.

Property: `com.sun.identity.client.notification.url`

Hot swap: no

### Agent Deployment URI Prefix

The default value is *agent-root-URL/amagent*.

Property: `com.sun.identity.agents.config.agenturi.prefix`

Hot swap: yes

Configuration Reload Interval

Interval in minutes to fetch agent configuration from OpenAM. Used if notifications are disabled. Default: 60.

Property: `com.sun.identity.agents.config.polling.interval`

Hot swap: no

Configuration Cleanup Interval

Interval in minutes to cleanup old agent configuration entries unless they are referenced by current requests. Default: 30.

Property: `com.sun.identity.agents.config.cleanup.interval`

Hot swap: no

Agent Root URL for CDSSO

The agent root URL 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 the goto URLs match one of the agent root URLs for CDSSO.

## General properties

SSO Only Mode

When enabled, the agent enforces authentication, so that upon verification of the user's identity, the user receives a session token.

Property: `com.sun.identity.agents.config.sso.only`

Resources Access Denied URL

The URL of the customized access denied page. If no value is specified (default), then the agent returns an HTTP status of 403 (Forbidden).

Property: `com.sun.identity.agents.config.access.denied.url`

Agent Debug Level

Default is Error. Increase to Message or even All for fine-grained detail.

Property: `com.sun.identity.agents.config.debug.level`

You can set the level in the configuration file by module using the format `module[:level][,module[:level]]*`, where `module` is one of `AuthService`, `NamingService`, `PolicyService`, `SessionService`, `PolicyEngine`, `ServiceEngine`, `Notification`, `PolicyAgent`, `RemoteLog`, or `all`, and `level` is one of the following.

- 0: Disable logging from specified module

At this level the agent nevertheless logs messages having the level value `always`.

- 1: Log error messages
- 2: Log warning and error messages
- 3: Log info, warning, and error messages
- 4: Log debug, info, warning, and error messages
- 5: Like level 4, but with even more debugging messages

When you omit `level`, the agent uses the default level, which is the level associated with the `all` module.

The following example used in the local configuration sets the log overall level to `debug` for all messages.

```
com.sun.identity.agents.config.debug.level=all:5
```

#### Agent Debug File Rotation

When enabled, rotate the debug file when specified file size is reached.

Property: `com.sun.identity.agents.config.debug.file.rotate`

#### Agent Debug File Size

Debug file size in bytes beyond which the log file is rotated. The minimum is 1048576 bytes (1 MB), and lower values are reset to 1 MB. OpenAM console sets a default of 10 MB.

Property: `com.sun.identity.agents.config.debug.file.size`

## Audit properties

### Audit Access Types

Types of messages to log based on user URL access attempts.

Property: `com.sun.identity.agents.config.audit.accessstype`

Valid values for the configuration file property include LOG\_NONE, LOG\_ALLOW, LOG\_DENY, and LOG\_BOTH.

**Audit Log Location**

Specifies where audit messages are logged. By default, audit messages are logged remotely.

Property: `com.sun.identity.agents.config.log.disposition`

Valid values for the configuration file property include REMOTE, LOCAL, and ALL.

**Remote Log Filename**

Name of file stored on OpenAM server that contains agent audit messages if log location is remote or all.

Property: `com.sun.identity.agents.config.remote.logfile`

Hot swap: no

**Remote Audit Log Interval**

Periodic interval in minutes in which audit log messages are sent to the remote log file.

Property: `com.sun.identity.agents.config.remote.log.interval`

Default: 5

Hot swap: no

**Rotate Local Audit Log**

When enabled, audit log files are rotated when reaching the specified size.

Property: `com.sun.identity.agents.config.local.log.rotate`

**Local Audit Log Rotation Size**

Beyond this size limit in bytes the agent rotates the local audit log file if rotation is enabled.

Property: `com.sun.identity.agents.config.local.log.size`

Default: 50 MB

**Fully Qualified Domain Name Checking properties**

**FQDN Check**

Enables checking of FQDN default value and FQDN map values.

Property: `com.sun.identity.agents.config.fqdn.check.enable`

#### FQDN Default

Fully qualified domain name that the users should use in order to access resources. Without this value, the web server can fail to start, thus you set the property on agent installation, and only change it when absolutely necessary.

This property ensures that when users access protected resources on the web server without specifying the FQDN, the agent can redirect the users to URLs containing the correct FQDN.

Property: `com.sun.identity.agents.config.fqdn.default`

#### FQDN Virtual Host Map

Enables virtual hosts, partial hostname and IP address to access protected resources. Maps invalid or virtual name keys to valid FQDN values so the agent can properly redirect users and the agents receive cookies belonging to the domain.

To map `myserver` to `myserver.mydomain.example`, enter `myserver` in the Map Key field, and enter `myserver.mydomain.example` in the Corresponding Map Value field. This corresponds to `com.sun.identity.agents.config.fqdn.mapping[myserver]= myserver.mydomain.example`.

Invalid FQDN values can cause the web server to become unusable or render resources inaccessible.

Property: `com.sun.identity.agents.config.fqdn.mapping`

### 5.5.2 Configuring Web Policy Agent Application Properties

This section covers application web agent properties. After creating the agent profile, you access these properties in the OpenAM console under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Application.

#### Not Enforced URL Processing properties

##### Ignore Path Info for Not Enforced URLs

When enabled, the path info and query are stripped from the request URL before being compared with the URLs of the not enforced list for those URLs containing a wildcard character. This prevents a user from accessing `http://host/index.html` by requesting `http://host/index.html/hack.gif` when the not enforced list includes `http://host/*.gif`.

#### Note

This setting is not supported by the Varnish Cache agent.

## Configuring Web Policy Agent Application Properties

---

For a more generally applicable setting, see [Ignore Path Info properties](#).

Property: com.sun.identity.agents.config.ignore.path.info.for.not.enforced.list

Enable Regular Expressions for Not Enforced URLs (Not yet in OpenAM console)  
Enable use of [Perl-compatible regular expressions](#) in Not Enforced URL settings by using the following property under Advanced > Custom Properties in the agent profile.

```
com.forgerock.agents.notenforced.url.regex.enable=true
```

### Not Enforced URLs

List of URLs for which no authentication is required. You can use wildcards to define a pattern for a URL.

The \* wildcard matches all characters except question mark (?), cannot be escaped, and spans multiple levels in a URL. Multiple forward slashes do not match a single forward slash, so \* matches mult/iple/dirs, yet mult/\*/dirs does not match mult dirs.

The -\* wildcard matches all characters except forward slash (/) or question mark (?), and cannot be escaped. As it does not match /, -\* does not span multiple levels in a URL.

OpenAM does not let you mix \* and -\* in the same URL.

Examples include `http://www.example.com/logout.html`, `http://www.example.com/images/*`, `http://www.example.com/css/-*-`, and `http://www.example.com/*.jsp?locale=*`.

Trailing forward slashes are not recognized as part of a resource name. Therefore `http://www.example.com/images//` and `http://www.example.com/images` are equivalent.

Property: com.sun.identity.agents.config.notenforced.url

If you enabled use of [Perl-compatible regular expressions](#) to match Not Enforced URLs, then all your settings must be done using regular expressions. (Do not mix settings; use either the mechanism described above or Perl-compatible regular expressions, but not both.)

The following example shows settings where no authentication is required for URLs whose path ends /publicA or /publicB (with or without query string parameters), and no authentication is required to access .png, .jpg, .gif, .js, or .css files under URLs that do not contain /protectedA/ or /protectedB/.

```
.*/(PublicServletA|PublicServletB)(\?.*|$)
^(?!.*(/protectedA//protectedB/)).*\.(png|jpg|gif|js|css)(\?.*|$)
```

#### Invert Not Enforced URLs

Only enforce not enforced list of URLs. In other words, enforce policy only for those URLs and patterns specified in the list.

Property: `com.sun.identity.agents.config.notenforced.url.invert`

#### Fetch Attributes for Not Enforced URLs

When enabled, the agent fetches profile, response, and session attributes that are mapped by doing policy evaluation, and forwards these attributes to not enforced URLs.

Property: `com.sun.identity.agents.config.notenforced.url.attributes.enable`

### Not Enforced IP Processing Properties

#### Not Enforced Client IP List

No authentication and authorization are required for the requests coming from these client IP addresses.

Property: `com.sun.identity.agents.config.notenforced.ip`

#### Note

Loopback addresses are not considered valid IPs on the Not Enforced IP list. If specified, the policy agent ignores the loopback address.

#### CIDR Client IP Specification (Not yet in OpenAM console)

As of version 3.0.4, web policy agents with this property set to `cidr` can use IPv4 netmasks and IP ranges instead of wildcards as values for Not Enforced Client IP addresses. Version 3.0.5 adds support for IPv6, including the IPv6 loopback address, `::1`.

When the parameter is defined, wildcards are ignored in Not Enforced Client IP settings. Instead, you can use settings such as those shown in the following examples.

#### Netmask Example

To disable policy agent enforcement for addresses in 192.168.1.1 to 192.168.1.255, use the following setting.

```
com.sun.identity.agents.config.notenforced.ip = 192.168.1.1/24
```

The following example shows an IPv6 address with a corresponding network mask.

```
com.sun.identity.agents.config.notenforced.ip = 2001:5c0:9168:0:0:0:2/128
```

Currently the policy agent stops evaluating properties after reaching an invalid netmask in the list.

#### IP Range Example

To disable policy agent enforcement for addresses between 192.168.1.1 to 192.168.4.3 inclusive, use the following setting.

```
com.sun.identity.agents.config.notenforced.ip = 192.168.1.1-192.168.4.3
```

The following example shows a range of IPv6 addresses. The example is displayed over two lines for formatting purposes.

```
com.sun.identity.agents.config.notenforced.ip = \
2001:5c0:9168:0:0:0:1-2001:5c0:9168:0:0:0:2
```

Property: com.forgerock.agents.config.notenforced.ip.handler

Hot swap: no

#### Client IP Validation

When enabled, validate that the subsequent browser requests come from the same IP address that the SSO token is initially issued against.

Property: com.sun.identity.agents.config.client.ip.validation.enable

### Profile Attributes Processing properties

#### Profile Attribute Fetch Mode

When set to HTTP\_COOKIE or HTTP\_HEADER, profile attributes are introduced into the cookie or the headers, respectively.

Property: com.sun.identity.agents.config.profile.attribute.fetch.mode

#### Profile Attribute Map

Maps the profile attributes to HTTP headers for the currently authenticated user. Map Keys are LDAP attribute names, and Map Values are HTTP header names.

To populate the value of profile attribute CN under CUSTOM-Common-Name: enter CN in the Map Key field, and enter CUSTOM-Common-Name in the Corresponding Map Value field. This corresponds to com.sun.identity.agents.config.profile.attribute.mapping[cn]=CUSTOM-Common-Name.

In most cases, in a destination application where an HTTP header name shows up as a request header, it is prefixed by HTTP\_, lower case letters become upper case, and hyphens (-) become underscores (\_). For example, common-name becomes HTTP\_COMMON\_NAME.

Property: com.sun.identity.agents.config.profile.attribute.mapping

## Response Attributes Processing properties

### Response Attribute Fetch Mode

When set to HTTP\_COOKIE or HTTP\_HEADER, response attributes are introduced into the cookie or the headers, respectively.

Property: com.sun.identity.agents.config.response.attribute.fetch.mode

### Response Attribute Map

Maps the policy response attributes to HTTP headers for the currently authenticated user. The response attribute is the attribute in the policy response to be fetched.

To populate the value of response attribute uid under CUSTOM-User-Name: enter uid in the Map Key field, and enter CUSTOM-User-Name in the Corresponding Map Value field. This corresponds to com.sun.identity.agents.config.response.attribute.mapping[uid]=Custom-User-Name.

In most cases, in a destination application where an HTTP header name shows up as a request header, it is prefixed by HTTP\_, lower case letters become upper case, and hyphens (-) become underscores (\_). For example, response-attr-one becomes HTTP\_RESPONSE\_ATTR\_ONE.

Property: com.sun.identity.agents.config.response.attribute.mapping

## Session Attributes Processing properties

### Session Attribute Fetch Mode

When set to HTTP\_COOKIE or HTTP\_HEADER, session attributes are introduced into the cookie or the headers, respectively.

Property: com.sun.identity.agents.config.session.attribute.fetch.mode

### Session Attribute Map

Maps session attributes to HTTP headers for the currently authenticated user. The session attribute is the attribute in the session to be fetched.

To populate the value of session attribute UserToken under CUSTOM-userid: enter UserToken in the Map Key field, and enter CUSTOM-userid in the Corresponding Map Value field. This corresponds to com.sun.identity.agents.config.session.attribute.mapping[UserToken] =CUSTOM-userid.

In most cases, in a destination application where an HTTP header name shows up as a request header, it is prefixed by HTTP\_, lower case letters become upper case, and hyphens (-) become underscores (\_). For example, success-url becomes HTTP\_SUCCESS\_URL.

Property: com.sun.identity.agents.config.session.attribute.mapping

### Common Attributes Fetching Processing properties

Attribute Multi Value Separator

Specifies separator for multiple values. Applies to all types of attributes such as profile, session and response attributes. Default: |.

Property: com.sun.identity.agents.config.attribute.multi.value.separator

## 5.5.3 Configuring Web Policy Agent SSO Properties

This section covers SSO web agent properties. After creating the agent profile, you access these properties in the OpenAM console under Access Control > *Realm Name* > Agents > Web > *Agent Name* > SSO

### Cookie properties

Cookie Name

Name of the SSO Token cookie used between the OpenAM server and the agent. Default: iPlanetDirectoryPro.

Property: com.sun.identity.agents.config.cookie.name

Hot swap: no

Cookie Security

When enabled, the agent marks cookies secure, sending them only if the communication channel is secure.

Property: com.sun.identity.agents.config.cookie.secure

Hot swap: no

HTTPOnly Cookies (Not yet in OpenAM console)

As of version 3.0.5, web policy agents with this property set to true mark cookies as HTTPOnly, to prevent scripts and third-party programs from accessing the cookies.

Property: com.sun.identity.cookie.httponly

### Cross Domain SSO properties

Cross Domain SSO

Enables Cross Domain Single Sign On.

Property: com.sun.identity.agents.config.cdsso.enable

#### CDSSO Servlet URL

List of URLs of the available CDSSO controllers that the agent can use for CDSSO processing. For example, `http://openam.example.com:8080/openam/cdc servlet.`

Property: `com.sun.identity.agents.config.cdss o.cdc servlet.url`

#### Cookies Domain List

List of domains, such as `.example.com`, in which cookies have to be set in CDSSO. If this property is left blank, then the fully qualified domain name of the cookie for the agent server is used to set the cookie domain, meaning that a host cookie rather than a domain cookie is set.

To set the list to `.example.com`, and `.example.net` using the configuration file property, include the following.

```
com.sun.identity.agents.config.cdss o.cookie.domain[0]=.example.com  
com.sun.identity.agents.config.cdss o.cookie.domain[1]=.example.net
```

Property: `com.sun.identity.agents.config.cdss o.cookie.domain`

### Cookie Reset properties

#### Cookie Reset

When enabled, agent resets cookies in the response before redirecting to authentication.

Property: `com.sun.identity.agents.config.cookie.reset.enable`

#### Cookie Reset Name List

List of cookies in the format `name[=value] [;Domain=value]`.

Concrete examples include the following with two list items configured.

- `LtpaToken`, corresponding to `com.sun.identity.agents.config.cookie.reset[0]=LtpaToken`. The default domain is taken from FQDN Default.
- `token=value;Domain=subdomain.domain.com`, corresponding to `com.sun.identity.agents.config.cookie.reset[1]= token=value;Domain=subdomain.domain.com`

Property: `com.sun.identity.agents.config.cookie.reset`

## 5.5.4 Configuring Web Policy Agent OpenAM Services Properties

This section covers OpenAM services web agent properties. After creating the agent profile, you access these properties in the OpenAM console under Access Control > *Realm Name* > Agents > Web > *Agent Name* > OpenAM Services.

## Login URL properties

### OpenAM Login URL

OpenAM login page URL, such as `http://openam.example.com:8080/openam/UI/Login`, to which the agent redirects incoming users without sufficient credentials so that they can authenticate.

Property: `com.sun.identity.agents.config.login.url`

### OpenAM Conditional Login URL (Not yet in OpenAM console)

To conditionally redirect users based on the incoming request URL, set this property.

This takes the incoming request domain to match, a vertical bar ( | ), and then a comma-separated list of URLs to which to redirect incoming users.

If the domain before the vertical bar matches an incoming request URL, then the policy agent uses the list of URLs to determine how to redirect the user-agent. If the global property FQDN Check (`com.sun.identity.agents.config.fqdn.check.enable`) is enabled for the policy agent, then the policy agent iterates through the list until it finds an appropriate redirect URL that matches the FQDN check. Otherwise, the policy agent redirects the user-agent to the first URL in the list.

Property: `com.forgerock.agents.conditional.login.url`

Examples: `com.forgerock.agents.conditional.login.url[0]= login.example.com|http://openam1.example.com/openam/UI/Login, http://openam2.example.com/openam/UI/Login, com.forgerock.agents.conditional.login.url[1]= signin.example.com|http://openam3.example.com/openam/UI/Login, http://openam4.example.com/openam/UI/Login`

If CDSSO is enabled for the policy agent, then this property takes CDSSO Servlet URLs for its values (`com.sun.identity.agents.config.cdsso.cdc servlet.url`), rather than OpenAM login URLs.

CDSSO examples: `com.forgerock.agents.conditional.login.url[0]= login.example.com|http://openam1.example.com/openam/cdc servlet, http://openam2.example.com/openam/cdc servlet, com.forgerock.agents.conditional.login.url[1]= signin.example.com|http://openam3.example.com/openam/cdc servlet, http://openam4.example.com/openam/cdc servlet`

### Agent Connection Timeout

Timeout period in seconds for an agent connection with OpenAM auth server.

Property: `com.sun.identity.agents.config.auth.connection.timeout`

Default: 2

#### Polling Period for Primary Server

Interval in minutes, agent polls to check the primary server is up and running. Default: 5.

Property: `com.sun.identity.agents.config.poll.primary.server`

Hot swap: no

#### Logout URL properties

##### OpenAM Logout URL

OpenAM logout page URL, such as `http://openam.example.com:8080/openam/UI/Logout`.

Property: `com.sun.identity.agents.config.logout.url`

##### Enable Logout URL Redirect (Not yet in OpenAM console)

Logout URL redirect is enabled by default.

When this is disabled, instead of redirecting the user-agent, the policy agent performs session logout in the background and then continues processing access to the current URL. Disable this using Advanced > Custom Properties in the agent profile.

```
com.forgerock.agents.config.logout.redirect.disable=true
```

#### Agent Logout URL properties

##### Logout URL List

List of application logout URLs, such as `http://www.example.com/logout.html`. The user is logged out of the OpenAM session when these URLs are accessed. When using this property, specify a value for the Logout Redirect URL property.

Property: `com.sun.identity.agents.config.agent.logout.url`

##### Agent Logout URL Regular Expression (Not yet in OpenAM console)

[Perl-compatible regular expression](#) that matches logout URLs. Set this using Advanced > Custom Properties in the agent profile.

For example, to match URLs with `protectedA` or `protectedB` in the path and `op=logout` in the query string, use the following setting.

```
com.forgerock.agents.agent.logout.url.regex= \
.*(/protectedA\?|/protectedB\?/).*(\&op=logout\&)(.*|$)
```

When you use this property, the agent ignores the settings for Logout URL List.

**Logout Cookies List for Reset**

Cookies to be reset upon logout in the same format as the cookie reset list.

Property: `com.sun.identity.agents.config.logout.cookie.reset`

**Logout Redirect URL**

User gets redirected to this URL after logout. Specify this property alongside a Logout URL List.

Property: `com.sun.identity.agents.config.logout.redirect.url`

## **Policy Client Service properties**

**Policy Cache Polling Period**

Polling interval in minutes during which an entry remains valid after being added to the agent's cache.

Property: `com.sun.identity.agents.config.policy.cache.polling.interval`

Hot swap: no

**SSO Cache Polling Period**

Polling interval in minutes during which an SSO entry remains valid after being added to the agent's cache.

Property: `com.sun.identity.agents.config.sso.cache.polling.interval`

Hot swap: no

**User ID Parameter**

Agent sets this value for User Id passed in the session from OpenAM to the `REMOTE_USER` server variable. Default: `UserToken`.

Property: `com.sun.identity.agents.config.userid.param`

**User ID Parameter Type**

User ID can be fetched from either SESSION or LDAP attributes. Default: `SESSION`.

Property: `com.sun.identity.agents.config.userid.param.type`

**Fetch Policies from Root Resource**

When enabled, the agent caches the policy decision of the resource and all resources from the root of the resource down. For example, if the resource is `http://host/a/b/c`, then the root of the resource is `http://host/`. This setting can be useful when a client is expect to access multiple resources on the same path. Yet, caching can be expensive if very many policies are defined for the root resource.

## Configuring Web Policy Agent OpenAM Services Properties

---

Property: `com.sun.identity.agents.config.fetch.from.root.resource`

Default: false

Hot swap: no

### Retrieve Client Hostname

When enabled, get the client hostname through DNS reverse lookup for use in policy evaluation. This setting can impact performance.

Property: `com.sun.identity.agents.config.get.client.host.name`

### Policy Clock Skew

Time in seconds used adjust time difference between agent system and OpenAM. Clock skew in seconds = AgentTime - OpenAMServerTime.

Use this property to adjust for small time differences encountered despite use of a time synchronization service. When this property is not set and agent time is greater than OpenAM server time, the agent can make policy calls to the OpenAM server before the policy subject cache has expired, or you can see infinite redirection occur.

Property: `com.sun.identity.agents.config.policy.clock.skew`

Hot swap: no

### Realm

Realm where OpenAM starts policy evaluation for this policy agent.

Default: / (top level realm)

Edit this property when OpenAM should start policy evaluation in a realm other than the top level realm, /, when handling policy decision requests from this policy agent.

This property is recognized by OpenAM, not the policy agent.

Property: `org.forgerock.openam.agents.config.policy.evaluation.realm`

Hot swap: yes

### Application

Application where OpenAM looks for policies to evaluate for this policy agent.

Default: `iPlanetAMWebAgentService`

Edit this property when OpenAM should look for policies that belong to an application other than `iPlanetAMWebAgentService` when handling policy decision requests from this policy agent.

This property is recognized by OpenAM, not the policy agent.

Property: `org.forgerock.openam.agents.config.policy.evaluation.application`

Hot swap: yes

## 5.5.5 Configuring Web Policy Agent Miscellaneous Properties

This section covers miscellaneous web agent properties. After creating the agent profile, you access these properties in the OpenAM console under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Miscellaneous.

### Advice Handling properties

Composite Advice Handling (Not yet in OpenAM console)

As of version 3.0.4, when set to true, the agent sends composite advice in the query (GET request) instead of sending it through a POST request.

Property: `com.sun.am.use_redirect_for_advice`

### Locale properties

Agent Locale

The default locale for the agent.

Property: `com.sun.identity.agents.config.locale`

Hot swap: no

### Anonymous user properties

Anonymous User

Enable or disable REMOTE\_USER processing for anonymous users.

Property: `com.sun.identity.agents.config.anonymous.user.enable`

### Cookie Processing properties

Encode special characters in Cookies

When enabled, use URL encoding for special characters in cookies. This is useful when profile, session, and response attributes contain special characters, and the attributes fetch mode is set to `HTTP_COOKIE`.

Property: com.sun.identity.agents.config.encode.cookie.special.chars.enable

Profile Attributes Cookie Prefix

Sets cookie prefix in the attributes headers. Default: HTTP\_.

Property: com.sun.identity.agents.config.profile.attribute.cookie.prefix

Profile Attributes Cookie Maxage

Maximum age in seconds of custom cookie headers. Default: 300.

Property: com.sun.identity.agents.config.profile.attribute.cookie.maxage

## URL Handling properties

URL Comparison Case Sensitivity Check

When enabled, enforces case insensitivity in both policy and not enforced URL evaluation.

Property: com.sun.identity.agents.config.url.comparison.case.ignore

Encode URL's Special Characters

When enabled, encodes the URL which has special characters before doing policy evaluation.

Property: com.sun.identity.agents.config.encode.url.special.chars.enable

## Ignore Naming URL properties

Ignore Preferred Naming URL in Naming Request

When enabled, do not send a preferred naming URL in the naming request.

Property: com.sun.identity.agents.config.ignore.preferred.naming.url

## Invalid URL properties (Not yet in OpenAM console)

Invalid URL Regular Expression

Use a [Perl-compatible regular expression](#) to filter out invalid request URLs. The policy agent reject requests to invalid URLs with HTTP 403 Forbidden status without further processing. Use Advanced > Custom Properties to set this in the agent profile.

For example, to filter out URLs containing the symbols in the list ., /, ., /, .., \, %00-%1f, %7f-%ff, %25, %2B, %2C, %7E, .info, use the following setting.

```
com.forgerock.agents.agent.invalid.url.regex= \
```

```
^((?!(|/\.|\.|/*|\.\.info)%25|%2B|%2C|[%0-1][0-9a-fA-F]|%[7-9a-fA-F][0-9a-fA-F])).)$
```

## Ignore Server Check properties

### Ignore Server Check

When enabled, do not check whether OpenAM is up before doing a 302 redirect.

Property: com.sun.identity.agents.config.ignore.server.check

## Ignore Path Info properties

### Ignore Path Info in Request URL

When enabled, strip path info from the request URL while doing the Not Enforced List check, and URL policy evaluation. This is designed to prevent a user from accessing a URI by appending the matching pattern in the policy or not enforced list.

### Note

This setting is not supported by the Varnish Cache agent.

For example, if the not enforced list includes `http://host/*.gif`, then stripping path info from the request URI prevents access to `http://host/index.html` by using `http://host/index.html?hack.gif`.

However, when a web server is configured as a reverse proxy for a J2EE application server, the path info is interpreted to map a resource on the proxy server rather than the application server. This prevents the not enforced list or the policy from being applied to the part of the URI below the application server path if a wildcard character is used.

For example, if the not enforced list includes `http://host/webapp/servcontext/*` and the request URL is `http://host/webapp/servcontext/example.jsp`, the path info is `/servcontext/example.jsp` and the resulting request URL with path info stripped is `http://host/webapp/`, which does not match the not enforced list. Thus when this property is enabled, path info is not stripped from the request URL even if there is a wildcard in the not enforced list or policy.

Make sure therefore when this property is enabled that there is nothing following the wildcard in the not enforced list or policy.

Property: com.sun.identity.agents.config.ignore.path.info

### **Multi-byte Enable properties**

#### Native Encoding of Profile Attributes

When enabled, the agent encodes the LDAP header values in the default encoding of operating system locale. When disabled, the agent uses UTF-8.

Property: `com.sun.identity.agents.config.convert.mbyte.enable`

### **Goto Parameter Name properties**

#### Goto Parameter Name

Property used only when CDSSO is enabled. Only change the default value, `goto` when the login URL has a landing page specified such as, `com.sun.identity.agents.config.cdsso.cdc servlet.url = http://openam.example.com:8080/openam/cdc servlet?goto= http://www.example.com/landing.jsp`. The agent uses this parameter to append the original request URL to this `cdcservlet` URL. The landing page consumes this parameter to redirect to the original URL.

As an example, if you set this value to `goto2`, then the complete URL sent for authentication is `http://openam.example.com:8080/openam/cdc servlet?goto= http://www.example.com/landing.jsp?goto2=http://www.example.com/original.jsp`.

Property: `com.sun.identity.agents.config.redirect.param`

### **Deprecated Agent properties**

#### Anonymous User Default Value

User ID of unauthenticated users. Default: `anonymous`.

Property: `com.sun.identity.agents.config.anonymous.user.id`

## **5.5.6**

### **Configuring Web Policy Agent Advanced Properties**

This section covers advanced web agent properties. After creating the agent profile, you access these properties in the OpenAM console under Access Control > *Realm Name* > Agents > Web > *Agent Name* > Advanced.

### **Client Identification properties**

If the agent is behind a proxy or load balancer, then the agent can get client IP and host name values from the proxy or load balancer. For proxies and load

balancer that support providing the client IP and host name in HTTP headers, you can use the following properties.

When multiple proxies or load balancers sit in the request path, the header values can include a comma-separated list of values with the first value representing the client, as in `client,next-proxy,first-proxy`.

**Client IP Address Header**

HTTP header name that holds the IP address of the client.

Property: `com.sun.identity.agents.config.client.ip.header`

**Client Hostname Header**

HTTP header name that holds the hostname of the client.

Property: `com.sun.identity.agents.config.client.hostname.header`

**Load Balancer properties**

**Load Balancer Setup**

Enable if a load balancer is used for OpenAM services.

Property: `com.sun.identity.agents.config.load.balancer.enable`

Hot swap: no

**Override Request URL Protocol**

Enable if the agent is sitting behind a SSL/TLS off-loader, load balancer, or proxy such that the protocol users use is different from the protocol the agent uses. When enabled, the protocol is overridden with the value from the Agent Deployment URI Prefix (property: `com.sun.identity.agents.config.agenturi.prefix`).

Property: `com.sun.identity.agents.config.override.protocol`

**Override Request URL Host**

Enable if the agent is sitting behind a SSL/TLS off-loader, load balancer, or proxy such that the host name users use is different from the host name the agent uses. When enabled, the host is overridden with the value from the Agent Deployment URI Prefix (property: `com.sun.identity.agents.config.agenturi.prefix`).

Property: `com.sun.identity.agents.config.override.host`

**Override Request URL Port**

Enable if the agent is sitting behind a SSL/TLS off-loader, load balancer, or proxy such that the port users use is different from the port the agent uses. When enabled, the port is overridden with the value from the Agent

Deployment URI Prefix (property: `com.sun.identity.agents.config.agenturi.prefix`).

Property: `com.sun.identity.agents.config.override.port`

#### Override Notification URL

Enable if the agent is sitting behind a SSL/TLS off-loader, load balancer, or proxy such that the URL users use is different from the URL the agent uses. When enabled, the URL is overridden with the value from the Agent Deployment URI Prefix (property: `com.sun.identity.agents.config.agenturi.prefix`).

Property: `com.sun.identity.agents.config.override.notification.url`

### Post Data Preservation properties

#### POST Data Preservation

Enables HTTP POST data preservation. This feature is available in the Apache 2.2, Microsoft IIS 6, Microsoft IIS 7, and Sun Java System Web Server web policy agents as of version 3.0.3.

Property: `com.sun.identity.agents.config.postdata.preserve.enable`

#### POST Data Entries Cache Period

POST cache entry lifetime in minutes. Default: 10.

Property: `com.sun.identity.agents.config.postcache.entry.lifetime`

#### POST Data Preservation Cookie Name (Not yet in OpenAM Console)

When HTTP POST data preservation is enabled, override properties are set to true, and the agent is behind a load balancer, then this property sets the name and value of the sticky cookie to use.

Property: `com.sun.identity.agents.config.postdata.preserve.lbcookie`

#### Post Data Preservation URI Prefix (Not yet in OpenAM Console)

If you run multiple web servers with policy agents behind a load balancer that directs traffic based on the request URI, and you need to preserve POST data, then set this property.

By default, policy agents use a dummy URL for POST data preservation, `http://agent.host:port/dummypost/sunpostpreserve`, to handle POST data across redirects to and from OpenAM. When you set this property, the policy agent prefixes the property value to the dummy URL path. In other words, when you set `com.forgerock.agents.config.pdpuri.prefix = app1`, the policy agent uses the dummy URL, `http://agent.host:port/app1/dummypost/sunpostpreserve`.

Next, use the prefix you set when you define load balancer URI rules. This ensures that clients end up being redirected to the policy agent that preserved the POST data.

Property: `com.forgerock.agents.config.pdpuri.prefix`

### **Sun Java System Proxy Server properties**

Override Proxy Server's Host and Port

When enabled ignore the host and port settings.

Property: `com.sun.identity.agents.config.proxy.override.host.port`

Hot swap: no

### **Microsoft IIS Server properties**

Authentication Type

The agent should normally perform authentication, so this is not required. If necessary, set to none.

Property: `com.sun.identity.agents.config.iis.auth.type`

Hot swap: no

Replay Password Key

DES key for decrypting the basic authentication password in the session.

Property: `com.sun.identity.agents.config.replaypasswd.key`

Filter Priority

The loading priority of filter, DEFAULT, HIGH, LOW, or MEDIUM.

Property: `com.sun.identity.agents.config.iis.filter.priority`

Filter configured with OWA

Enable if the IIS agent filter is configured for OWA.

Property: `com.sun.identity.agents.config.iis.owa.enable`

Change URL Protocol to https

Enable to avoid IE6 security pop-ups.

Property: `com.sun.identity.agents.config.iis.owa.enable.change.protocol`

Idle Session Timeout Page URL

URL of the local idle session timeout page.

Property: com.sun.identity.agents.config.iis.owa.enable.session.timeout.url

### IBM Lotus Domino Server properties

#### Check User in Domino Database

When enabled, the agent checks whether the user exists in the Domino name database.

Property: com.sun.identity.agents.config.domino.check.name.database

#### Use LTPA token

Enable if the agent needs to use LTPA Token.

Property: com.sun.identity.agents.config.domino.ltpa.enable

#### LTPA Token Cookie Name

The name of the cookie that contains the LTPA token.

Property: com.sun.identity.agents.config.domino.ltpa.cookie.name

#### LTPA Token Configuration Name

The configuration name that the agent uses in order to employ the LTPA token mechanism.

Property: com.sun.identity.agents.config.domino.ltpa.config.name

#### LTPA Token Organization Name

The organization name to which the LTPA token belongs.

Property: com.sun.identity.agents.config.domino.ltpa.org.name

### Custom properties

#### Custom Properties

Additional properties to augment the set of properties supported by agent. Such properties take the following forms.

- customproperty=custom-value1
- customlist[0]=customlist-value-0
- customlist[1]=customlist-value-1
- custommap[key1]=custommap-value-1
- custommap[key2]=custommap-value-2

Property: `com.sun.identity.agents.config.freeformproperties`

## 5.6 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 OpenAM console. Alternatively, you can choose to store the agent configuration locally and configure the agent by changing values in the properties file. This section covers centralized configuration, indicating the corresponding properties for use in a local configuration file where applicable.<sup>2</sup>

### Tip

To show the agent properties in configuration file format that correspond to what you see in the console, click Export Configuration after editing agent properties.

After changing properties specified as "Hot swap: no" you must restart the application server or web server, or the agent's container.

### 5.6.1 Configuring Java EE Policy Agent Global Properties

This section covers global Java EE agent properties. After creating the agent profile, you access these properties in the OpenAM console under Access Control > *Realm Name* > Agents > Java EE > *Agent Name* > Global.

#### Profile properties

##### Group

For assigning the agent to a previously configured Java EE agent group in order to inherit selected properties from the group.

##### Password

Agent password used when creating the password file and when installing the agent.

##### Status

Status of the agent configuration.

##### Agent Notification URL

URL used by agent to register notification listeners.

---

<sup>2</sup>The configuration file syntax is that of a standard Java properties file. See [java.util.Properties.load\(\)](#) for a description of the format. The value of a property specified multiple times is not defined.

Property: `com.sun.identity.client.notification.url`

Hot swap: no

Location of Agent Configuration Repository

Indicates agent's configuration located either on agent's host or centrally on OpenAM server.

If you change this to a local configuration, you can no longer manage the policy agent configuration through OpenAM console.

Property: `com.sun.identity.agents.config.repository.location`

Configuration Reload Interval

Interval in seconds to fetch agent configuration from OpenAM. Used if notifications are disabled. Default: 0

Property: `com.sun.identity.agents.config.load.interval`

Agent Configuration Change Notification

Enable agent to receive notification messages from OpenAM server for configuration changes.

Property: `com.sun.identity.agents.config.change.notification.enable`

Agent Root URL for CDSSO

The agent root URL 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.

## General properties

Agent Filter Mode

Specifies how the agent filters requests to protected web applications. The global value functions as a default, and applies for protected applications that do not have their own filter settings. Valid settings include the following.

ALL

Enforce both the Java EE policy defined for the web container where the protected application runs, and also OpenAM policies.

When setting the filter mode to ALL, set the Map Key, but do not set any Corresponding Map Value.

**J2EE\_POLICY**

Enforce only the J2EE policy defined for the web container where the protected application runs.

**NONE**

Do not enforce policies to protect resources. In other words, turn off access management. Not for use in production.

**SSO\_ONLY**

Enforce only authentication, not policies.

**URL\_POLICY**

Enforce only URL resource-based policies defined in OpenAM.

When setting the filter mode to URL\_POLICY, set the Map Key to the application name and the Corresponding Map Value to URL\_POLICY.

Property: `com.sun.identity.agents.config.filter.mode`

Hot swap: no

**HTTP Session Binding**

When enabled the agent invalidates the HTTP session upon login failure, when the user has no SSO session, or when the principal user name does not match the SSO user name.

Property: `com.sun.identity.agents.config.httpsession.binding`

**Login Attempt Limit**

When set to a value other than zero, this defines the maximum number of failed login attempts allowed during a single browser session, after which the agent blocks requests from the user.

Property: `com.sun.identity.agents.config.login.attempt.limit`

**Custom Response Header**

Specifies the custom headers the agent sets for the client. The key is the header name. The value is the header value.

Property: `com.sun.identity.agents.config.response.header`

For example, `com.sun.identity.agents.config.response.header[Cache-Control]=no-cache`.

**Redirect Attempt Limit**

When set to a value other than zero, this defines the maximum number of redirects allowed for a single browser session, after which the agent blocks the request.

Property: `com.sun.identity.agents.config.redirect.attempt.limit`

**Agent Debug Level**

Default is Error. Increase to Message for fine-grained detail.

Property: `com.iplanet.services.debug.level`

## **User Mapping properties**

**User Mapping Mode**

Specifies the mechanism used to determine the user ID.

Property: `com.sun.identity.agents.config.user.mapping.mode`

**User Attribute Name**

Specifies the data store attribute that contains the user ID.

Property: `com.sun.identity.agents.config.user.attribute.name`

**User Principal Flag**

When enabled, OpenAM uses both the principal user name and also the user ID for authentication.

Property: `com.sun.identity.agents.config.user.principal`

**User Token Name**

Specifies the session property name for the authenticated user's ID. Default: `UserToken`.

Property: `com.sun.identity.agents.config.user.token`

## **Audit properties**

**Audit Access Types**

Types of messages to log based on user URL access attempts.

Property: `com.sun.identity.agents.config.audit.accessType`

Valid values for the configuration file property include `LOG_NONE`, `LOG_ALLOW`, `LOG_DENY`, and `LOG_BOTH`.

**Audit Log Location**

Specifies where audit messages are logged. By default, audit messages are logged remotely.

Property: `com.sun.identity.agents.config.log.disposition`

Valid values for the configuration file property include `REMOTE`, `LOCAL`, and `ALL`.

**Remote Log Filename**

Name of file stored on OpenAM server that contains agent audit messages if log location is remote or all.

Property: `com.sun.identity.agents.config.remote.logfile`

Hot swap: no

**Rotate Local Audit Log**

When enabled, audit log files are rotated when reaching the specified size.

Property: `com.sun.identity.agents.config.local.log.rotate`

**Local Audit Log Rotation Size**

Beyond this size limit in bytes the agent rotates the local audit log file if rotation is enabled.

Property: `com.sun.identity.agents.config.local.log.size`

Default: 50 MB

## Fully Qualified Domain Name Checking properties

**FQDN Check**

Enables checking of FQDN default value and FQDN map values.

Property: `com.sun.identity.agents.config.fqdn.check.enable`

**FQDN Default**

Fully qualified domain name that the users should use in order to access resources.

This property ensures that when users access protected resources on the web server without specifying the FQDN, the agent can redirect the users to URLs containing the correct FQDN.

Property: `com.sun.identity.agents.config.fqdn.default`

**FQDN Virtual Host Map**

Enables virtual hosts, partial hostname and IP address to access protected resources. Maps invalid or virtual name keys to valid FQDN values so the agent can properly redirect users and the agents receive cookies belonging to the domain.

To map `myserver` to `myserver.mydomain.example`, enter `myserver` in the Map Key field, and enter `myserver.mydomain.example` in the Corresponding Map Value field. This corresponds to `com.sun.identity.agents.config.fqdn.mapping[myserver]= myserver.mydomain.example`.

Property: `com.sun.identity.agents.config.fqdn.mapping`

## 5.6.2 Configuring Java EE policy agent Application Properties

This section covers application J2EE agent properties. After creating the agent profile, you access these properties in the OpenAM console under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Application.

### Login Processing properties

#### Login Form URI

Specifies the list of absolute URIs corresponding to a protected application's `web.xml` `form-login-page` element, such as `/myApp/jsp/login.jsp`.

Property: `com.sun.identity.agents.config.login.form`

#### Login Error URI

Specifies the list of absolute URIs corresponding to a protected application's `web.xml` `form-error-page` element, such as `/myApp/jsp/error.jsp`.

Property: `com.sun.identity.agents.config.login.error.uri`

#### Use Internal Login

When enabled, the agent uses the internal default content file for the login.

Property: `com.sun.identity.agents.config.login.use.internal`

#### Login Content File Name

Full path name to the file containing custom login content when Use Internal Login is enabled.

Property: `com.sun.identity.agents.config.login.content.file`

### Logout Processing properties

#### Application Logout Handler

Specifies how logout handlers map to specific applications. The key is the web application name. The value is the logout handler class.

To set a global logout handler for applications without other logout handlers defined, leave the key empty and set the value to the global logout handler class name, `GlobalApplicationLogoutHandler`.

To set a logout handler for a specific application, set the key to the name of the application, and the value to the logout handler class name.

Property: `com.sun.identity.agents.config.logout.application.handler`

**Application Logout URI**

Specifies request URIs that indicate logout events. The key is the web application name. The value is the application logout URI.

To set a global logout URI for applications without other logout URIs defined, leave the key empty and set the value to the global logout URI, `/logout.jsp`.

To set a logout URI for a specific application, set the key to the name of the application, and the value to the application logout page.

Property: `com.sun.identity.agents.config.logout.uri`

**Logout Request Parameter**

Specifies parameters in the HTTP request that indicate logout events. The key is the web application name. The value is the logout request parameter.

To set a global logout request parameter for applications without other logout request parameters defined, leave the key empty and set the value to the global logout request parameter, `logoutparam`.

To set a logout request parameter for a specific application, set the key to the name of the application, and the value to the application logout request parameter, such as `logoutparam`.

Property: `com.sun.identity.agents.config.logout.request.param`

**Logout Introspect Enabled**

When enabled, the agent checks the HTTP request body to locate the Logout Request Parameter you set.

Property: `com.sun.identity.agents.config.logout.introspect.enabled`

**Logout Entry URI**

Specifies the URIs to return after successful logout and subsequent authentication. The key is the web application name. The value is the URI to return.

To set a global logout entry URI for applications without other logout entry URIs defined, leave the key empty and set the value to the global logout entry URI, `/welcome.html`.

To set a logout entry URI for a specific application, set the key to the name of the application, and the value to the application logout entry URI, such as `/myApp/welcome.html`.

Property: `com.sun.identity.agents.config.logout.entry.uri`

## Access Denied URI Processing properties

### Resource Access Denied URI

Specifies the URIs of custom pages to return when access is denied. The key is the web application name. The value is the custom URI.

To set a global custom access denied URI for applications without other custom access denied URIs defined, leave the key empty and set the value to the global custom access denied URI, /sample/accessdenied.html.

To set a custom access denied URI for a specific application, set the key to the name of the application, and the value to the application access denied URI, such as /myApp/accessdenied.html.

Property: com.sun.identity.agents.config.access.denied.uri

## Not Enforced URI Processing properties

### Not Enforced URIs

List of URIs for which no authentication is required, and the agent does not protect access. You can use wildcards to define a pattern for a URI.

The \* wildcard matches all characters except question mark (?), cannot be escaped, and spans multiple levels in a URI. Multiple forward slashes do not match a single forward slash, so \* matches mult/iple/dirs, yet mult/\*/dirs does not match mult/dirs.

The -\*- wildcard matches all characters except forward slash (/) or question mark (?), and cannot be escaped. As it does not match /, -\*- does not span multiple levels in a URI.

OpenAM does not let you mix \* and -\*- in the same URI.

Examples include /logout.html, /images/\*, /css/-\*-, and /\*.jsp?locale=\*.

Trailing forward slashes are not recognized as part of a resource name. Therefore /images// and /images are equivalent.

Property: com.sun.identity.agents.config.notenforced.uri

### Invert Not Enforced URIs

Only enforce not enforced list of URIs. In other words, enforce policy only for those URIs and patterns specified in the list.

Property: com.sun.identity.agents.config.notenforced.uri.invert

**Not Enforced URIs Cache Enabled**

When enabled, the agent caches evaluation of the not enforced URI list.

Property: `com.sun.identity.agents.config.notenforced.uri.cache.enable`

**Not Enforced URIs Cache Size**

When caching is enabled, this limits the number of not enforced URIs cached.

Property: `com.sun.identity.agents.config.notenforced.uri.cache.size`

Default: 1000

**Refresh Session Idle Time**

When enabled, the agent reset the session idle time when granting access to a not enforced URI, prolonging the time before the user must authenticate again.

Property: `com.sun.identity.agents.config.notenforced.refresh.session.idletime`

## **Not Enforced IP Processing properties**

**Not Enforced Client IP List**

No authentication and authorization are required for the requests coming from these client IP addresses.

Property: `com.sun.identity.agents.config.notenforced.ip`

### **Note**

Loopback addresses are not considered valid IPs on the Not Enforced IP list. If specified, the policy agent ignores the loopback address.

**Not Enforced IP Invert List**

Only enforce the not enforced list of IP addresses. In other words, enforce policy only for those client addresses and patterns specified in the list.

Property: `com.sun.identity.agents.config.notenforced.ip.invert`

**Not Enforced IP Cache Flag**

When enabled, the agent caches evaluation of the not enforced IP list.

Property: `com.sun.identity.agents.config.notenforced.ip.cache.enable`

**Not Enforced IP Cache Size**

When caching is enabled, this limits the number of not enforced addresses cached.

Property: `com.sun.identity.agents.config.notenforced.ip.cache.size`

Default: 1000

## Profile Attributes Processing properties

### Profile Attribute Fetch Mode

When set to `HTTP_COOKIE` or `HTTP_HEADER`, profile attributes are introduced into the cookie or the headers, respectively. When set to `REQUEST_ATTRIBUTE`, profile attributes are part of the HTTP request.

Property: `com.sun.identity.agents.config.profile.attribute.fetch.mode`

### Profile Attribute Map

Maps the profile attributes to HTTP headers for the currently authenticated user. Map Keys are attribute names, and Map Values are HTTP header names. The user profile can be stored in LDAP or any other arbitrary data store.

To populate the value of profile attribute CN under `CUSTOM-Common-Name`: enter CN in the Map Key field, and enter `CUSTOM-Common-Name` in the Corresponding Map Value field. This corresponds to `com.sun.identity.agents.config.profile.attribute.mapping[cn]=CUSTOM-Common-Name`.

In most cases, in a destination application where an HTTP header name shows up as a request header, it is prefixed by `HTTP_`, lower case letters become upper case, and hyphens (-) become underscores (\_). For example, `common-name` becomes `HTTP_COMMON_NAME`.

Property: `com.sun.identity.agents.config.profile.attribute.mapping`

## Response Attributes Processing properties

### Response Attribute Fetch Mode

When set to `HTTP_COOKIE` or `HTTP_HEADER`, response attributes are introduced into the cookie or the headers, respectively. When set to `REQUEST_ATTRIBUTE`, response attributes are part of the HTTP request.

Property: `com.sun.identity.agents.config.response.attribute.fetch.mode`

### Response Attribute Map

Maps the policy response attributes to HTTP headers for the currently authenticated user. The response attribute is the attribute in the policy response to be fetched.

To populate the value of response attribute `uid` under `CUSTOM-User-Name`: enter `uid` in the Map Key field, and enter `CUSTOM-User-Name` in the Corresponding

Map Value field. This corresponds to `com.sun.identity.agents.config.response.attribute.mapping[uid]=Custom-User-Name`.

In most cases, in a destination application where an HTTP header name shows up as a request header, it is prefixed by `HTTP_`, lower case letters become upper case, and hyphens (-) become underscores (\_). For example, `response-attr-one` becomes `HTTP_RESPONSE_ATTR_ONE`.

Property: `com.sun.identity.agents.config.response.attribute.mapping`

### **Common Attributes Fetching Processing properties**

#### **Cookie Separator Character**

Specifies the separator for multiple values of the same attribute when it is set as a cookie. Default: | (also known as the vertical bar character).

Property: `com.sun.identity.agents.config.attribute.cookie.separator`

#### **Fetch Attribute Date Format**

Specifies the `java.text.SimpleDateFormat` of date attribute values used when an attribute is set in an HTTP header. Default: EEE, d MMM yyyy hh:mm:ss z.

Property: `com.sun.identity.agents.config.attribute.date.format`

#### **Attribute Cookie Encode**

When enabled, attribute values are URL encoded before being set as a cookie.

Property: `com.sun.identity.agents.config.attribute.cookie.encode`

### **Session Attributes Processing properties**

#### **Session Attribute Fetch Mode**

When set to `HTTP_COOKIE` or `HTTP_HEADER`, session attributes are introduced into the cookie or the headers, respectively. When set to `REQUEST_ATTRIBUTE`, session attributes are part of the HTTP request.

Property: `com.sun.identity.agents.config.session.attribute.fetch.mode`

#### **Session Attribute Map**

Maps session attributes to HTTP headers for the currently authenticated user. The session attribute is the attribute in the session to be fetched.

To populate the value of session attribute `UserToken` under `CUSTOM-userid`: enter `UserToken` in the Map Key field, and enter `CUSTOM-userid` in the

Corresponding Map Value field. This corresponds to `com.sun.identity.agents.config.session.attribute.mapping[UserToken]=CUSTOM-userid`.

In most cases, in a destination application where an HTTP header name shows up as a request header, it is prefixed by `HTTP_`, lower case letters become upper case, and hyphens (-) become underscores (\_). For example, `success-url` becomes `HTTP_SUCCESS_URL`.

Property: `com.sun.identity.agents.config.session.attribute.mapping`

### **Privilege Attributes Processing properties**

Privileged attributes are used when the agent is running in ALL or J2EE\_POLICY filter mode. Privileged attributes contain the list of declarative Java EE roles that the user can have.

#### **Default Privileged Attribute**

Specifies that every authenticated user with a valid OpenAM session will have the AUTHENTICATED\_USERS role.

Property: `com.sun.identity.agents.config.default.privileged.attribute`

#### **Privileged Attribute Type**

Specifies the group and role memberships that will be turned into roles for each user.

Property: `com.sun.identity.agents.config.privileged.attribute.type`

#### **Privileged Attributes To Lower Case**

Specifies how privileged attribute types should be converted to lower case.

Property: `com.sun.identity.agents.config.privileged.attribute.tolowercase`

#### **Privileged Session Attribute**

Specifies the list of session property names when an authenticated user's roles are stored within a session property.

Property: `com.sun.identity.agents.config.privileged.session.attribute`

#### **Enable Privileged Attribute Mapping**

When enabled, lets you use Privileged Attribute Mapping.

Property: `com.sun.identity.agents.config.privileged.attribute.mapping.enable`

#### **Privileged Attribute Mapping**

OpenAM allows original attribute values to be mapped to other values. For example, you can map UUIDs to principal names in roles specified in

a web application's deployment descriptor. For example, to map the UUID `id=employee,ou=group,o=openam` to the principal name `am_employee_role` in the deployment descriptor, set the key to `id=employee,ou=group,o=openam`, and the value to `am_employee_role`.

Property: `com.sun.identity.agents.config.privileged.attribute.mapping`

### **Custom Authentication Processing properties**

#### **Custom Authentication Handler**

Specifies custom authentication handler classes for users authenticated with the application server. The key is the web application name and the value is the authentication handler class name.

Property: `com.sun.identity.agents.config.auth.handler`

#### **Custom Logout Handler**

Specifies custom logout handler classes to log users out of the application server. The key is the web application name and the value is the logout handler class name.

Property: `com.sun.identity.agents.config.logout.handler`

#### **Custom Verification Handler**

Specifies custom verification classes to validate user credentials with the local user repository. The key is the web application name and the value is the validation handler class name.

Property: `com.sun.identity.agents.config.verification.handler`

## **5.6.3 Configuring Java EE policy agent SSO Properties**

This section covers SSO J2EE agent properties. After creating the agent profile, you access these properties in the OpenAM console under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > SSO

### **Cookie properties**

#### **Cookie Name**

Name of the SSO Token cookie used between the OpenAM server and the agent. Default: `iPlanetDirectoryPro`.

Property: `com.iplanet.am.cookie.name`

Hot swap: no

## Caching properties

### SSO Cache Enable

When enabled, the agent exposes SSO Cache through the agent SDK APIs.

Property: `com.sun.identity.agents.config.amsso.cache.enable`

## Cross Domain SSO properties

### Cross Domain SSO

Enables Cross Domain Single Sign On.

Property: `com.sun.identity.agents.config.cdsso.enable`

### CDSSO Redirect URI

Specifies a URI the agent uses to process CDSSO requests.

Property: `com.sun.identity.agents.config.cdsso.redirect.uri`

### CDSSO Servlet URL

List of URLs of the available CDSSO controllers that the agent can use for CDSSO processing. For example, `http://openam.example.com:8080/openam/cdc servlet`.

Property: `com.sun.identity.agents.config.cdsso.cdc servlet.url`

### CDSSO Clock Skew

When set to a value other than zero, specifies the clock skew in seconds that the agent accepts when determining the validity of the CDSSO authentication response assertion.

Property: `com.sun.identity.agents.config.cdsso.clock.skew`

### CDSSO Trusted ID Provider

Specifies the list of OpenAM servers or identity providers the agent trusts when evaluating CDC Liberty Responses.

Property: `com.sun.identity.agents.config.cdsso.trusted.id.provider`

### CDSSO Secure Enable

When enabled, the agent marks the SSO Token cookie as secure, thus the cookie is only transmitted over secure connections.

Property: `com.sun.identity.agents.config.cdsso.secure.enable`

### CDSSO Domain List

List of domains, such as `.example.com`, in which cookies have to be set in CDSSO.

## Configuring Java EE policy agent OpenAM Services Properties

---

Property: `com.sun.identity.agents.config.cdsso.domain`

### Cookie Reset properties

#### Cookie Reset

When enabled, agent resets cookies in the response before redirecting to authentication.

Property: `com.sun.identity.agents.config.cookie.reset.enable`

#### Cookie Reset Name List

List of cookies to reset if Cookie Reset is enabled.

Property: `com.sun.identity.agents.config.cookie.reset.name`

#### Cookie Reset Domain Map

Specifies how names from the Cookie Reset Name List correspond to cookie domain values when the cookie is reset.

Property: `com.sun.identity.agents.config.cookie.reset.domain`

#### Cookie Reset Path Map

Specifies how names from the Cookie Reset Name List correspond to cookie paths when the cookie is reset.

Property: `com.sun.identity.agents.config.cookie.reset.path`

## 5.6.4 Configuring Java EE policy agent OpenAM Services Properties

This section covers OpenAM services J2EE agent properties. After creating the agent profile, you access these properties in the OpenAM console under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > OpenAM Services.

### Login URL properties

#### OpenAM Login URL

OpenAM login page URL, such as `http://openam.example.com:8080/openam/UI/Login`, to which the agent redirects incoming users without sufficient credentials so that they can authenticate. If CDSSO is enabled, this property is not used, instead the CDCServlet URL will be used.

Property: `com.sun.identity.agents.config.login.url`

#### OpenAM Conditional Login URL (Not yet in OpenAM console)

To conditionally redirect users based on the incoming request URL, set this property.

## Configuring Java EE policy agent OpenAM Services Properties

---

This takes the incoming request domain to match, a vertical bar ( | ), and then a comma-separated list of URLs to which to redirect incoming users.

If the domain before the vertical bar matches an incoming request URL, then the policy agent uses the list of URLs to determine how to redirect the user-agent. If the global property FQDN Check (com.sun.identity.agents.config.fqdn.check.enable) is enabled for the policy agent, then the policy agent iterates through the list until it finds an appropriate redirect URL that matches the FQDN check. Otherwise, the policy agent redirects the user-agent to the first URL in the list.

Property: com.sun.identity.agents.config.conditional.login.url

Examples: com.sun.identity.agents.config.conditional.login.url[0]= login.example.com|http://openam1.example.com/openam/UI/Login, http://openam2.example.com/openam/UI/Login, com.sun.identity.agents.config.conditional.login.url[1]= signin.example.com|http://openam3.example.com/openam/UI/Login, http://openam4.example.com/openam/UI/Login

If CDSSO is enabled for the policy agent, then this property takes CDSSO Servlet URLs for its values (com.sun.identity.agents.config.cdssocdcsvr.url), rather than OpenAM login URLs.

CDSSO examples: com.sun.identity.agents.config.conditional.login.url[0]= login.example.com|http://openam1.example.com/openam/cdcsvr, http://openam2.example.com/openam/cdcsvr, com.sun.identity.agents.config.conditional.login.url[1]= signin.example.com|http://openam3.example.com/openam/cdcsvr, http://openam4.example.com/openam/cdcsvr

### Login URL Prioritized

When enabled, OpenAM uses the priority defined in the OpenAM Login URL list as the priority for Login and CDSSO URLs when handling failover.

Property: com.sun.identity.agents.config.login.url.prioritized

### Login URL Probe

When enabled, OpenAM checks the availability of OpenAM Login URLs before redirecting to them.

Property: com.sun.identity.agents.config.login.url.probe.enabled

### Login URL Probe Timeout

Timeout period in milliseconds for OpenAM to determine whether to failover between Login URLs when Login URL Probe is enabled.

Property: com.sun.identity.agents.config.login.url.probe.timeout

Default: 2000

## Logout URL properties

### OpenAM Logout URL

OpenAM logout page URLs, such as `http://openam.example.com:8080/openam/UI/Logout`. The user is logged out of the OpenAM session when accessing these URLs.

Property: `com.sun.identity.agents.config.logout.url`

### OpenAM Conditional Logout URL (Not yet in OpenAM console)

The values take the incoming request URL to match and a comma-separated list of URLs to which to redirect users logging out.

Property: `com.sun.identity.agents.config.conditional.logout.url`

Example: `com.sun.identity.agents.config.conditional.logout.url[0]=logout.example.com|http://openam1.example.com/openam/UI/Logout, http://openam2.example.com/openam/UI/Logout`

### Logout URL Prioritized

When enabled, OpenAM uses the priority defined in the OpenAM Logout URL list as the priority for Logout URLs when handling failover.

Property: `com.sun.identity.agents.config.logout.url.prioritized`

### Logout URL Probe

When enabled, OpenAM checks the availability of OpenAM Logout URLs before redirecting to them.

Property: `com.sun.identity.agents.config.logout.url.probe.enabled`

### Logout URL Probe Timeout

Timeout period in milliseconds for OpenAM to determine whether to failover between Logout URLs when Logout URL Probe is enabled.

Property: `com.sun.identity.agents.config.logout.url.probe.timeout`

Default: 2000

## Authentication Service properties

### OpenAM Authentication Service Protocol

Specifies the protocol used by the OpenAM authentication service.

Property: `com.iplanet.am.server.protocol`

## Configuring Java EE policy agent OpenAM Services Properties

---

Hot swap: no

**OpenAM Authentication Service Host Name**  
Specifies the OpenAM authentication service host name.

Property: `com.iplanet.am.server.host`

Hot swap: no

**OpenAM Authentication Service Port**  
Specifies the OpenAM authentication service port number.

Property: `com.iplanet.am.server.port`

Hot swap: no

### **Policy Client Service properties**

**Realm**

Realm where OpenAM starts policy evaluation for this policy agent.

Default: `/` (top level realm)

Edit this property when OpenAM should start policy evaluation in a realm other than the top level realm, `/`, when handling policy decision requests from this policy agent.

This property is recognized by OpenAM, not the policy agent.

Property: `org.forgerock.openam.agents.config.policy.evaluation.realm`

Hot swap: yes

**Application**

Application where OpenAM looks for policies to evaluate for this policy agent.

Default: `iPlanetAMWebAgentService`

Edit this property when OpenAM should look for policies that belong to an application other than `iPlanetAMWebAgentService` when handling policy decision requests from this policy agent.

This property is recognized by OpenAM, not the policy agent.

Property: `org.forgerock.openam.agents.config.policy.evaluation.application`

Hot swap: yes

## Configuring Java EE policy agent OpenAM Services Properties

---

### Enable Policy Notifications

When enabled, OpenAM sends notification about changes to policy.

Property: `com.sun.identity.agents.notification.enabled`

Hot swap: no

### Policy Client Polling Interval

Specifies the time in minutes after which the policy cache is refreshed.

Property: `com.sun.identity.agents.polling.interval`

Default: 3

Hot swap: no

### Policy Client Cache Mode

Set to cache mode subtree when only a small number of policy rules are defined. For large numbers of policy rules, set to self.

Property: `com.sun.identity.policy.client.cacheMode`

Default: self

Hot swap: no

### Policy Client Boolean Action Values

Specifies the values, such as allow and deny, that are associated with boolean policy decisions. The string is presented below in multiple lines for readability purposes.

Default: iPlanetAMWebAgentService|GET|allow|deny:  
iPlanetAMWebAgentService|POST|allow|deny: iPlanetAMWebAgentService|  
PUT|allow|deny: iPlanetAMWebAgentService|DELETE|allow|deny:  
iPlanetAMWebAgentService|HEAD|allow|deny: iPlanetAMWebAgentService|  
OPTIONS|allow|deny: iPlanetAMWebAgentService|PATCH|allow|deny

Property: `com.sun.identity.policy.client.booleanActionValues`

Hot swap: no

### Policy Client Resource Comparators

Specifies the comparators used for service names in policy.

Default: `serviceType=iPlanetAMWebAgentService| class=com.sun.identity.policy.plugins.HttpURLResourceName|wildcard=*&| delimiter=/& caseSensitive=false`

Property: `com.sun.identity.policy.client.resourceComparators`

Hot swap: no

**Policy Client Clock Skew**

Time in seconds used to adjust time difference between agent system and OpenAM. Clock skew in seconds = AgentTime - OpenAMServerTime.

Default: 10.

Property: `com.sun.identity.policy.client.clockSkew`

Hot swap: no

**URL Policy Env GET Parameters**

Specifies the list of HTTP GET request parameters whose names and values the agents sets in the environment map for URL policy evaluation by the OpenAM server.

Property: `com.sun.identity.agents.config.policy.env.get.param`

**URL Policy Env POST Parameters**

Specifies the list of HTTP POST request parameters whose names and values the agents sets in the environment map for URL policy evaluation by the OpenAM server.

Property: `com.sun.identity.agents.config.policy.env.post.param`

**URL Policy Env jsession Parameters**

Specifies the list of HTTP session attributes whose names and values the agents sets in the environment map for URL policy evaluation by the OpenAM server.

Property: `com.sun.identity.agents.config.policy.env.jsession.param`

**Use HTTP-Redirect for composite advice**

When enabled, the remote policy client is configured to use HTTP-Redirect instead of HTTP-POST for composite advice.

Property: `com.sun.identity.agents.config.policy.advice.use.redirect`

**User Data Cache Service properties**

**Enable Notification of User Data Caches**

When enabled, receive notification from OpenAM to update user management data caches.

Property: `com.sun.identity.idm.remote.notification.enabled`

Hot swap: no

## Configuring Java EE policy agent OpenAM Services Properties

---

### User Data Cache Polling Time

If notifications are not enabled and set to a value other than zero, specifies the time in minutes after which the agent polls to update cached user management data.

Property: `com.iplanet.am.sdk.remote.pollingTime`

Default: 1

Hot swap: no

### Enable Notification of Service Data Caches

When enabled, receive notification from OpenAM to update service configuration data caches.

Property: `com.sun.identity.sm.notification.enabled`

Hot swap: no

### Service Data Cache Time

If notifications are not enabled and set to a value other than zero, specifies the time in minutes after which the agent polls to update cached service configuration data.

Property: `com.sun.identity.sm.cacheTime`

Default: 1

Hot swap: no

## **Session Client Service properties**

### Enable Client Polling

When enabled, the session client polls to update the session cache rather than relying on notifications from OpenAM.

Property: `com.iplanet.am.session.client.polling.enable`

Hot swap: no

### Client Polling Period

Specifies the time in seconds after which the session client requests an update from OpenAM for cached session information.

Property: `com.iplanet.am.session.client.polling.period`

Default: 180

Hot swap: no

## 5.6.5 Configuring Java EE policy agent Miscellaneous Properties

This section covers miscellaneous J2EE agent properties. After creating the agent profile, you access these properties in the OpenAM console under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Miscellaneous.

### Locale properties

#### Locale Language

The default language for the agent.

Property: `com.sun.identity.agents.config.locale.language`

Hot swap: no

#### Locale Country

The default country for the agent.

Property: `com.sun.identity.agents.config.locale.country`

Hot swap: no

### Port Check Processing properties

#### Port Check Enable

When enabled, activate port checking, correcting requests on the wrong port.

Property: `com.sun.identity.agents.config.port.check.enable`

#### Port Check File

Specifies the name of the file containing the content to handle requests on the wrong port when port checking is enabled.

Property: `com.sun.identity.agents.config.port.check.file`

#### Port Check Setting

Specifies which ports correspond to which protocols. The agent uses the map when handling requests with invalid port numbers during port checking.

Property: `com.sun.identity.agents.config.port.check.setting`

### Bypass Principal List properties

#### Bypass Principal List

Specifies a list of principals the agent bypasses for authentication and search purposes, such as guest or testuser.

Property: com.sun.identity.agents.config.bypass.principal

### **Agent Password Encryptor properties**

#### Encryption Provider

Specifies the agent's encryption provider class.

Default: com.iplanet.services.util.JCEEEncryption

Property: com.iplanet.security.encryptor

Hot swap: no

### **Ignore Path Info properties**

#### Ignore Path Info in Request URL

When enabled, strip path info from the request URL while doing the Not Enforced List check, and URL policy evaluation. This is designed to prevent a user from accessing a URI by appending the matching pattern in the policy or not enforced list.

For example, if the not enforced list includes `/*.gif`, then stripping path info from the request URL prevents access to `http://host/index.html` by using `http://host/index.html?hack.gif`.

Property: com.sun.identity.agents.config.ignore.path.info

### **Deprecated Client Browser User Agent Properties**

#### Goto Parameter Name

Property used only when CDSSO is enabled. Only change the default value, `goto` when the login URL has a landing page specified such as, `com.sun.identity.agents.config.cdsso.cdc servlet.url = http://openam.example.com:8080/openam/cdc servlet?goto= http://www.example.com/landing.jsp`. The agent uses this parameter to append the original request URL to this `c dc servlet` URL. The landing page consumes this parameter to redirect to the original URL.

As an example, if you set this value to `goto2`, then the complete URL sent for authentication is `http://openam.example.com:8080/openam/cdc servlet?goto= http://www.example.com/landing.jsp?goto2=http://www.example.com/original.jsp`.

Property: com.sun.identity.agents.config.redirect.param

#### Legacy User Agent Support Enable

When enabled, provide support for legacy browsers.

Property: `com.sun.identity.agents.config.legacy.support.enable`

**Legacy User Agent List**

List of header values that identify legacy browsers. Entries can use the wildcard character, \*.

Property: `com.sun.identity.agents.config.legacy.user.agent`

**Legacy User Agent Redirect URI**

Specifies a URI the agent uses to redirect legacy user agent requests.

Property: `com.sun.identity.agents.config.legacy.redirect.uri`

## 5.6.6 Configuring Java EE policy agent Advanced Properties

This section covers advanced J2EE agent properties. After creating the agent profile, you access these properties in the OpenAM console under Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Advanced.

### Client Identification properties

If the agent is behind a proxy or load balancer, then the agent can get client IP and host name values from the proxy or load balancer. For proxies and load balancer that support providing the client IP and host name in HTTP headers, you can use the following properties.

When multiple proxies or load balancers sit in the request path, the header values can include a comma-separated list of values with the first value representing the client, as in `client,next-proxy,first-proxy`.

**Client IP Address Header**

HTTP header name that holds the IP address of the client.

Property: `com.sun.identity.agents.config.client.ip.header`

**Client Hostname Header**

HTTP header name that holds the hostname of the client.

Property: `com.sun.identity.agents.config.client.hostname.header`

### Web Service Processing properties

**Web Service Enable**

Enable web service processing.

Property: `com.sun.identity.agents.config.webservice.enable`

**Web Service End Points**

Specifies a list of web application end points that represent web services.

Property: `com.sun.identity.agents.config.webservice.endpoint`

**Web Service Process GET Enable**

When enabled, the agent processes HTTP GET requests for web service endpoints.

Property: `com.sun.identity.agents.config.webservice.process.get.enable`

**Web Service Authenticator**

Specifies a class implementing `com.sun.identity.agents.filter.IWebServiceAuthenticator`, used to authenticate web service requests.

Property: `com.sun.identity.agents.config.webservice.authenticator`

**Web Service Response Processor**

Specifies a class implementing `com.sun.identity.agents.filter.IWebServiceResponseProcessor`, used to process web service responses.

Property: `com.sun.identity.agents.config.webservice.responseprocessor`

**Web Service Internal Error Content File**

Specifies a file the agent uses to generate an internal error fault for the client application.

Property: `com.sun.identity.agents.config.webservice.internalerror.content`

**Web Service Authorization Error Content File**

Specifies a file the agent uses to generate an authorization error fault for the client application.

Property: `com.sun.identity.agents.config.webservice.autherror.content`

**Alternate Agent URL properties**

**Alternative Agent Host Name**

Specifies the host name of the agent protected server to show to client browsers, rather than the actual host name.

Property: `com.sun.identity.agents.config.agent.host`

**Alternative Agent Port Name**

Specifies the port number of the agent protected server to show to client browsers, rather than the actual port number.

Property: `com.sun.identity.agents.config.agent.port`

#### Alternative Agent Protocol

Specifies the protocol used to contact the agent from the browser client browsers, rather than the actual protocol used by the server. Either `http` or `https`.

Property: `com.sun.identity.agents.config.agent.protocol`

### JBoss Application Server properties

#### WebAuthentication Available

When enabled, allow programmatic authentication with the JBoss container using the WebAuthentication feature. This feature works only with certain versions of JBoss when the `J2EE_POLICY` or `ALL` filter mode is in use.

Property: `com.sun.identity.agents.config.jboss.webauth.available`

#### Note

This setting is not necessary for the JBoss v7 agent.

### Cross Site Scripting Detection properties

#### Possible XSS code elements

Specifies strings that, when found in the request, cause the agent to redirect the client to an error page.

Property: `com.sun.identity.agents.config.xss.code.elements`

#### XSS detection redirect URI

Maps applications to URIs of customized pages to which to redirect clients upon detection of XSS code elements.

For example, to redirect clients of `MyApp` to `/myapp/error.html`, enter `MyApp` as the Map Key and `/myapp/error.html` as the Corresponding Map Value.

Property: `com.sun.identity.agents.config.xss.redirect.uri`

### Post Data Preservation properties

#### POST Data Preservation

Enables HTTP POST data preservation, storing POST data before redirecting the browser to the login screen, and then autosubmitting the same POST after successful authentication to the original URL.

Property: `com.sun.identity.agents.config.postdata.preserve.enable`

**Missing PDP entry URI**

Specifies a list of application-specific URIs if the referenced Post Data Preservation entry cannot be found in the local cache because it has exceeded its POST entry TTL. Either the agent redirects to a URI in this list, or it shows an HTTP 403 Forbidden error.

Property: `com.sun.identity.agents.config.postdata.preserve.cache.noentry.url`

**POST entry TTL**

POST data storage lifetime in milliseconds. Default: 300000.

Property: `com.sun.identity.agents.config.postdata.preserve.cache.entry.ttl`

**PDP Stickysession mode**

Specifies whether to create a cookie, or to append a query string to the URL to assist with sticky load balancing.

Property: `com.sun.identity.agents.config.postdata.preserve.stickyssession.mode`

**PDP Stickysession key-value**

Specifies the key-value pair for stickyssession mode. For example, a setting of `lb=myserver` either sets an `lb` cookie with `myserver` value, or adds `lb=myserver` to the URL query string.

Property: `com.sun.identity.agents.config.postdata.preserve.stickyssession.value`

## Custom properties

### Custom Properties

Additional properties to augment the set of properties supported by agent. Such properties take the following forms.

- `customproperty=custom-value1`
- `customlist[0]=customlist-value-0`
- `customlist[1]=customlist-value-1`
- `custommap[key1]=custommap-value-1`
- `custommap[key2]=custommap-value-2`

Property: `com.sun.identity.agents.config.freeformproperties`

## 5.7 Configuring Web Service Provider Policy Agents

This section covers Web Service Provider (WSP) properties. WSPs both validate incoming web service requests from Web Service Clients (WSC), and also secure outgoing responses sent back to WSCs.

## Note

The current implementation of the WSSAuth Authentication module, along with the Web Service Provider, Web Service Client and STS Client Policy agents, were developed for versions 1.0 and 1.3 of the WS-Trust standard. ForgeRock now includes configuration options for the RESTful Security Token Service (STS), compatible with version 1.4 of the WS-Trust standard, which uses client certificate and Kerberos JASPI authentication modules.

For more information on the new implementation of STS, see [The RESTful Security Token Service](#).

After creating a WSP profile, you access WSP properties in the OpenAM console under Access Control > *Realm Name* > Agents > Web Service Provider > *Agent Name*.

### General properties

#### Group

For assigning the agent to a previously configured agent group in order to inherit selected properties from the group.

#### Password

Agent password used when creating the password file and when installing the agent.

#### Status

Status of the agent configuration.

#### Universal Identifier

OpenAM identifier for the agent configuration.

### Security properties

#### Security Mechanism

Specifies the mechanisms allowed to validate the web service request.

#### Authentication Chain

Specifies which OpenAM authentication chain consumes the credentials from the web service request to authenticate the WSC.

**Token Conversion Type**

Specifies how to convert the incoming token before issuing requests to other WSPs.

**Preserve Security Headers in Message**

Yes means the agent preserves SOAP security headers from the request for subsequent processing.

**Detect Message Replay**

Yes means the agent checks whether the request is a replay of an earlier request, and if so, rejects the request.

**Detect User Token Replay**

Yes means the agent checks whether the user token is a replay from an earlier request, and if so, rejects the request.

**Private Key Type**

Specifies the type of key, such as PublicKey, used to verify the request signature.

**Liberty Service Type URN**

Specifies the Universal Resource Name for the Liberty service type used for lookups.

**DNS Claim**

Specifies a Uniform Resource Identifier shared by the WSP and WSC.

**Credential for User Token**

Specifies the user name and password credentials compared with the user name security token in a request.

## **SAML Configuration properties**

**SAML Attribute Mapping**

Maps SAML attribute names from the incoming request to attribute names as retrieved from the SSOToken or the identity repository, used to have the Security Token Service generate an appropriate SAML assertion.

**SAML NameID Mapper Plugin**

Specifies the class name of a plugin used to perform SAML account mapping.

**SAML Attributes Namespace**

Identifies the attribute name space used when generating SAML assertions.

**Include Memberships**

Yes means the agent includes the principal's membership as a SAML attribute.

## **Signing and Encryption properties**

Is Request Signature Verified

Yes means verify signatures in requests.

Is Response Signed Enabled

Yes means the agent signs the specified parts of the response with its x509 certificate.

Signing Reference Type

Specifies how the x509 certificate used to sign responses is referenced in the response.

Is Request Decrypted

Yes means do decrypt the specified parts of incoming requests.

Is Response Encrypted

Yes means do encrypt the outgoing response.

Encryption Algorithm

Specifies whether to use Advanced Encryption Standard, corresponding to an Encryption Strength of 128, 192, or 256, or to use Triple DES with a key length of 0, 112, or 168.

Encryption Strength

Specifies the key length used for encryption.

## **Key Store properties**

Public Key Alias of Web Service Client

Specifies the alias of the certificate in the key store used to verify request signatures and encrypt responses.

Private Key Alias

Specifies the alias of the certificate in the key store used to sign responses and decrypt requests.

Key Store Usage

If you use your own, custom key store, specify how to access it here.

## **End Points properties**

Web Service Security Proxy End Point

If the WSC sends requests through a web service proxy, specify that as the end point here.

**Web Service End Point**

Specifies the end point to which the WSC sends requests.

**Kerberos Configuration properties**

**Kerberos Domain Server**

Specifies the fully qualified domain name of the Kerberos Distribution Center service.

**Kerberos Domain**

Specifies the Kerberos Distribution Center domain name. For Windows environments this is the domain controller domain name.

**Kerberos Service Principal**

Specifies the Kerberos principal used by OpenAM, using the form `HTTP/openam-fqdn@krb-domain`, where `openam-fqdn` is the fully qualified domain name for OpenAM, and `krb-domain` is the Kerberos Domain.

**Kerberos Key Tab File**

Specifies the Kerberos keytab file using the form `openam-host.HTTP.keytab`, where `openam-host` is the host name for OpenAM.

**Verify Kerberos Signature**

Yes means the agent signs the Kerberos token.

## 5.8

## Configuring Web Service Client Policy Agents

This section covers Web Service Client (WSC) properties. WSCs both secure outgoing requests sent to Web Service Providers (WSP), and also validate incoming from WSPs.

**Note**

The current implementation of the WSSAuth Authentication module, along with the Web Service Provider, Web Service Client and STS Client Policy agents, were developed for versions 1.0 and 1.3 of the WS-Trust standard. ForgeRock now includes configuration options for the RESTful Security Token Service (STS), compatible with version 1.4 of the WS-Trust standard, which uses client certificate and Kerberos JASPI authentication modules.

For more information on the new implementation of STS, see [The RESTful Security Token Service](#).

After creating a WSC profile, you access WSC properties in the OpenAM console under Access Control > *Realm Name* > Agents > Web Service Client > *Agent Name*.

## General properties

### Group

For assigning the agent to a previously configured agent group in order to inherit selected properties from the group.

### Password

Agent password used when creating the password file and when installing the agent.

### Status

Status of the agent configuration.

### Universal Identifier

OpenAM identifier for the agent configuration.

## Security properties

### Security Mechanism

Specifies the mechanism used to secure web service requests.

### STS Configuration

Specifies the agent used to secure requests to the Security Token Service.  
Associated with the STSSecurity Security Mechanism.

### Discovery Configuration

Specifies the agent used to secure requests to the Discovery Service.  
Associated with the LibertyDiscoverySecurity Security Mechanism.

### User Authentication Required

Yes means users must authenticate to access the WSC's protected page.

### Preserve Security Headers in Message

Yes means the agent preserves SOAP security headers in the request for subsequent processing.

### User Pass Through Security Token

Yes means the agent passes along the Security Token from the Subject, rather than generating a token or requesting it from the Security Token Service.

### Liberty Service Type URN

Specifies the Universal Resource Name for the Liberty service type used for lookups.

**Credential for User Token**

Specifies the user name and password credentials shared with the WSP and used to generate a Username Security Token.

**DNS Claim**

Specifies a Uniform Resource Identifier shared by the WSP and WSC.

**SAML Configuration properties**

**SAML Attribute Mapping**

Maps SAML attribute names from the outgoing request to attribute names as retrieved from the SSOToken or the identity repository.

**SAML NameID Mapper Plugin**

Specifies the class name of a plugin used to perform SAML account mapping.

**SAML Attributes Namespace**

Identifies the attribute name space used when generating SAML assertions.

**Include Memberships**

Yes means the agent includes the principal's membership as a SAML attribute.

**Signing and Encryption properties**

**Is Request Signed Enabled**

Yes means the agent signs the specified parts of the request with its x509 certificate.

**Signing Reference Type**

Specifies how the x509 certificate used to sign requests is referenced in the request.

**Is Response Signature Verified**

Yes means verify signatures in responses.

**Is Request Encryption Enabled**

Yes means do encrypt the specified parts of outgoing requests.

**Encryption Algorithm**

Specifies whether to use Advanced Encryption Standard, corresponding to an Encryption Strength of 128, 192, or 256, or to use Triple DES with a key length of 0, 112, or 168.

**Encryption Strength**

Specifies the key length used for encryption.

**Is Response Decrypted**

Yes means do decrypt the incoming response.

**Key Store properties**

**Public Key Alias of Web Service Provider**

Specifies the alias of the certificate in the key store used to sign requests and decrypt responses.

**Private Key Alias**

Specifies the alias of the certificate in the key store used to verify response signatures and encrypt requests.

**Key Store Usage**

If you use your own, custom key store, specify how to access it here.

**End Points properties**

**Web Service Security Proxy End Point**

If the WSC sends requests through a web service proxy, specify that as the end point here.

**Web Service End Point**

Specifies the end point to which the WSC sends requests.

**Kerberos Configuration properties**

**Kerberos Domain Server**

Specifies the fully qualified domain name of the Kerberos Distribution Center service.

**Kerberos Domain**

Specifies the Kerberos Distribution Center domain name. For Windows environments this is the domain controller domain name.

**Kerberos Service Principal**

Specifies the Kerberos principal used by OpenAM, using the form `HTTP/openam-fqdn@krb-domain`, where `openam-fqdn` is the fully qualified domain name for OpenAM, and `krb-domain` is the Kerberos Domain.

**Kerberos Ticket Cache Directory**

Specifies the directory in which Kerberos Ticket Granting Tickets (TGT) are cached. The **kinit** command stores the TGT from the KDC here.

---

## 5.9 Configuring Security Token Service Client Policy Agents

This section covers Security Token Service (STS) Client properties. STS clients both secure outgoing requests to trust authorities, and also validate incoming requests from trust authorities. You can configure STS clients to work with OpenAM's Security Token Service and with its Discovery Service.

## Note

The current implementation of the WSSAuth Authentication module, along with the Web Service Provider, Web Service Client and STS Client Policy agents, were developed for versions 1.0 and 1.3 of the WS-Trust standard. ForgeRock now includes configuration options for the RESTful Security Token Service (STS), compatible with version 1.4 of the WS-Trust standard, which uses client certificate and Kerberos JASPI authentication modules.

For more information on the new implementation of STS, see [The RESTful Security Token Service](#).

After creating an STS Client profile, you access STS Client properties in the OpenAM console under Access Control > *Realm Name* > Agents > STS Client > *Agent Name*.

### General properties

#### Group

For assigning the agent to a previously configured agent group in order to inherit selected properties from the group.

#### Password

Agent password used when creating the password file and when installing the agent.

#### Status

Status of the agent configuration.

#### WS-Trust Version

Specifies whether to use WS-Trust 1.3 or 1.0.

#### Universal Identifier

OpenAM identifier for the agent configuration.

### Security properties

#### Security Mechanism

Specifies the mechanism used to secure the STS request.

**STS Configuration**

Specifies the STS Client agent profile to use if the security mechanism is STS Security.

**Preserve Security Headers in Message**

Yes means the agent preserves SOAP security headers for subsequent processing.

**Credential for User Token**

Specifies the user name and password credentials the agent uses to generate a Username security token.

**Requested Key Type**

Specifies the type of key, such as PublicKey, used to encrypt responses.

**Requested Claims**

Specifies the Uniform Resource Identifiers for the claims to be represented in the Security Token.

**DNS Claim**

Specifies a Uniform Resource Identifier shared by the agent and the WSC.

**SAML Configuration properties**

**SAML Attribute Mapping**

Maps SAML attribute names from the incoming request to attribute names as retrieved from the SSOToken or the identity repository, used to have the Security Token Service generate an appropriate SAML assertion.

**SAML NameID Mapper Plugin**

Specifies the class name of a plugin used to perform SAML account mapping.

**SAML Attributes Namespace**

Identifies the attribute name space used when generating SAML assertions.

**Include Memberships**

Yes means the agent includes the principal's membership as a SAML attribute.

**Signing and Encryption properties**

**Is Response Signature Verified**

Yes means verify signatures in responses.

**Is Request Signed Enabled**

Yes means the agent signs the specified parts of the request with its x509 certificate.

**Signing Reference Type**

Specifies how the x509 certificate used to sign requests is referenced in the request.

**Is Request Encryption Enabled**

Yes means do encrypt the specified parts of requests.

**Is Response Decrypted**

Yes means do decrypt the response.

**Encryption Algorithm**

Specifies whether to use Advanced Encryption Standard, corresponding to an Encryption Strength of 128, 192, or 256, or to use Triple DES with a key length of 0, 112, or 168.

**Encryption Strength**

Specifies the key length used for encryption.

**Key Store properties**

**Public Key Alias of Web Service Provider**

Specifies the alias of the certificate in the key store used to verify response signatures and encrypt requests.

**Private Key Alias**

Specifies the alias of the certificate in the key store used to sign requests and decrypt responses.

**Key Store Usage**

If you use your own, custom key store, specify how to access it here.

**End Points properties**

**Security Token Service End Point**

Specifies the URL to the Security Token Service end point.

**Security Token Service MEX End Point**

Specifies the URL to the Security Token Service message exchange end point.

**Kerberos Configuration properties**

**Kerberos Domain Server**

Specifies the fully qualified domain name of the Kerberos Distribution Center service.

**Kerberos Domain**

Specifies the Kerberos Distribution Center domain name. For Windows environments this is the domain controller domain name.

**Kerberos Service Principal**

Specifies the Kerberos principal used by OpenAM, using the form `HTTP/openam-fqdn@krb-domain`, where `openam-fqdn` is the fully qualified domain name for OpenAM, and `krb-domain` is the Kerberos Domain.

**Kerberos Ticket Cache Directory**

Specifies the directory in which Kerberos Ticket Granting Tickets (TGT) are cached. The `kinit` command stores the TGT from the KDC here.

## 5.10 Configuring Version 2.2 Policy Agents

This section covers version 2.2 agent properties. Version 2.2 agents store their configurations locally, with a user name, password combination used to connect to OpenAM.

### Warning

ForgeRock no longer supports 2.2 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 OpenAM console under Access Control > *Realm Name* > Agents > 2.2 Agents > *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.

**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.

## 5.11 Configuring OAuth 2.0 & OpenID Connect 1.0 Clients

When you want 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 OpenAM console, then create an OAuth 2.0 Client agent profile. After creating the agent profile, you can further configure the properties in the OpenAM console under Access Control > *Realm Name* > Agents > OAuth 2.0 Client > *Client Name*.

The topmost configuration fields are for both OAuth 2.0 and OpenID Connect 1.0, whereas others are specifically for OpenID Connect 1.0.

- [Common Client Configuration](#)
- [OpenID Connect 1.0 Client Configuration](#)

## Common Client Configuration

The following configuration fields are common to OAuth 2.0 and OpenID Connect 1.0 clients.

### Group

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

### Status

Whether the client profile is active for use.

### Client password

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

### Client type

Confidential clients can maintain confidentiality of their credentials. Public clients cannot.

A web application running on a server where its credentials are protected is an example of a confidential client.

A JavaScript client running in a browser is an example of a public client.

### 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.

Redirection URIs are required for OpenID Connect 1.0 clients.

**Scopes**

Specify scopes in *scope* or *scope|locale|localized description* format. These scopes are to be presented to the resource owner when the resource owner is asked to authorize client access to protected resources.

**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*.

**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*.

**Default Scope(s)**

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

**Response Types**

Specify response type lists that the client uses.

The default lists are the following.

- code
- token
- id\_token
- code token
- token id\_token
- code id\_token
- code token id\_token

**Contacts**

Specify email address of users who administer the client.

**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.

## Configuring OAuth 2.0 & OpenID Connect 1.0 Clients

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

```
-----BEGIN CERTIFICATE-----
MIIDETCCAfmgAwIBAgIEU8SXLjANBgkqhkiG9w0BAQsFADA5MRswGQYDVQQKExJvcGVuYW0uZXhh
bXBsZS5jb20xGjAYBgnVBAMTEwp3dC1iZWfYZXItY2xpZW50MB4XDTE0MTAyNzExNTY1Nl0XDTI0
MTAyNDExNTY1Nl0wOTEbMBkGA1UEChMSb3BlbmFtLmV4YW1wbGUuY29tMRowGAYDVQQDExFqd3Qt
YmVhcmVylWNsaWVudCCASiWxDQYJKoZIhvcaNAQEBBQAQggEPADCCAQoCggEBAID4ZZ/DIGEBr4QC
2uz0GYFOCULAPanx21aYHsvELsWyMa7DJLD+mnjaF8cPRRMkhYZFXDJo/AVcjyblyT3ntql+2Js
3D7TmS6BSjKxZwsJyjhJIYEoUwvloc0kizgSm15MwMbcbnksQVN5VWi0e4y4JMb30t6k38lM62K
KtaSPP6jvnW1lTmL9uiqlWz54AM6hU3NLc13J6Rfh8waBIPAEjmHNzq0l2uGwUmzubYDFJbomL
SQq058RuKVaSVMwDbmENtMYWXIKQL2xTt5XAbwEQEgJ/zskwpA2a0t1HE6de3Uym0hONhRiu4rk3
AIEnEVbxrvy4Ik+wXg7LZVsCAwEAAsMhMB8wHQYDVR00BBYEFiIuI7ejuZTg5tJsh1XyRopGOMBcs
MA0GCSqGSIb3DQEBCwUA4IAQBM/+tYYVIS6LvPl3mfE8V7x+VPXqj/uK6UecAbfmRTrPk1ph+
jji6nmLX9ncomYALWL/JFiSXcVsZt3/412f0qjakFVs0PmK1vEPxdlav1drnVA33icy1w0RRRus5/
qA6mwDYPAZSbm5cDVvCR7Lt6VqJ+D0V8GABFxUw9IaX6ajTqkWhldY77usvNeTD0Xc4R70qSBrnA
SNcaUlJogWyzhbFlmE9Ne28j4RVpbz/EZn0oc/cHTJ6Lryzsivf4uD01m3M3kM/MUyXc1Zv3rqBj
TeGsgcqEAd6XLGXY1l+M/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, O=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-----
MIIDETCCAfmgAwIBAgIEU8SXLjANBgkqhkiG9w0BAQsFADA5MRswGQYDVQQKExJvcGVuYW0uZXhh
bXBsZS5jb20xGjAYBgnVBAMTEwp3dC1iZWfYZXItY2xpZW50MB4XDTE0MTAyNzExNTY1Nl0XDTI0
MTAyNDExNTY1Nl0wOTEbMBkGA1UEChMSb3BlbmFtLmV4YW1wbGUuY29tMRowGAYDVQQDExFqd3Qt
YmVhcmVylWNsaWVudCCASiWxDQYJKoZIhvcaNAQEBBQAQggEPADCCAQoCggEBAID4ZZ/DIGEBr4QC
2uz0GYFOCULAPanx21aYHsvELsWyMa7DJLD+mnjaF8cPRRMkhYZFXDJo/AVcjyblyT3ntql+2Js
```

## Configuring Agent Authenticators

```
-----END CERTIFICATE-----
```

### OpenID Connect 1.0 Client Configuration

The following optional configuration fields are for OpenID Connect 1.0 clients.

#### ID Token Signed Response Algorithm

Algorithm that the ID Token for this client must be signed with

Default: RS256 (RSA with SHA256, where the RSA key comes from the OpenAM keystore)

Valid values are listed in *JSON Web Algorithms (JWA)*: "[alg](#)" (*Algorithm Header Parameter Values for JWS*). OpenAM supports HmacSHA256, HmacSHA384, and HmacSHA512.

#### Post Logout Redirect URI

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

#### Access Token

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

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

## 5.12 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 OpenAM console under Access Control > *Realm Name* > 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 allow 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.

---

## Chapter 6

# Configuring Audit Logging

OpenAM servers generate two types of log files: audit logs and debug logs.

Audit logs capture normal operational information about OpenAM usage. Audit file records are structured: they adhere to a consistent, documented file format. You can configure OpenAM to write audit logs to flat files, relational database tables, or a syslog server.

OpenAM policy agents also produce agent audit logs. Agent audit logs provide a trail of policy agent activity.

OpenAM services can capture a variety of information in debug logs, which you can use when troubleshooting OpenAM. For more information about the OpenAM debug logs, see [Debug Logging](#).

The OpenAM Logging Service encapsulates the audit logging configuration. The Logging Service is an OpenAM system service. There is a single audit logging configuration for the entire OpenAM deployment.

OpenAM lets you change the log level on the fly. OpenAM also supports log rotation, secure logging, and log message buffering.

## 6.1 Configuring Audit Logging

To configure OpenAM logging properties, log in to the OpenAM console as OpenAM administrator, and browse to Configuration > System > Logging.

For more information on the available settings, see the [Audit Logging](#) reference.

## 6.2 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.

See the [Log Messages](#) reference for details.

## 6.3 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 OpenAM console, or the ssoadm command.

### Procedure 6.1. Enabling Syslog Audit Logging by Using the OpenAM Console

1. Login to the OpenAM console as OpenAM administrator.
2. Browse to Configuration > System > Logging.
3. Set the *Logging Type* option to **Syslog**.
4. Complete the following settings as appropriate for your syslog server:
  - **Syslog server host**
  - **Syslog server port**
  - **Syslog server protocol**
  - **Syslog facility**
  - **Syslog connection timeout**

For information on these settings, see the [Audit Logging](#) reference.

5. Save your work.

### Procedure 6.2. Enabling Syslog Audit Logging by Using SSOADM

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.

## 6.4 Audit Logging in OpenAM 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 browse to Access Control > *Realm Name* > Agents > *Agent Type* > *Agent Name* > Global, and then scroll down to the Audit section.



---

## **Chapter 7**

# **Working with Mobile Devices & Applications**

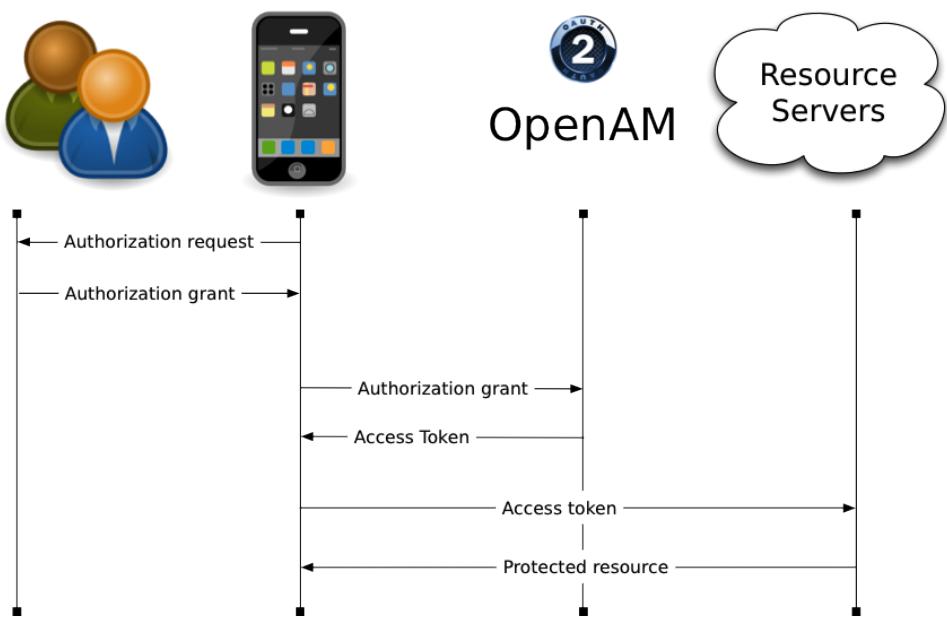
When building applications that run on mobile devices, you can use the same OpenAM service that you also use for access management in your web, cloud, and other applications. OpenAM has features that make it particularly well suited for the mobile world, too.

## **7.1**

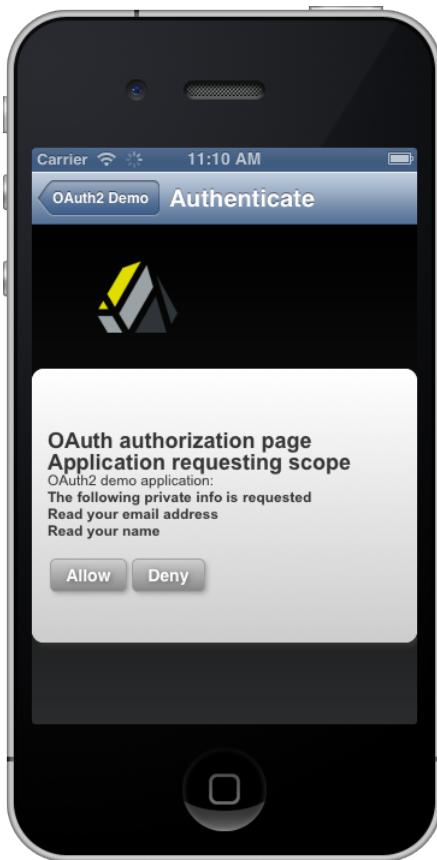
### **Simplifying Access on Mobile Devices**

On many mobile devices, users want to avoid repeatedly entering credentials such as an email address or user name and a password. They do not want new credentials to manage for every application they try. They do not want to share their credentials across applications. Instead users want single sign-on with few identity providers. They want to authorize access for applications rather than share their credentials.

OpenAM supports modern web standards including [OAuth 2.0](#), [OpenID Connect 1.0](#), and [GSMA Mobile Connect](#). After registering an application with OpenAM as an OAuth/OpenID Connect client, the application can then redirect a user to OpenAM to authenticate and to authorize access to resources that the user owns, such as profile data. The application gets an access token that can be used to gain authorized access without requiring the user to share credentials. OpenID Connect extends OAuth, standardizing how client applications discover and register with identity providers, and also defining how applications can manage user sessions and handle logout when they no longer want to authorize access.



An OAuth 2.0 client application can thus make simplify the user experience on the phone to authorizing access.



In addition to serving as an identity provider, OpenAM can also function as an OAuth 2.0 client, protecting access to resources within your control based on authorization granted by an identity provider who users already know and use, such as Facebook, Google, MSN and others. OpenAM's built in authorization policy management makes it straightforward to integrate this capability into your applications.

The OAuth and OpenID Connect standards specify REST interfaces, making them essentially programming language-independent and accessible for web applications and mobile applications alike.

Mobile Connect is an application of OpenID Connect that enables authentication to work through a mobile phone, regardless of the service provided or the device consuming the service. Mobile Connect therefore allows Mobile Network Operators to act as an identity provider for their customers. OpenAM fits well in Mobile Connect deployments as it can play both the role of OpenID Provider and

also of Authenticator, with many authentication modules built in as described in [Section 7.2, “Protecting Access for Mobile Users”](#). For details on using OpenAM in a Mobile Connect installation, see [Using OpenAM with Mobile Connect](#).

OpenAM also supports [Open Authentication](#) architecture with the OATH module mentioned in the next section.

## 7.2 Protecting Access for Mobile Users

You must give users access to your organization's resources while they are on the go. At the same time you must manage risk. OpenAM supports risk-based adaptive authentication, device fingerprints, one-time passwords and other multi-factor authentication capabilities that help you do both. As OpenAM handles authentication through plugin modules that you can chain, your OpenAM service can meet a variety of requirements.

OpenAM's [Adaptive Risk authentication module](#) lets you add risk assessment to any authentication module chain, dynamically requiring stronger authentication when circumstances require it (new location, ancient last login time, new device, new IP address, specific application, and so forth). You can add the [Device Print module](#) to an authentication chain to fingerprint users' devices for additional risk assessment, making it easier to handle sign-on when users bring their own devices.

**AdaptiveRisk - Properties**

Save   Reset   Back to Authentication

(3 Item(s))

|                          | Instance     | Criteria   | Options |
|--------------------------|--------------|------------|---------|
| <input type="checkbox"/> | LDAP         | REQUIRED   |         |
| <input type="checkbox"/> | AdaptiveRisk | SUFFICIENT |         |
| <input type="checkbox"/> | HOTP         | REQUIRED   |         |

OpenAM also lets you decide exactly what stronger authentication means in your situation. You can for example add multi-factor authentication involving mobile devices using OpenAM's [OATH](#) and [HOTP](#) modules either to use a one-time password soft token generated on a device, or to send a one-time password in a text message to a mobile phone.

In addition to capabilities supporting new applications, OpenAM integrates well with existing systems needed by users on the move. Whether users are authenticating from a mobile device through a gateway using an MSISDN, starting single sign-on by logging on to a laptop, or connecting to a VPN with certificate based authentication, OpenAM has an authentication module for that.

New Module Instance

\* Name:

\* Type:

- Active Directory
- Adaptive Risk
- Anonymous
- Certificate
- Data Store
- Device Print
- Federation
- HOTP
- HTTP Basic
- JDBC
- LDAP
- Membership
- MSISDN
- OATH
- OAuth 2.0
- Persistent Cookie
- RADIUS
- SAE
- SecurID
- Windows Desktop SSO
- Windows NT
- WSSAuth

OK Cancel

\* Indicates required field

All of these capabilities are available with OpenAM out of the box.

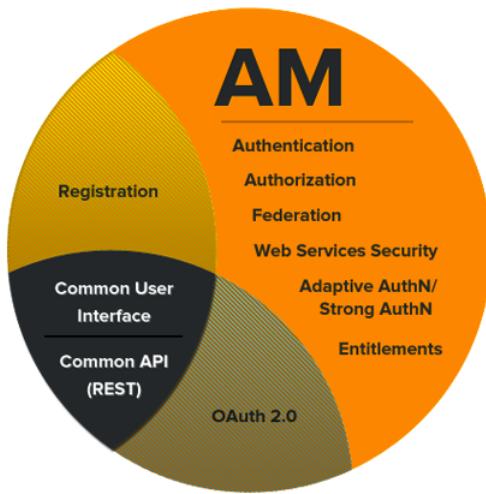
### 7.3

## Simplifying Access with REST APIs

Representational State Transfer (REST) is a architectural style designed in parallel with HTTP. REST simplifies integration and deployment while enabling layered, web-scale services. REST APIs in OpenAM implement REST in a way that reuses common HTTP verbs and decouples APIs from the programming languages that developers use to interact with them. OpenAM exposes REST APIs for many capabilities such as those in the following list.

- Authentication (including a callback mechanism so applications can work with all OpenAM authentication modules)
- Logout
- Managing groups
- Managing policy agent profiles
- Managing realms

- Managing user profiles
- OAuth 2.0 authorization
- OpenAM native authorization
- OpenID Connect 1.0 authorization
- Resetting forgotten passwords
- Token validation
- User self-registration



As [OpenAM REST APIs](#) provide language-independent access, they make it easier to build cross-device applications. Developers can use the same APIs to access OpenAM both from web applications and also from native mobile applications.

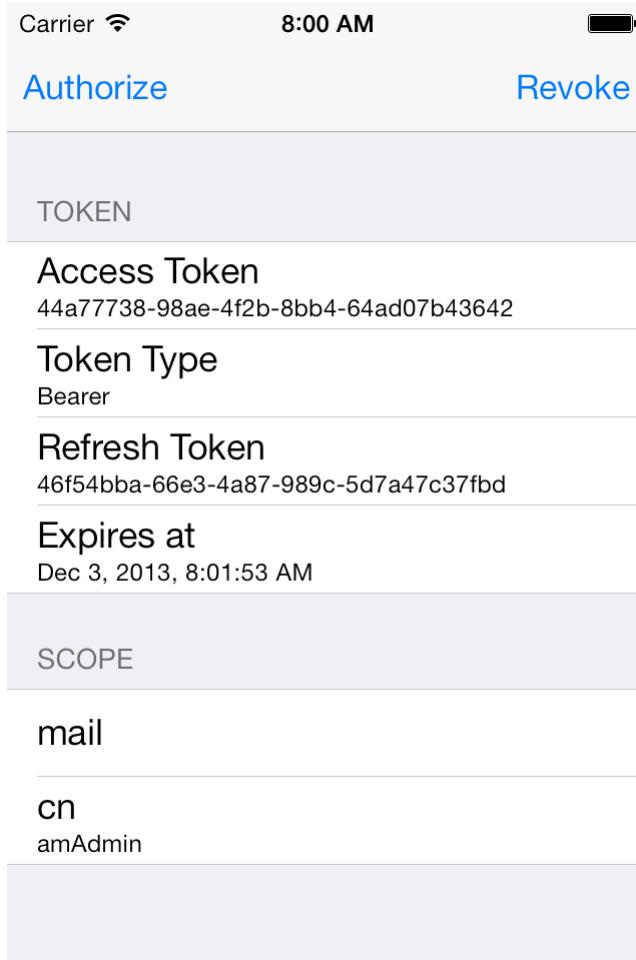
Furthermore OpenAM REST APIs are built on an underlying common REST framework, designed to provide common access to resource providers. The common REST framework standardizes both how resource providers serve standard requests (create, read, update, delete, query, patch), and also how resource providers offer extended operations in a managed way (using actions). Applications built to interact with OpenAM REST APIs increasingly can interoperate with other products in the ForgeRock stack such as [OpenIDM](#) for identity management and [OpenDJ](#) for highly available data.

## 7.4

## Getting Source Code for Sample Mobile Applications

You can get source code for sample mobile applications from ForgeRock's public source code repository under <https://svn.forgerock.org/commons/samples/mobile>.

For example, if you have a Mac running OS X 10.8 or later with Xcode installed, try the [OpenAM OAuth 2.0 iOS Sample App](#).





---

## **Chapter 8**

# **Configuring User Self-Service Features**

This chapter focuses on how to enable OpenAM features that allow users to self register from the Login page and reset their own passwords in secure fashion.

## **8.1**

### **Configuring User Self-Registration**

OpenAM provides a self-registration feature that allows users to add themselves to the system. On the Login page, the user clicks a Register link, which sends a request to the OpenAM server. OpenAM responds to request by sending a Register Your Account page where the user enters his or her email address.

After the user enters his or her email, OpenAM responds by sending a notification containing a confirmation link to the user's email address. When the user clicks the link, OpenAM confirms the operation and presents the user with a registration page where the user enters their account information.

#### **Procedure 8.1. To Configure User Self-Registration**

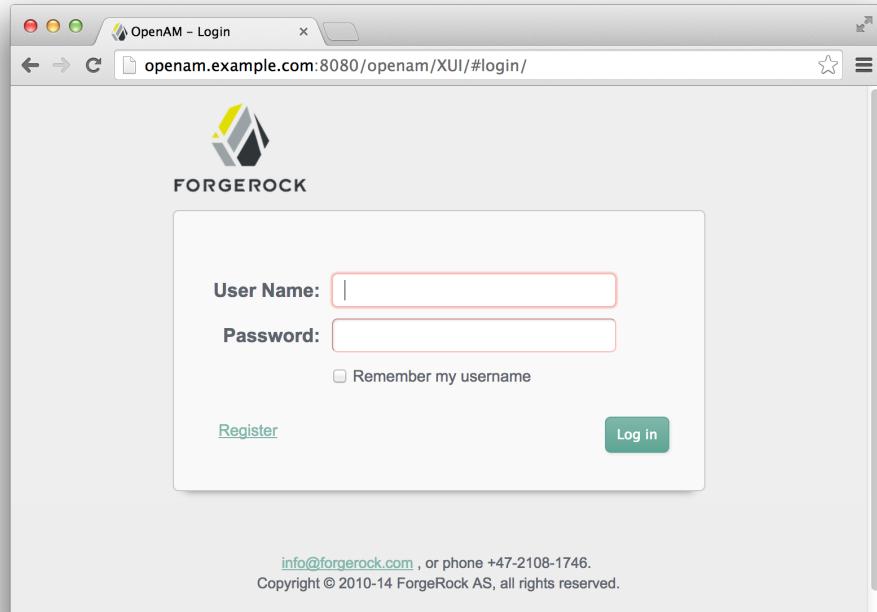
1. Configure the Email Service to send mail notifications to users who self-register.

You can configure these globally in OpenAM console at Configuration > Global > Email Service. Alternatively, you can configure them for an individual realm under Access Control > *Realm Name* > Services.

### 2. Configure User Self Service to enable self-registration.

You can configure these globally in OpenAM console at Configure > Global > User Self Service. On the User Self Service page, click the Enabled checkbox next to Self-Registration for Users, and then click Save.

At this point users can self-register by clicking a Register link on the Login page.



## 8.2 About Password Reset

Users who know their passwords, but must reset them because for example the password is going to expire, can reset their passwords by successfully authenticating to OpenAM, visiting their end user pages, such as <http://openam.example.com:8080/openam/XUI/#profile/>, and clicking Change Security Data to display the change password page.

**Figure 8.1. OpenAM Security Data Change page**

The screenshot shows a modal window titled "Security data change". It contains three input fields: "Old password", "New password", and "Confirm Password". Below the "Old password" field is a note: "Please enter new password in the fields below to change your password.". To the right of the "New password" and "Confirm Password" fields are two validation messages: "At least 8 characters" and "Confirmation matches password". At the bottom right of the modal is a "Update" button.

You therefore do not need to configure password reset for users who can remember their current password. Instead, you point them to the XUI/#profile page to let them do it themselves.

### 8.3 Resetting Forgotten Passwords (Legacy)

OpenAM can provide self-service password reset for forgotten passwords when end user pages are served by the classic UI. To enable self-service password reset, you must configure the password reset service itself, which consists mainly of setting up secret questions, and configuring an SMTP mail server to send reset passwords to the users of the service.

#### Tip

Users must be able to access their mail after the service resets their passwords, or they will not be able to receive the new password. Therefore, do not set up the service to reset the password used to access the email account specified in the user's profile.

#### Procedure 8.2. To Set Up the Password Reset Service

You can configure the password reset service for OpenAM, letting each realm inherit the global settings. Alternatively, you can choose to configure the service only for an individual realm.

## Tip

Resetting a user password can have repercussions on the user profile. For example, a user data store directory service could enforce a policy to require password changes on reset. OpenAM's LDAP authentication module can deal with policies based on the Behera Internet-Draft, [Password Policy for LDAP Directories](#), though its default DataStore authentication module cannot.

When the user data store directory service enforces password reset policies, then either use the LDAP authentication module in the authentication chain, or rely on some other means of resetting user passwords.

1. Configure the Password Reset service in one of the following ways.
  - To configure the service globally for all realms, login to OpenAM Console as administrator and browse to Configuration > Global > Password Reset in the Global Properties list.
  - To configure the service for a particular realm, login to OpenAM console as the realm administrator and browse to Access Control > *Realm Name* > Services, then click Add... to add a new Password Reset service configuration.
2. In the Password Reset page, use the following hints to adjust settings, and then save your work.

In addition to the User Validation and Secret Question values provided, you must configure at least the Bind DN and Bind Password of the user who can reset passwords in the LDAP data store.

### User Validation

OpenAM uses this LDAP attribute and the value entered by the user to look up the user profile in the data store.

### Secret Question

This list corresponds to property values held in the file `amPasswordReset.properties` inside `openam-core-12.0.0.jar`, which you can find under `WEB-INF/lib/` where OpenAM is installed.

To make changes, extract a version from `openam-core-12.0.0.jar`, copy it to `WEB-INF/classes/` where OpenAM is deployed, and then edit `WEB-INF/classes/amPasswordReset.properties`.

Localized versions of this file are named `amPasswordReset_locale.properties`. You should localize only the questions

at the end, leaving the rest of the localized file as is. For example if the default properties file contains:

```
favourite-restaurant=What is your favorite restaurant?
```

Then WEB-INF/classes/amPasswordReset\_fr.properties ought to contain:

```
favourite-restaurant=Quel est votre restaurant préféré ?
```

After changing these files, you must restart OpenAM.

#### Search Filter

An additional LDAP search filter you specify here is &-ed with the filter constructed for user validation to find the user entry in the data store.

#### Base DN

If you specify no base DN for the search, the search for the user entry starts from the base DN for the realm.

#### Bind DN

The DN of the user with access to change passwords in the LDAP data store.

#### Bind Password

The password of the user with access to change passwords in the LDAP data store.

#### Reset Password Creator

Classname of a plugin that implements the PasswordGenerator interface.

Default: com.sun.identity.password.plugins.RandomPasswordGenerator

#### Password Reset Notification Class

Classname of a plugin that implements the NotifyPassword interface.

Default: com.sun.identity.password.plugins.EmailPassword

#### Password Reset

Enables the service.

#### Personal Question

When enabled, allows the user to create custom secret questions.

#### Maximum Number of Questions

Maximum number of questions to ask during password reset.

#### Force Change Password on Next Login

When enabled, the user must change her password next time she logs in after OpenAM resets her password.

#### Password Reset Failure Lockout

When enabled, the user only gets the specified number of tries before her account is locked.

#### Password Reset Failure Lockout Count

If Password Reset Failure Lockout is enabled, this specifies the maximum number of tries to reset a password within the specified interval before the user's account is locked.

#### Password Reset Failure Lockout Interval

This interval applies when Password Reset Failure Lockout is enabled, and when Password Reset Failure Lockout Count is set. During this interval, a user can try to reset her password the specified number of times before being locked out. For example, if this interval is 5 minutes and the count is set to 3, a user gets 3 tries during a given 5 minute interval to reset her password.

#### Email Address to Send Lockout Notification

This specifies the administrator address(es) which receive(s) notification on user account lockout. Each address must be a full email address such as `admin@example.com`, or `admin@host.domain`.

OpenAM must be able to send mail through an SMTP-capable service for this to work. See [Procedure 8.3, “To Set Up SMTP Mail Notification”](#).

#### Warn User After N Failures

If you configure Password Reset Failure Lockout, set this to warn users who are about to use up their count of tries.

#### Password Reset Failure Lockout Duration

If you configure Password Reset Failure Lockout, set this to a number of minutes other than 0 so that lockout is temporary, requiring only that the locked-out user wait to try again to reset her password, rather than necessarily require help from an administrator.

#### Password Reset Lockout Attribute Name

If you configure Password Reset Failure Lockout, then OpenAM sets sets data store attribute to `inactive` upon lockout.

#### Password Reset Lockout Attribute Value

If set to `inactive`, then a user who is locked out cannot attempt to reset her password if the Password Reset Failure Lockout Duration is 0.

**Password Reset E-mail Attribute Name**  
Identity attribute that holds the user's email address.

Default: mail

3. If you changed Secret Questions in the WEB-INF/classes/amPasswordReset.properties file or in any localized versions, restart OpenAM for the changes to take effect.

### **Procedure 8.3. To Set Up SMTP Mail Notification**

By default, OpenAM expects the SMTP service to listen on localhost:25. You can change these settings.

1. In the OpenAM console, click the Configuration > Servers and Sites > Default Server Settings.
2. In the Edit server-default page, scroll down to Mail Server to change the Mail Server Host Name or Mail Server Port Number.
3. Save your work.
4. By default, OpenAM sends password reset notifications from <Password-Administrator>.

To set a valid from address, extract amPasswordResetModuleMsgs.properties from openam-core-12.0.0.jar, copy it to WEB-INF/classes/ where OpenAM is deployed, and then edit the file to change the fromAddress.label property value, as in the following example.

```
fromAddress.label=no-reply@example.com
```

Save your work, and then restart OpenAM for the properties file change to take effect.

### **Procedure 8.4. To Prepare Users to Reset Passwords**

Before a user can reset her password, she must choose answers for secret questions.

1. When her account is first created, direct the user to her idm/EndUser page, such as <http://openam.example.com:8080/openam/idm/EndUser>, where she can provide a valid email address to recover the reset password and can edit Password Reset Options.

## Resetting Forgotten Passwords (Legacy)

**Password User**

Save | Reset | \* Indicates required field

|                         |  |
|-------------------------|--|
| First Name:             | Password   |
| * Last Name:            | User   |
| * Full Name:            | Password User                                    |
| Password:               | <a href="#">Edit</a>                             |
| Email Address:          | my.email@example.com                             |
| Telephone Number:       |  |
| Home Address:           |  |
| Password Reset Options: | <a href="#">Edit</a>                             |
| Universal ID:           | id=pwduser,ou=user,dc=openam,dc=forgerock,dc=org |

By default OpenAM console redirects end users to this page when they login.

2. After the user updates her secret questions, she can use the password reset service when necessary.

**Password Reset Options**

Save | Reset | Close

| Questions (2 Questions)             |                                   |
|-------------------------------------|-----------------------------------|
| <input checked="" type="checkbox"/> | Question                          |
| <input checked="" type="checkbox"/> | What is your favorite restaurant? |
| <input checked="" type="checkbox"/> | Answer                            |
| <input checked="" type="checkbox"/> | Anything Tibetan                  |
| <input checked="" type="checkbox"/> | What is your favorite password?   |
| <input checked="" type="checkbox"/> | secret12                          |

### Note

Answers to secret questions are case sensitive.

### Procedure 8.5. To Direct Users to Reset Passwords

Having setup her email and answers to secret questions, the user can use the reset password service.

Create a test subject and use these steps to validate your configuration.

1. Send the user with a forgotten password to enter her user ID at the password reset URL.

If the user is in the default realm use password at the end of the URL to OpenAM, as in `http://openam.example.com:8080/openam/password`.

If the password reset service is enabled only for the user's realm and not the parent realm, or the realm to reset the password is different from the user's default realm, use `ui/PWResetUserValidation?realm=realm name`, as in `http://openam.example.com:8080/openam/ui/PWResetUserValidation?realm=realm name`.

## Password Reset User Validation

User ID:

**NEXT**

2. The user answers the specified questions, and clicks OK.

OpenAM resets the password, sending mail to the SMTP service you configured.

## Password Question for pwduser

What is your favorite restaurant?:

What is your favorite password?:

**OK** | **PREVIOUS**

When the user clicks OK, OpenAM sends the email and shows a confirmation message.

The user receives the email with a line such as the following.

Your OpenAM password was changed to: 647bWluw

3. The user logs in using the new password.

If you configured the system to force a change on password reset, then OpenAM requires the user to change her password.

---

## Chapter 9

# Configuring Single Sign-On within One Domain

This chapter describes the configuration of *Single Sign-On* (SSO) services for multiple resources on one domain. To understand how SSO works, you need to understand some key elements of the HTTP cookie, as described in RFC 6525, [HTTP State Management Mechanism](#).

With SSO a user can access multiple independent services from a single session.

## 9.1 The Basics of the HTTP Cookie

Within an HTTP cookie, you can store a single custom *name=value* pair, such as *sessionid=value*. Other custom names within a cookie are as follows.

### Domain

Normally set to the domain from where the cookie was issued. To work with multiple subdomains, the *Domain* should be set to a URL like *Domain=example.net*. This is also known as the cookie domain, as defined in the [Configuration Reference](#) chapter of the Reference document. A cookie domain set to *example.net* will work for subdomains such as *app1.example.net* and *service1.example.net*.

### Path

The directory in the URL to which the cookie applies. If the *Path =/openam*, the cookie applies to the */openam* subdirectory of the FQDN, and lower level directories, including *openam/UI* and *openam/UI/Login*.

**Secure**

If the `Secure` name is included, the cookie can be transferred only over HTTPS. When a request is made over HTTP, the cookie is not made available to the application.

**HttpOnly**

When the `HttpOnly` name is included, that cookie will not be accessible through JavaScript. According to [RFC 6265](#), the noted flag "instructs the user agent to omit the cookie when providing access to cookies via 'non-HTTP' APIs (such as a web browser API that exposes cookies to scripts)."

**Expires**

The lifetime of a cookie can be limited, with an `Expires` name configured with a time, based on UTC (GMT).

## Note

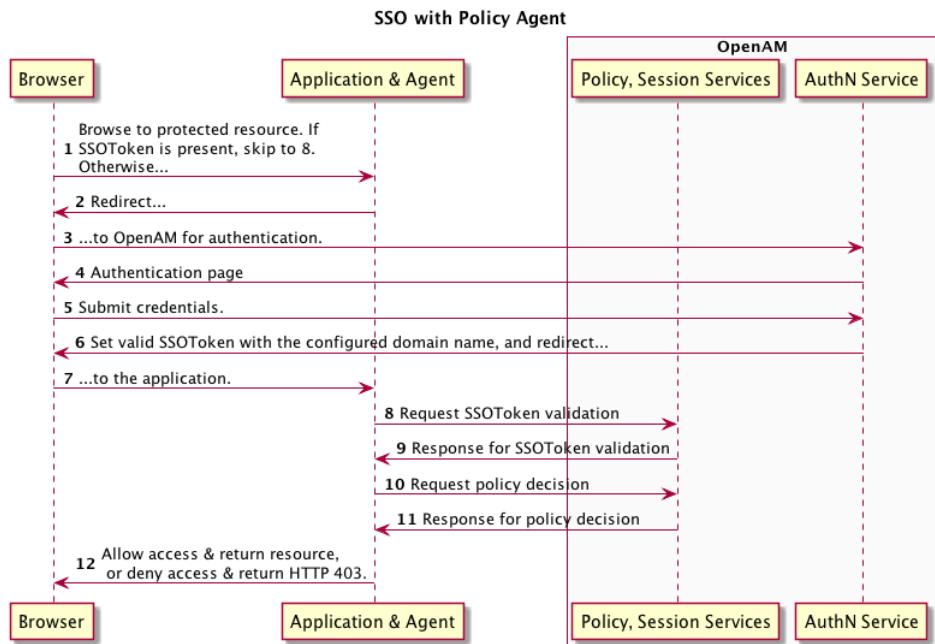
Be careful. Do not take a shortcut with a top-level domain. Web browser clients today are designed to ignore cookies set to top-level domains including `.com`, `.net`, and `.co.uk`. In addition, a cookie with a value like `Domain= app1.example.net` will not work for similar subdomains such as `app2.example.net`.

## 9.2 Cookies and the SSO Session Process

OpenAM uses cookies to track user sessions. The diagram shown next illustrates how OpenAM assigns and tracks cookies.

In the diagram:

- The domain shown in the description is `example.net`
- The protected resource application can be found on `app.example.net`
- The OpenAM server is located on `sso.example.net`.



A client points his browser to a protected resource application. An agent on the application checks the client browser cookies for the presence of a session ID, a component of an SSO Token. If such a Session ID exists and is valid, the agent requests validation (see arrow 8).

If no valid session ID currently exists, the agent redirects the client to OpenAM for authentication (AuthN). The client is then sent to OpenAM for AuthN. If the client submits valid credentials, the AuthN service creates a session cookie. The SSO Token and configured domain name is embedded in that cookie. OpenAM issues an HTTP redirect to send the client browser back to the protected resource. The SSO Token is actually a Java Object.

The agent then verifies the validity of the session with the OpenAM session service, before granting access.

### 9.3

### Potential Problems

In general, problems with SSO relate to some sort of mismatch of domain names. For example, a cookie that is configured on a third-level domain such as `sso.example.net` will not work with an application on a similar domain such as `app.example.net`. Even if the Session ID is valid, the application will not receive the SSO Token. The request is then redirected to OpenAM. The client gets what

appears as a SSO Token in the diagram, which is actually a valid SSO tracking cookie that redirects immediately, and the cycle continues. Other issues that may lead to similar problems are shown here.

- When a cookie domain does not match a domain for the protected application

Assume the application is configured on a domain named `example.org`. That application will not receive an SSO Token configured on the `example.net` domain.

- When a third-level domain is used for the SSO Token

If an SSO Token is configured on `sso.example.net`, an application on `app.example.net` does not receive the corresponding cookie. In this case, the solution is to configure the SSO Token on `example.net`.

- When the Secure flag is used with a regular HTTP application

If you need encrypted communications for an application protected by OpenAM, use the Secure flag and make sure the application is accessible over HTTPS.

- When the path listed in the cookie does not match the path for the application

Perhaps the cookie is configured with an `/helloworld` path; that won't match an application that might be configured with an `/hellomars` path. In that case, the application will not receive the cookie.

- When an inappropriate name is used for the cookie domain

As noted earlier, client browsers are configured to ignore first-level domains such as `com` and `net` as well as functional equivalents such as `co.uk` and `co.jp`.

- When working with different browsers

The `name=value` pairs described earlier may not apply to all browsers. The requirements for an HTTP cookie sent to an IE browser may differ from the requirements for other standard browsers such as Firefox and Chrome. Based on anecdotal reports, IE does not recognize domain names that start with a number. In addition, IE reportedly refuses cookies that include the underscore (`_`) character in the FQDN.

## 9.4

## Configure SSO on One Domain

Now that you have read about the SSO process, you should be able to set it up on a server configured with OpenAM and a web service protected by an OpenAM agent. The following procedure assumes that you know how to configure OpenAM, the Apache Web server, and associated OpenAM Apache agent.

### Procedure 9.1. Configure SSO on One Domain

1. Install OpenAM as described in the [OpenAM 12.0.0 Installation Guide](#). This procedure uses a Server URL of `http://openam.example.net:8080/openam`.
2. Install the appropriate policy agent, as described in the [OpenAM Web Policy Agent User's Guide](#) or the [OpenAM Java EE Policy Agent User's Guide](#). This procedure uses an agent URL of `http://app.example.net:80`, and an agent name of `webagent1`.
3. Make sure that both URLs are configured with IP addresses, as described in the chapter on [Installing OpenAM Core Services](#).
4. Return to the OpenAM server on `http://openam.example.net:8080/openam`. Log in as the administrative user, normally `amadmin`. To activate and configure the agent, follow the procedure described in the [OpenAM Web Policy Agent User's Guide](#) or the [OpenAM Java EE Policy Agent User's Guide](#).
5. Now you can configure SSO Only mode. In the OpenAM console, click Access Control > *Realm Name* > Agents > `webagent1`. Scroll down to SSO Only Mode and activate the Enabled box.
6. Save your changes.
7. Make sure you have configured the SSO domain, in this case, `example.net`. Click Configuration > System > Platform. Make sure `example.net` (or your chosen domain) is selected as a cookie domain.
8. Save your changes.
9. Restart the web server. The agent should be active. You should now be able to log out of the OpenAM server.
10. Verify the agent URL, in this case, `http://app.example.net`. The OpenAM web agent should now redirect requests to the OpenAM server.

If you want to configure OpenAM and an application on two different cookie domains, such as `example.org` and `example.net`, you will need to set up Cross-Domain SSO (CDSSO). For more information, see the chapter on [Configuring Cross-Domain Single Sign On](#).



---

## Chapter 10

# Configuring Cross-Domain Single Sign On

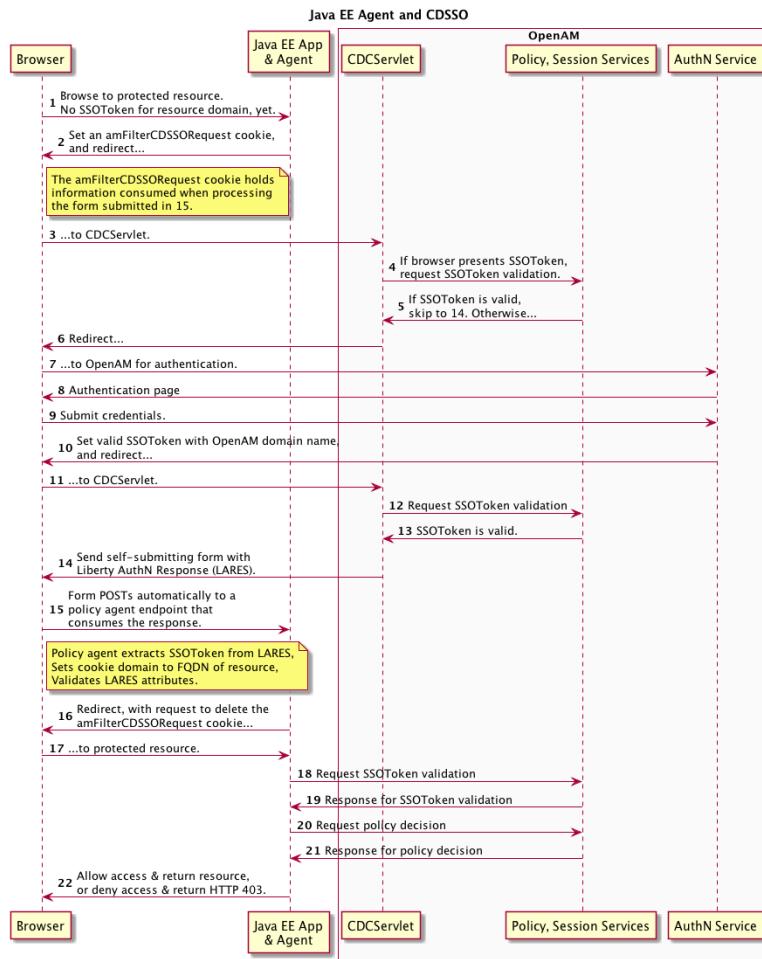
This chapter shows you how to configure cross-domain single sign on (CDSSO). When you have multiple domains in a single organization, CDSSO lets your OpenAM servers in one domain work with policy agents from other domains.

CDSSO is an OpenAM-specific capability. For single sign on across multiple organizations or when integrating with other access management software, use OpenAM's federation capabilities.

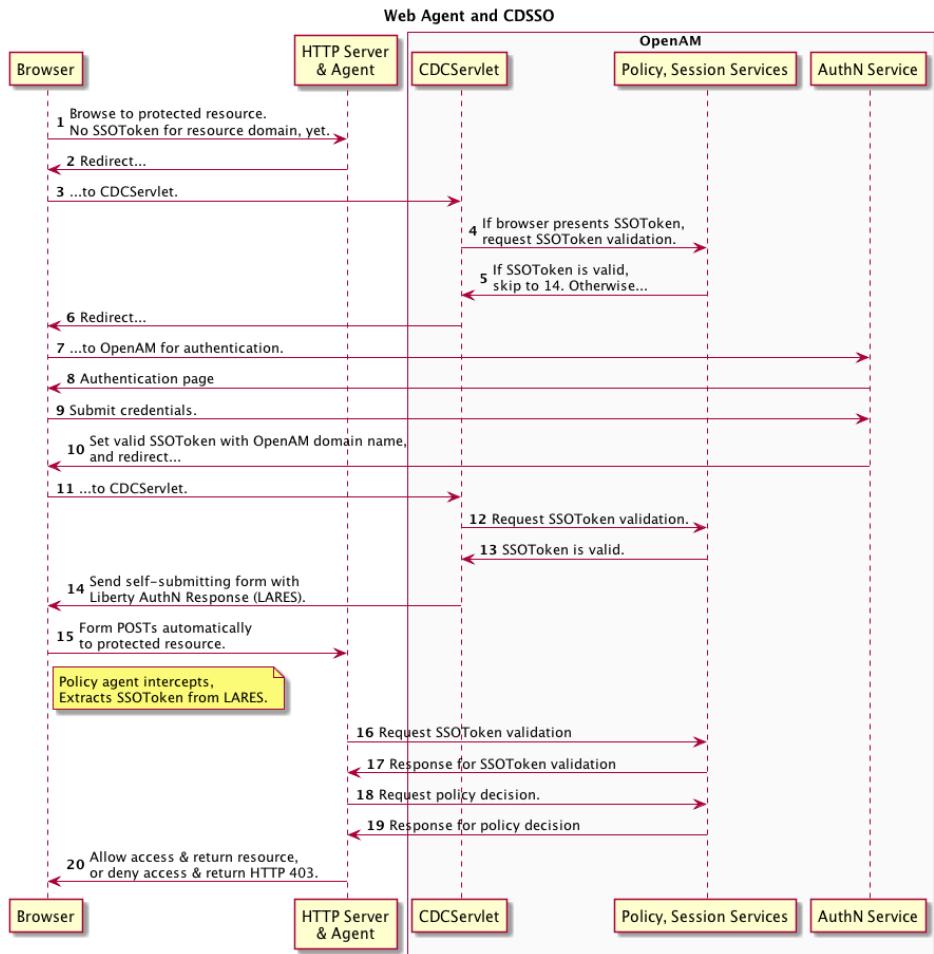
*Cross-domain single sign on* provides a safe mechanism for managing access across multiple different domains that you control. CDSSO lets OpenAM authenticate users redirected by policy agents in other DNS domains.

Single sign on depends on cookies to store session information. Yet for security reasons, browsers do not let a web site in one domain to get access to a cookie from another domain. With CDSSO, the policy agents work around this by negotiating with OpenAM to allow access.

The Java EE policy agent allows CDSSO by using a mechanism to write the SSO token from OpenAM authentication to a cookie with the domain the host where the agent runs. The following sequence diagram illustrates this mechanism.



Whereas the Java EE policy agent has an endpoint specifically to handle the cookie domain translation, the web policy agent handles the request directly as shown in the following sequence diagram.



This chapter includes the following procedures.

- Procedure 10.1, “To Enable CDSSO For a Java EE Policy Agent”
- Procedure 10.2, “To Enable CDSSO For a Web Policy Agent”
- Procedure 10.3, “To Indicate Progress During CDSSO Login”
- Procedure 10.5, “To Protect Against Cookie Hijacking”

The federation mechanism associated with SAML 2.0 can be used as an alternative to CDSSO for both Web and Java EE policy agents. While using SAML 2.0 adds complexity, it supports attribute mapping, which may be useful when

---

the two domains are associated with data stores that use different attribute names. See the section, [SAML 2.0 & Policy Agents](#), for details.

### **Procedure 10.1. To Enable CDSSO For a Java EE Policy Agent**

1. In the OpenAM console, browse to Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > SSO.

2. Scroll down and enable Cross Domain SSO.

3. Check that the CDSSO Redirect URI is set.

Depending on where you deployed your Java EE agent application, the default is something like /agentapp/sunwCDSSORedirectURI.

4. Set the list of URLs for CDSSO Servlet URL to the Cross Domain Controller Servlet URLs of the servers the agent accesses, such as `http://openam.example.com:8080/openam/cdc servlet`.

If the agent accesses OpenAM through a load balancer, use the load balancer URLs, such as `http://load-balancer.example.com:8080/openam/cdc servlet`.

5. Leave the CDSSO Clock Skew set to 0.

Make sure instead that the clocks on the servers where you run OpenAM and policy agents are synchronized.

6. Set the list of URLs for CDSSO Trusted ID Provider to the Cross Domain Controller Servlet URLs of the OpenAM servers the agent accesses, such as `http://openam.example.com:8080/openam/cdc servlet`.

This list should include one CDC Servlet URL for every OpenAM server the agent might access. You do not need to include site or load balancer URLs.

7. To protect the SSO token from network snooping, you can select CDSSO Secure Enable to mark the SSO token cookie as secure.

If you select this, then the SSO token cookie can only be sent over a secure connection (HTTPS).

8. Add the domains involved in CDSSO in the CDSSO Domain List.

9. If necessary, update the Agent Root URL for CDSSO list on the Global tab page.

If the policy agent is on a server with virtual host names, add the virtual host URLs to the list.

---

If the policy agent is behind a load balancer, add the load balancer URL to the list.

10. Save your work.

### **Procedure 10.2. To Enable CDSSO For a Web Policy Agent**

1. In the OpenAM console, browse to Access Control > *Realm Name* > Agents > Web > *Agent Name* > SSO.
2. Enable Cross Domain SSO.
3. Set the list of URLs for CDSSO Servlet URL to the Cross Domain Controller Servlet URLs of the servers the agent accesses, such as `http://openam.example.com:8080/openam/cdc servlet`.  
If the agent accesses OpenAM through a load balancer, use the load balancer URLs, such as `http://load-balancer.example.com:8080/openam/cdc servlet`.
4. Add the domains involved in CDSSO in the Cookies Domain List.
5. If necessary, update the Agent Root URL for CDSSO list on the Global tab page.  
If the policy agent is on a server with virtual host names, add the virtual host URLs to the list.  
If the policy agent is behind a load balancer, add the load balancer URL to the list.
6. Save your work.

### **Procedure 10.3. To Indicate Progress During CDSSO Login**

The default self-submitting form page that OpenAM presents to users contains hidden fields, but is otherwise blank. If you want to show users that the operation is in progress, then customize the necessary JSP.

1. Edit a copy of the file `config/federation/default/cdclogin.jsp` to add a clue that SSO is in progress, such as an image.

You can find this file where you deployed OpenAM, such as `/path/to/tomcat/webapps/openam/config/federation/default/cdclogin.jsp`.

When you add an image or other presentation element, make sure that you retain the form and JavaScript as is.

- 
2. Unpack OpenAM-12.0.0.war, and replace the file with your modified version.  
Also include any images you reference in the page.
  3. Pack up your custom version of OpenAM, and then deploy it in your web container.

#### **Procedure 10.4. To Access the CDSSO Authentication Login**

When a client makes an access request to some protected resource in a cross domain single sign-on (CDSSO) deployment, the policy agent redirects the client to the Cross Domain Controller Servlet (CDCServlet) URL. The CDCServlet determines that the client needs to be authenticated and proxies the request through to an authentication interface, which typically is at /UI/Login:

```
http://openam.example.com:8080/openam/UI/Login
```

If your application requires access to a specific URL, you can use the `loginURI` parameter to do so.

1. For example, you can access the previous authentication UI URL as follows:

```
http://openam.example.com:8080/openam/cdcServlet?loginURI=/UI/Login
```

2. If you have another authentication UI deployed at `/openam/customLoginURI`, you can access this URL at:

```
http://openam.example.com:8080/openam/cdcServlet?loginURI=/customLoginURI
```

In this case, you must also add the custom login URI to the whitelist that is specified by using the `org.forgerock.openam.cdc.validLoginURIs` property.

- a. In OpenAM console, browse to Configuration > Servers and Sites > Default Server Settings > Advanced.
- b. Add a property with the following settings.
  - Property Name: `org.forgerock.openam.cdc.validLoginURIs`
  - Property Value: `/UI/Login,/customLoginURI`
- c. Save your changes.

For more information about this property, see the *Reference* section on advanced properties, [\*Servers > Advanced\*](#).

---

### **Procedure 10.5. To Protect Against Cookie Hijacking**

When cookies are set for an entire domain such as `.example.com`, an attacker who steals a cookie can use it from any host in the domain such as `untrusted.example.com`. Cookie hijacking protection restricts cookies to the fully-qualified domain name (FQDN) of the host where they are issued, such as `openam-server.example.com` and `server-with-agent.example.com`, using CDSSO to handle authentication and authorization.

For CDSSO with cookie hijacking protection, when a client successfully authenticates OpenAM issues the master SSO token cookie for its FQDN. OpenAM issues *restricted token* cookies for the other FQDNs where the policy agents reside. The client ends up with cookies having different session identifiers for different FQDNs, and the OpenAM server stores the correlation between the master SSO token and restricted tokens, such that the client only has one master session internally in OpenAM.

To protect against cookie hijacking you restrict the OpenAM server domain to the server where OpenAM runs. This sets the domain of the SSO token cookie to the host running the OpenAM server that issued the token. You also enable use of a unique SSO token cookie. For your Java EE policy agents, you enable use of the unique SSO token cookie in the agent configuration as well.

1. In the OpenAM console, browse to Configuration > System > Platform.
2. Remove all domains from the Cookies Domains list, or if OpenAM is behind a load balancer, use the load balancer host name such as `load-balancer.example.com`.
3. Save your work.
4. In the OpenAM console, browse to Configuration > Servers and Sites > Default Server Settings > Advanced, and then make the necessary changes.
  - a. Change the setting for the property `com.sun.identity.enableUniqueSSOTokenCookie` to true, from the default false.
  - b. Make sure that the property `com.sun.identity.authentication.uniqueCookieName` is set to the name of the cookie that will hold the URL to the OpenAM server that authenticated the user.

The default name is `sunIdentityServerAuthNServer`.

Save your work.

5. Browse to Configuration > Servers and Sites > *Server Name* > Advanced, and add the property `com.sun.identity.authentication.uniqueCookieDomain`,

---

setting the value to the fully-qualified domain name of the current OpenAM server, such as `openam.example.com`.

Then Save your work.

6. For each Java EE policy agent, browse in the OpenAM console to Access Control > *Realm Name* > Agents > J2EE > *Agent Name* > Advanced > Custom Properties, and add `com.sun.identity.enableUniqueSSOTokenCookie=true` to the list.
7. Save your work.
8. Restart OpenAM or the container in which it runs for the configuration changes to take effect.

---

## Chapter 11

# Managing SAML 2.0 Federation

This chapter addresses how to set up and manage SAML 2.0 SSO for single sign on and single log out across resources belonging to organizations participating in a circle of trust.

## 11.1 About SAML 2.0 SSO & Federation

SAML 2.0 SSO is part of federated access management. Federation lets access management cross organizational boundaries. Federation helps organizations share identities and services without giving away their identity information, or the services they provide.

To bridge heterogeneous systems, federation requires interoperability, and thus depends on standards for orchestrating interaction and exchanging information between providers. OpenAM federation relies on standards such as [Security Assertion Markup Language \(SAML\) 2.0](#). SAML 2.0 describes the messages, how they are relayed, how they are exchanged, and common use cases.

To achieve SAML 2.0 SSO, OpenAM separates *identity providers* from *service providers*, lets you include them in a *circle of trust*, and has you configure how the providers in the circle of trust interact.

- An identity provider stores and serves identity profiles, and handles authentication.
- A service provider offers services that access protected resources, and handles authorization.

- A circle of trust groups at least one identity provider and at least one service provider who agree to share authentication information, with assertions about authenticated users that let service providers make authorization decisions.

Providers in a circle of trust share *metadata*, configuration information that federation partners require to access each others' services.

- SAML 2.0 SSO maps attributes from accounts at the identity provider to attributes on accounts at the service provider. The identity provider makes assertions to the service provider, for example to attest that a user has authenticated with the identity provider. The service provider then consumes assertions from the identity provider to make authorization decisions, for example to let an authenticated user complete a purchase that gets charged to the user's account at the identity provider.

In federation deployments where not all providers support SAML 2.0, OpenAM can act as a multi-protocol hub, translating for providers who rely on other and older standards such as SAML 1.x, Liberty Alliance Project frameworks, and WS-Federation (for integration with Active Directory Federation Services, for example).

## 11.2 Setting Up SAML 2.0 SSO

Before you set up SAML 2.0 SSO in OpenAM, you must:

- Know which providers participate in the circle of trust.
- Know how OpenAM installations act as identity providers, or service providers.
- Agree with other providers on a synchronized time service.
- For identity information exchanged with other participants in a circle of trust, define how to map shared user attributes. Local user profile attribute names should map to user profile attribute names at other providers.

For example, if you exchange user identifiers with your partners, and you call it uid whereas another partner calls it userid, then you map your uid to your partner's userid.

- Import the keys used to sign assertions into the JKS key store in your OpenAM configuration directory. You can use the Java **keytool** command.

The OpenAM configuration key store is located at the top level of the configuration directory, such as \$HOME/openam/keystore.jks. The password, stored in \$HOME/openam/.keypass, is changeit by default. Also by default the only key available is for a self-signed certificate (alias: test) installed with OpenAM.

During set up, you must share metadata for providers that you host with other providers in the circle of trust. You must also configure remote providers, connecting to other providers by importing their metadata.

In OpenAM terms, a hosted provider is one served by the current OpenAM server, whereas a remote provider is one hosted elsewhere.

This section covers the following topics.

- [Procedure 11.1, “To Create a Hosted Identity Provider”](#)
- [Procedure 11.2, “To Create a Hosted Service Provider”](#)
- [Procedure 11.3, “To Create a Remote Identity Provider”](#)
- [Procedure 11.4, “To Create a Remote Service Provider”](#)
- [Procedure 11.5, “To Create a Fedlet for Service Providers”](#)
- [Section 11.2.1, “Deploying the Identity Provider Discovery Service”](#)

### **Procedure 11.1. To Create a Hosted Identity Provider**

1. On the OpenAM console Common Tasks page, click Create Hosted Identity Provider.
2. Unless you already have metadata for the provider, accept the Name for this identity provider in the field provided, or provide your own unique identifier.  
The default name is the URL to the current server which hosts the identity provider.
3. Select the Signing Key you imported into the OpenAM key store.
4. Either add the provider to the circle of trust you already created, or select Add to new and provide a New Circle of Trust name.
5. For the attributes you share, map service provider attribute names (Name in Assertion), to user profile names from your identity repository (Local Attribute Name).

Use this approach to set up a mapping with all SPs in the Circle of Trust that do not have their own specific mappings configured.

The default mapping implementation has additional features beyond simply retrieving string attributes from the user profile.

- Add an attribute that takes a static value by enclosing the profile attribute name in double quotes (").

For example, you can add a static SAML attribute called `partnerID` with a value of `staticPartnerIDValue` by adding `partnerID` as the Name in Assertion with "staticPartnerIDValue" as the Local Attribute Name.

- Base64 encode binary attributes when adding them to the SAML attributes by adding ;binary to the end of the attribute name, as in the following example:

```
objectGUID=objectGUID;binary
```

This maps the local binary attribute `objectGUID` to a SAML attribute called `objectGUID` that is Base64 encoded.

- Use `NameFormatURI` format as shown in the following example:

```
urn:oasis:names:tc:SAML:2.0:attrname-format:uri|objectGUID=objectGUID;binary
```

6. Click Configure to save your configuration.
7. Export the XML-based metadata from your provider to share with other providers in your circle of trust.

```
$ curl \
--output metadata.xml \
"http://www.idp.example:8080/openam/saml2/jsp/exportmetadata.jsp"
?entityid=
http://www.idp.example:8080/openam&realm=/realm-name"
```

When you have configured only the top-level realm, `/`, you can omit the query string.

Alternatively, provide the URL, to other providers so they can load the metadata.

### Procedure 11.2. To Create a Hosted Service Provider

1. On the OpenAM console Common Tasks page, click Create Hosted Service Provider.
2. Unless you already have metadata for the provider, accept the Name for this service provider in the field provided, or provide your own unique identifier.

The default name is the URL to the current server which hosts the service provider.

3. Either add the provider to the circle of trust you already created, or select Add to new and provide a New Circle of Trust name.
4. If the identity provider has not already mapped the attributes you share, map identity provider attribute names (Name in Assertion), to user profile names from your identity repository (Local Attribute Name).
5. Click Configure to save your configuration.
6. Export the XML-based metadata from your provider to share with other providers in your circle of trust.

```
$ curl \
--output metadata.xml \
"http://www.sp.example:8080/openam/saml2/jsp/exportmetadata.jsp
?entityid=
http://www.sp.example:8080/openam&realm=/realm-name"
```

When you have configured only the top-level realm, `/`, you can omit the query string.

Alternatively, provide the URL, to other providers so they can load the metadata.

#### **Procedure 11.3. To Create a Remote Identity Provider**

1. Obtain the identity provider metadata, or the URL where you can obtain it.
2. On the OpenAM console Common Tasks page, click Register Remote Identity Provider.
3. Provide the identity provider metadata or link to obtain metadata.
4. Either add the provider to the circle of trust you already created, or select Add to new and provide a New Circle of Trust name.
5. Click Configure to save your configuration.

#### **Procedure 11.4. To Create a Remote Service Provider**

1. Obtain the service provider metadata, or the URL where you can obtain it.
2. On the OpenAM console Common Tasks page, click Register Remote Service Provider.

3. Provide the identity provider metadata or link to obtain metadata.
4. If the identity provider has not already mapped the attributes you share, map identity provider attribute names (Name in Assertion), to user profile names from your identity repository (Local Attribute Name).

Use this approach to set up a mapping that is specific to this SP.

The default mapping implementation has additional features beyond simply retrieving string attributes from the user profile.

- Add an attribute that takes a static value by enclosing the profile attribute name in double quotes ("").

For example, you can add a static SAML attribute called partnerID with a value of staticPartnerIDValue by adding partnerID as the Name in Assertion with "staticPartnerIDValue" as the Local Attribute Name.

- Base64 encode binary attributes when adding them to the SAML attributes by adding ;binary to the end of the attribute name, as in the following example:

```
objectGUID=objectGUID;binary
```

This maps the local binary attribute objectGUID to a SAML attribute called objectGUID that is Base64 encoded.

- Use NameFormatURI format as shown in the following example:

```
urn:oasis:names:tc:SAML:2.0:attrname-format:uri|objectGUID=objectGUID;binary
```

5. Either add the provider to the circle of trust you already created, or select Add to new and provide a New Circle of Trust name.
6. Click Configure to save your configuration.

#### **Procedure 11.5. To Create a Fedlet for Service Providers**

When your organization acts as the identity provider, and you want quickly to enable service providers to federate their services with yours, you can provide them with a *fedlet*. A fedlet is a small Java or .NET web application that can act as a service provider for a specific identity provider without requiring that you install all of OpenAM.

Fedlets support the following SAML 2.0 features.

**Table 11.1. Fedlet Support for SAML 2.0 Features**

| SAML 2.0 Feature                                  | Java Fedlet | .NET Fedlet   |
|---|-------------|---------------|
| IDP & SP-initiated Single Sign-On (HTTP Artifact) | Supported   | Supported     |
| IDP & SP-initiated Single Sign-On (HTTP POST)     | Supported   | Supported     |
| IDP & SP-initiated Single Logout (HTTP POST)      | Supported   | Supported     |
| IDP & SP-initiated Single Logout (HTTP Redirect)  | Supported   | Supported     |
| Sign Requests & Responses                         | Supported   | Supported     |
| Encrypt Assertion, Attribute, & NameID Elements   | Supported   | Supported     |
| Export SP Metadata                                | Supported   | Supported     |
| Attribute Queries                                 | Supported   | Supported     |
| XACML Requests                                    | Supported   | Not supported |
| Multiple IDPs                                     | Supported   | Supported     |
| External IDP Discovery Service                    | Supported   | Supported     |
| Bundled IDP Reader Service for Discovery          | Supported   | Not supported |

For more information on using fedlets, see [Using Fedlets in Java Web Applications](#) and [Using Fedlets in .NET Applications](#) in the *Developer's Guide*.

The following procedure describes how to create a Java Fedlet.

1. If you have not done so already, set up your identity provider.
2. Enter the URL where the service provider will deploy the fedlet you create, and name the fedlet. If you create multiple fedlets, use the URL as a unique name that shows who has deployed the fedlet.
3. For the attributes you share, map service provider attribute names (Name in Assertion), to user profile names from your identity repository (Local Attribute Name).
4. Click Create to generate the `Fedlet.zip` file under the OpenAM configuration directory, such as `$HOME/openam/myfedlets/httpwwwexamplecom80myapp/Fedlet.zip`.
5. Give the `Fedlet.zip` file to the service provider for deployment.

### 11.2.1 Deploying the Identity Provider Discovery Service

When your circle of trust includes multiple identity providers, then service providers must discover which identity provider corresponds to a request. You can deploy the identity provider discovery service for this purpose as a separate web application.

Browsers only send cookies for the originating domain. Therefore when a browser accesses the service provider in the `www.sp.example` domain, the service provider has no way of knowing whether the user has perhaps already authenticated at `www.this-idp.example` or at `www.that-idp.example`. The providers therefore host an identity provider discovery service in a common domain, such as `www.disco.example`, and use that service to discover where the user logged in. The identity provider discover service essentially writes and reads cookies from the common domain. The providers configure their circle of trust to use the identity provider discovery service as part of SAML 2.0 federation.

Deploying the identity provider discovery service involves the following stages.

1. Deploy the .war file into your web application container.
2. Configure the discovery service.
3. Add the identity provider discovery service endpoints for writing cookies to and reading cookies from the common domain to the circle of trust configurations for the providers.
4. Share metadata between identity providers and the service provider.

#### Procedure 11.6. To Deploy the Discovery Service on Tomcat

How you deploy the discovery service .war file depends on your web application container. The procedure in this section shows how to deploy on Apache Tomcat.

1. Copy the `IDPDiscovery-12.0.0.war` file to the `webapps/` directory.

```
$ cp ~/Downloads/openam/IDPDiscovery-12.0.0.war \
/path/to/tomcat/webapps/disco.war
```

2. Access the configuration screen through your browser.

In this example, Apache Tomcat listens for HTTP requests on `www.disco.example:8080`, and Tomcat has unpacked the application under `/disco`, so the URL is `http://www.disco.example:8080/disco`, which redirects to `Configurator.jsp`.

### Procedure 11.7. To Configure the Discovery Service

1. Configure the identity provider discovery service.

#### Configuring IDP Discovery Service

Please provide the IDP Discovery service information

|   |   |
|---|---|
| Debug directory:  | <input type="text" value="/tmp/debug"/>                                   |
| Debug Level:  | <input type="button" value="error"/>                                      |
| Cookie Type:  | <input checked="" type="radio"/> PERSISTENT <input type="radio"/> SESSION |
| Cookie Domain:  | <input type="text" value=".disco.example"/>                               |
| Secure Cookie:  | <input type="radio"/> True <input checked="" type="radio"/> False         |
| Encode Cookie:  | <input checked="" type="radio"/> True <input type="radio"/> False         |
| HTTP-Only Cookie:   | <input type="radio"/> True <input checked="" type="radio"/> False         |
| <input type="button" value="Configure"/> <input type="button" value="Reset"/> |   |

Hints for discovery service configuration parameters follow.

##### Debug Directory

The discovery service logs to flat files in this directory.

##### Debug Level

Default is `error`. Other options include `error`, `warning`, `message`, and `off`.

Set this to `message` in order to see the service working when you run your initial tests.

##### Cookie Type

Set to `PERSISTENT` if you have configured OpenAM to use persistent cookies, meaning single sign on cookies that can continue to be valid after the browser is closed.

##### Cookie Domain

The cookie domain is the common cookie domain used in your circle of trust for identity provider discovery, in this case `.disco.example`.

##### Secure Cookie

Set this to true if clients should only return cookies when a secure connection is used.

#### Encode Cookie

Leave this true unless your OpenAM installation requires that you do not encode cookies. Normally cookies are encoded such that cookies remain valid in HTTP.

#### HTTP-Only Cookie

Set to true to use HTTPOnly cookies if needed to help prevent third-party programs and scripts from accessing the cookies.

2. Restrict permissions to the discovery service configuration file in `$HOME/libIDPDiscoveryConfig.properties`, where `$HOME` corresponds to the user who runs the web container where you deployed the service.

### **Procedure 11.8. To Add the Discovery Service to Your Circles of Trust**

Each provider has a circle of trust including itself. You configure each of these circles of trust to use the identity provider discovery service as described in the following steps.

1. On the service provider console, login as OpenAM Administrator.
2. On the service provider console, under Federation > Circle of Trust > *Circle of Trust Name* add SAML2 Writer and Reader Service URLs for the identity provider discovery service endpoints, and Save your work.  
  
In this example, the writer URL is `http://www.disco.example:8080/disco/saml2writer`, and the reader URL is `http://www.disco.example:8080/disco/saml2reader`.
3. On each identity provider console, login as OpenAM Administrator.
4. On the identity provider console, under Federation > Circle of Trust Configuration > *Circle of Trust Name* also add SAML2 Writer and Reader Service URLs for the identity provider discovery service endpoints, and Save your work.

### **Procedure 11.9. To Share Identity & Service Provider Metadata**

Before performing these steps, install the administration tools for each provider as described in [To Set Up Administration Tools](#). The administration tools include the **ssoadm** tool that you need to export metadata.

1. On each identity provider console, register the service provider as a remote service provider adding to the circle of trust you configured to use the identity provider discovery service.

The URL to the service provider metadata is something like `http://www.sp.example:8080/openam/saml2/jsp/exportmetadata.jsp`.

2. Obtain metadata for each identity provider.

```
$ ssh www.this-idp.example
$ cd /path/to/openam-tools/admin/openam/bin
$ ./ssoadm \
  create-metadata-temp1 \
  --entityid "http://www.this-idp.example:8080/openam" \
  --adminid amadmin \
  --password-file /tmp/pwd.txt \
  --identityprovider /idp \
  --meta-data-file this-standard.xml \
  --extended-data-file this-extended.xml
Hosted entity configuration was written to this-extended.xml.
Hosted entity descriptor was written to this-standard.xml.

$ ssh www.that-idp.example
$ cd /path/to/openam-tools/admin/openam/bin
$ ./ssoadm \
  create-metadata-temp1 \
  --entityid "http://www.that-idp.example:8080/openam" \
  --adminid amadmin \
  --password-file /tmp/pwd.txt \
  --identityprovider /idp \
  --meta-data-file that-standard.xml \
  --extended-data-file that-extended.xml

Hosted entity configuration was written to that-extended.xml.
Hosted entity descriptor was written to that-standard.xml.
```

3. For each identity provider extended metadata file, change the value of the `hosted` attribute to `0`, meaning the identity provider is remote.
4. On the service provider, add the identity providers to the circle of trust using the identity provider metadata.

```
$ ssh www.sp.example
$ cd /path/to/openam-tools/admin/openam/bin
$ ./ssoadm \
  import-entity \
  --cot discocot \
  --meta-data-file ~/Downloads/this-standard.xml \
  --extended-data-file ~/Downloads/this-extended.xml \
  --adminid amadmin \
  --password-file /tmp/pwd.txt

Import file, /Users/mark/Downloads/this-standard.xml.
Import file, /Users/mark/Downloads/this-extended.xml.
$ ./ssoadm \
  import-entity \
  --cot discocot \
  --meta-data-file ~/Downloads/that-standard.xml \
```

## Deploying the Identity Provider Discovery Service

```
--extended-data-file ~/Downloads/that-extended.xml \
--adminid amadmin \
--password-file /tmp/pwd.txt

Import file, /Users/mark/Downloads/that-standard.xml.
Import file, /Users/mark/Downloads/that-extended.xml.
```

5. Test your work by using the Federation Connectivity Test that you start from the service provider console under Common Tasks > Test Federation Connectivity.

When the test is done, you can see messages from the CookieWriterServlet in the libIDPDiscovery log file where you set up logging when you configured the identity provider discovery service, such as /tmp/debug/libIDPDiscovery. Output generated during a test follows, with some lines folded to fit on the printed page.

```
08/08/2012 11:43:38:341 AM CEST: Thread[http-bio-8080-exec-4,5,main]
CookieUtils.init : idpDiscoveryOnlyWar=true
08/08/2012 11:43:38:341 AM CEST: Thread[http-bio-8080-exec-4,5,main]
CookieWriterServlet Initializing...
08/08/2012 11:43:38:341 AM CEST: Thread[http-bio-8080-exec-4,5,main]
CookieWriterServlet doGetPost: Preferred Cookie Name is _saml_idp
08/08/2012 11:43:38:341 AM CEST: Thread[http-bio-8080-exec-4,5,main]
CookieWriterServlet doGetPost: URL Scheme is null, set to https.
08/08/2012 11:43:38:341 AM CEST: Thread[http-bio-8080-exec-4,5,main]
CookieWriterServlet doGetPost: Preferred IDP Cookie Not found
08/08/2012 11:43:38:342 AM CEST: Thread[http-bio-8080-exec-4,5,main]
CookieWriterServlet doGetPost: Cookie Type is PERSISTENT
08/08/2012 11:43:38:342 AM CEST: Thread[http-bio-8080-exec-4,5,main]
CookieWriterServlet doGetPost: Cookie value is
aHR0cDovL3d3dy50aGF0LWlkC5jb2060DA4MC9vcGVuYW0=
08/08/2012 11:43:38:342 AM CEST: Thread[http-bio-8080-exec-4,5,main]
CookieWriterServlet doGetPost: Preferred Cookie Name _saml_idp
08/08/2012 11:43:38:343 AM CEST: Thread[http-bio-8080-exec-4,5,main]
CookieWriterServlet doGetPost: Redirect to
http://www.that-idp.example:8080/openam/SSORedirect/metaAlias/idp?resInfoID=
s28bc4db004f1365d78d07d69846c54a3c850fe801
08/08/2012 11:43:46:957 AM CEST: Thread[http-bio-8080-exec-4,5,main]
CookieWriterServlet doGetPost: Preferred Cookie Name is _saml_idp
08/08/2012 11:43:46:957 AM CEST: Thread[http-bio-8080-exec-4,5,main]
CookieUtils:cookieValue=aHR0cDovL3d3dy50aGF0LWlkC5jb2060DA4MC9vcGVuYW0=,
result=aHR0cDovL3d3dy50aGF0LWlkC5jb2060DA4MC9vcGVuYW0=
08/08/2012 11:43:46:957 AM CEST: Thread[http-bio-8080-exec-4,5,main]
CookieWriterServlet doGetPost: Cookie Type is PERSISTENT
08/08/2012 11:43:46:957 AM CEST: Thread[http-bio-8080-exec-4,5,main]
CookieWriterServlet doGetPost: Cookie value is
aHR0cDovL3d3dy50aGF0LWlkC5jb2060DA4MC9vcGVuYW0=
08/08/2012 11:43:46:957 AM CEST: Thread[http-bio-8080-exec-4,5,main]
CookieWriterServlet doGetPost: Preferred Cookie Name _saml_idp
08/08/2012 11:43:46:957 AM CEST: Thread[http-bio-8080-exec-4,5,main]
CookieWriterServlet doGetPost: Redirect to
http://www.that-idp.example:8080/openam/SSORedirect/metaAlias/idp?resInfoID=
s2ce9c465cf39c96f31e1dcf009cf9943695d82901
```

## 11.3 Configuring Identity Providers

Once you have set up an identity provider, you can configure it through the OpenAM console under Federation > Entity Providers > *Provider Name*.

### 11.3.1 Hints for Assertion Content

Use the following hints to adjust settings on the Assertion Content tab page.

#### Signing and Encryption

##### Request/Response Signing

Specifies what parts of messages the identity provider requires the service provider to sign digitally.

##### Encryption

When selected, the service provider must encrypt NameID elements.

##### Certificate Aliases

Specifies aliases for certificates in the OpenAM key store that are used to handle digital signatures, and to handle encrypted messages.

Specify a Key Pass if the private key password is different from the key store password, which is stored encrypted in the .keypass file for the server. For instructions on working with key pairs, also see [To Change the Signing Key for Federation](#).

#### NameID Format

##### NameID Format List

Specifies the supported name identifiers for users that are shared between providers for single sign on. If no name identifier is specified when initiating single sign on, then the identity provider uses the first one in the list.

##### NameID Value List

Maps name identifier formats to user profile attributes. The persistent and transient name identifiers need not be mapped.

#### Authentication Context

##### Mapper

Specifies a class that implements the IDPAuthnContextMapper interface and sets up the authentication context.

**Default Authentication Context**

Specifies the authentication context used if no authentication context specified in the request.

**Supported Contexts**

Specifies the supported authentication contexts, where the Key and Value can specify a corresponding OpenAM authentication method, and the Level corresponds to an authentication module authentication level.

**Assertion Time**

**Not-Before Time Skew**

Grace period in seconds for the `NotBefore` time in assertions.

**Effective Time**

Validity in seconds of an assertion.

**Basic Authentication**

**Enabled, User Name, Password**

When enabled, authenticate with the specified user name and password at SOAP end points.

**Assertion Cache**

**Enabled**

When enabled, cache assertions.

### **11.3.2 Hints for Assertion Processing**

Use the following hints to adjust settings on the Assertion Processing tab page.

**Attribute Mapper**

**Attribute Mapper**

Specifies a class that implements the attribute mapping.

The default implementation attempts to retrieve the mapped attribute values from the user profile first. If the attribute values are not present in the user's profile, then it attempts to retrieve them from the user's session.

Default: `com.sun.identity.saml2.plugins.DefaultIDPAttributeMapper`

**Attribute Map**

Maps SAML attributes to user profile attributes.

The user profile attributes used here must both be allowed in user profiles, and also be specified for the identity repository. See the *Developer's Guide* chapter, [Customizing Profile Attributes](#), for instructions on allowing additional attributes in user profiles.

To specify the list of profile attributes for an LDAP identity repository, login to OpenAM Console as administrator and browse to Access Control > *Realm Name* > Data Stores, and click the data store name to open the configuration page. Scroll down to User Configuration, and edit the LDAP User Attributes list, and then click Save to keep your work.

The default IDP mapping implementation allows you to add static values in addition to values taken from the user profile. You add a static value by enclosing the profile attribute name in double quotes ("), as in the following examples.

To add a static SAML attribute called `nameID` with a value of `staticNameIDValue` with a name format of `urn:oasis:names:tc:SAML:2.0:attrname-format:uri`, add the following mapping.

```
urn:oasis:names:tc:SAML:2.0:attrname-format:uri|nameID="staticNameIDValue"
```

## Account Mapper

### Account Mapper

Specifies a class that implements `AccountMapper` to map remote users to local user profiles.

## Local Configuration

### Auth URL

URL where users are redirected to authenticate.

### Reverse Proxy URL

When a reverse proxy is used for SAML endpoints, it is specified here.

### External Application Logout URL

URL to which to send an HTTP POST including all cookies when receiving a logout request. To add a user session property as a POST parameter, include it in the URL query string as a `appsessionproperty` parameter.

### 11.3.3 Hints for Services

Use the following hints to adjust settings on the Services tab page.

## **MetaAlias**

### MetaAlias

Used to locate the providers entity identifier, specified as `[/realm-name]*/provider-name`, where neither `realm-name` nor `provider-name` can contain slash characters (/). For example: /myRealm/mySubrealm/idp.

## **IDP Service Attributes**

### Artifact Resolution Service

Specifies the end point to handle artifact resolution. The Index is a unique number identifier for the end point.

### Single Logout Service

Specifies the end points to handle single logout, depending on the SAML binding selected.

### Manage NameID Service

Specifies the end points to handle name identifiers, depending on the SAML binding selected.

### Single SignOn Service

Specifies the end points to handle single sign on.

## **NameID Mapping**

### URL

Specifies the end point to handle name identifier mapping.

### **11.3.4 Hints for Advanced Settings**

Use the following hints to adjust settings on the Advanced tab page.

## **SAE Configuration**

### IDP URL

Specifies the end point to handle Secure Attribute Exchange requests.

### Application Security Configuration

Specifies how to handle encryption for Secure Attribute Exchange operations.

## ECP Configuration

### IDP Session Mapper

Specifies the class that finds a valid session from an HTTP servlet request to an identity provider with a SAML Enhanced Client or Proxy profile.

## Session Synchronization

### Enabled

When enabled, the identity provider notifies service providers to log the user out when a session expires.

## IDP Finder Implementation

### IDP Finder Implementation Class

Specifies a class that finds the preferred identity provider to handle a proxied authentication request.

### IDP Finder JSP

Specifies a JSP that presents the list of identity providers to the user.

### Enable Proxy IDP Finder For All SPs

When enabled, apply the finder for all remote service providers.

## Relay State URL List

### Relay State URL List

List of URLs to which to redirect users after the request has been handled.  
Used if not specified in the service provider extended metadata.

## IDP Adapter

### IDP Adapter Class

Specifies a class to invoke immediately before sending a SAML 2.0 response.

## 11.4 Configuring Service Providers

Once you have set up a service provider, you can configure it through the OpenAM console under Federation > Entity Providers > *Provider Name*.

### 11.4.1 Hints for Assertion Content

Use the following hints to adjust settings on the Assertion Content tab page.

#### Signing and Encryption

##### Request/Response Signing

Specifies what parts of messages the service provider requires the identity provider to sign digitally.

##### Encryption

The identity provider must encrypt selected elements.

##### Certificate Aliases

Specifies aliases for certificates in the OpenAM key store that are used to handle digital signatures, and to handle encrypted messages.

#### NameID Format

##### NameID Format List

Specifies the supported name identifiers for users that are shared between providers for single sign on. If no name identifier is specified when initiating single sign on, then the service provider uses the first one in the list supported by the identity provider.

##### Disable Federation persistence if NameID Format is unspecified

When enabled, the NameID Format in the authentication response is `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`, and the Account Mapper has identified the local user, the service provider does not persist federation information in the user profile.

#### Authentication Context

##### Mapper

Specifies a class that implements the `SPAuthnContextMapper` interface and sets up the authentication context.

##### Default Authentication Context

Specifies the authentication context used if no authentication context specified in the request.

##### Supported Contexts

Specifies the supported authentication contexts. The Level corresponds to an authentication module authentication level.

#### Comparison Type

How the authentication context in the assertion response must compare to the supported contexts.

#### Assertion Time

##### Assertion Time Skew

Grace period in seconds for the `NotBefore` time in assertions.

#### Basic Authentication

##### Enabled, User Name, Password

When enabled, authenticate with the specified user name and password at SOAP end points.

## 11.4.2 Hints for Assertion Processing

Use the following hints to adjust settings on the Assertion Processing tab page.

#### Attribute Mapper

##### Attribute Mapper

Specifies a class that implements the attribute mapping.

##### Attribute Map

Maps SAML attributes to user profile attributes.

#### Auto Federation

##### Enabled

When enabled, automatically federate user's accounts at different providers based on the specified profile attribute.

##### Attribute

Specifies the user profile attribute to match accounts at different providers.

#### Account Mapper

##### Account Mapper

Specifies a class that implements `AccountMapper` to map remote users to local user profiles.

**Use Name ID as User ID**

When selected, fall back to using the name identifier from the assertion to find the user.

**Artifact Message Encoding**

**Encoding**

Specifies the message encoding format for artifacts.

**Transient User**

**Transient User**

Specifies the user profile to which to map all identity provider users when sending transient name identifiers.

**URL**

**Local Authentication URL**

Specifies the local login URL.

**Intermediate URL**

Specifies a URL to which the user is redirected after authentication but before the original URL requested.

**External Application Logout URL**

Specifies the URL to which to send an HTTP POST including all cookies when receiving a logout request. To add a user session property as a POST parameter, include it in the URL query string as a `appsessionproperty` parameter.

**Default Relay State URL**

**Default Relay State URL**

Specifies the URL to which to redirect users after the request has been handled. Used if not specified in the response.

**Adapter**

**Adapter**

Specifies a class that implements the `FederationSPAdapter` interface and performs application specific processing during the federation process.

**Adapter Environment**

Specifies environment variables passed to the adapter class.

### 11.4.3 Hints for Services

Use the following hints to adjust settings on the Services tab page.

#### MetaAlias

##### MetaAlias

Used to locate the providers entity identifier, specified as `[/realm-name]*[/provider-name]`, where neither `realm-name` nor `provider-name` can contain slash characters (/). For example: `/myRealm/mySubrealm/sp`.

#### SP Service Attributes

##### Single Logout Service

Specifies the end points to handle single logout, depending on the SAML binding selected.

##### Manage NameID Service

Specifies the end points to handle name identifiers, depending on the SAML binding selected.

##### Assertion Consumer Service

Specifies the end points to consume assertions, with Index corresponding to the index of the URL in the standard metadata.

### 11.4.4 Hints for Advanced Settings

Use the following hints to adjust settings on the Advanced tab page.

#### SAE Configuration

##### SP URL

Specifies the end point to handle Secure Attribute Exchange requests.

##### SP Logout URL

Specifies the end point of the service provider that can handle global logout requests.

##### Application Security Configuration

Specifies how to handle encryption for Secure Attribute Exchange operations.

## ECP Configuration

### Request IDP List Finder Implementation

Specifies a class that returns a list of preferred identity providers trusted by the SAML Enhanced Client or Proxy profile.

### Request IDP List Get Complete

Specifies a URI reference used to retrieve the complete identity provider list if the IDPLIST element is not complete.

### Request IDP List

Specifies a list of identity providers for the SAML Enhanced Client or Proxy to contact, used by the default implementation of the IDP Finder.

## IDP Proxy

### IDP Proxy

When enabled, allow proxied authentication for this service provider.

### Introduction

When enabled, use introductions to find the proxy identity provider.

### Proxy Count

Specifies the maximum number of proxy identity providers.

### IDP Proxy List

Specifies a list of URIs identifying preferred proxy identity providers.

## Session Synchronization

### Enabled

When enabled, the service provider notifies identity providers to log the user out when a session expires.

## Relay State URL List

### Relay State URL List

List of URLs to which to redirect users after the request has been handled.  
Used if not specified in the service provider extended metadata.

## 11.5 Configuring Circles of Trust

Once you have set up a circle of trust, you can configure it through the OpenAM console under Federation > Circle of Trust > *Circle of Trust Name*.

**Name**

String to refer to the circle of trust.

**Description**

Short description of the circle of trust.

**IDFF Writer Service URL**

Liberty Identity Federation Framework service that writes identity provider entity identifiers to Common Domain cookies after successful authentication, used in identity provider discovery. Example: `http://www.disco.example:8080/openam/idffwriter`.

**IDFF Reader Service URL**

Liberty Identity Federation Framework service that reads identity provider entity identifiers from Common Domain cookies, used in identity provider discovery. Example: `http://www.disco.example:8080/openam/transfer`.

**SAML2 Writer Service URL**

SAML 2.0 service that writes identity provider entity identifiers to Common Domain cookies after successful authentication, used in identity provider discovery. Example: `http://www.disco.example:8080/openam/saml2writer`.

**SAML2 Reader Service URL**

SAML 2.0 service that reads identity provider entity identifiers from Common Domain cookies, used in identity provider discovery. Example: `http://www.disco.example:8080/openam/saml2reader`.

**Status**

Whether this circle of trust is operational.

**Realm**

Name of the realm participating in this circle of trust.

**Entity Providers**

Known hosted and remote identity and service providers participating in this circle of trust.

## 11.6 Configuring Providers for Failover

OpenAM servers can function in a site configuration behind a load balancer, as described in the *Installation Guide* chapter, [Setting Up OpenAM Session Failover](#).

When you set up the same SAML 2.0 Provider on multiple servers behind a load balancer, however, you must ensure the metadata points to the load balancer rather than the individual servers.

1. Before configuring the provider, follow the instructions in the *Installation Guide* mentioned above, and make sure that failover works through the load balancer for normal OpenAM sessions.
2. Configure the provider on one of the servers using the load balancer URL as the entity ID.
3. Export the metadata and extended metadata for the provider. You can export metadata either by using the **ssoadm** command, or by using the **ssoadm.jsp** page in OpenAM console after setting **ssoadm.disabled** to false under Servers and Sites > *Server Name* > Advanced.

With the **ssoadm** command, you can export the metadata as shown in the following example for an Identity Provider, where the entity ID is `http://lb.example.com:80/openam`.

```
$ ssoadm \
  export-entity \
  --entityid "http://lb.example.com:80/openam" \
  --adminid amadmin \
  --password-file /tmp/pwd.txt \
  --meta-data-file idp.xml \
  --extended-data-file idp-extended.xml
```

4. Edit both the metadata and the extended metadata, changing all URLs in both files to use the load balancer URL.
5. Delete the provider configuration in OpenAM Console.
6. Import the edited provider configuration in OpenAM Console.
7. Enable SAMLv2 failover in OpenAM Console.

Under Configuration > Global, click SAMLv2 Service Configuration.

Select Enabled next to Enable SAMLv2 failover, and then click Save.

At this point failover is operational for the provider you configured.

## 11.7 Configuring Google Apps as a Remote Service Provider

OpenAM can serve as the identity provider when you use [Google Apps](#) as a service provider, allowing users to have single sign-on with their Google Apps account.

In order to use this service, you must have a Google Apps account for at least one of your domains, such as `example.com`.

### Procedure 11.10. To Integrate With Google Apps

1. If you have not yet done so, set up OpenAM as described in [Procedure 11.1, "To Create a Hosted Identity Provider"](#), using a signing certificate that is needed by Google Apps.

See the procedure [\*To Change the Signing Key for Federation\*](#) for details regarding the signing certificate.

2. On the OpenAM console Common Tasks page, click Configure Google Apps.
3. On the first Configure Google Apps for Single Sign-On page, add your domain name(s) such as example.com to the list, and then click Create.
4. On the second Configure Google Apps for Single Sign-On page, save the OpenAM verification certificate to a text file, such as OpenAM.pem.
5. Follow the instructions under To Enable Access to the Google Apps API before clicking Finish.
  - a. Access the Google Apps administration page for the first of your domains in a new browser tab or window.
  - b. Login as Google Apps administrator.
  - c. Select Enable Single Sign-On.
  - d. Copy the URLs from the OpenAM page into the Google Apps setup screen.
  - e. Upload the certificate file you saved such as OpenAM.pem as the Google Apps Verification Certificate.
  - f. Select Use a domain specific issuer.
  - g. Save changes in Google Apps setup.
  - h. Repeat the steps above for each domain you have configured.
  - i. Click Finish to complete the process.

## 11.8 Configuring Salesforce CRM as a Remote Service Provider

OpenAM can serve as the identity provider when you use [Salesforce CRM](#) as a service provider, allowing users to have single sign-on with their Salesforce CRM account.

In order to use this service, you must have Salesforce CRM accounts for your organization.

### Procedure 11.11. To Integrate With Salesforce CRM

1. If you have not yet done so, set up OpenAM as described in [Procedure 11.1, "To Create a Hosted Identity Provider"](#), using a signing certificate that is needed by Salesforce CRM.

See the procedure [To Change the Signing Key for Federation](#) for details regarding the signing certificate.

2. On the OpenAM console Common Tasks page, click Configure Salesforce CRM.
3. Enter the EntityID for your Salesforce service provider.

This ID is used as the persistent EntityDescriptor metadata element so that users can have multiple service provider instances. This field is used for the EntityDescriptor on the next page.

4. On the first Salesforce CRM Single Sign-On Configuration page, configure attribute mapping to associate the appropriate attribute from Salesforce CRM with the user profile attribute on your IDP.

For example, add a mapping for `IDPEmail` to `mail`, and then click Create. Make sure the attribute mapper is sending the correct attribute to be used for the federated identity.

5. On the second Salesforce CRM Single Sign-On Configuration page, follow the instructions below before clicking Finish.

- a. In a new browser tab or window, login to [Salesforce CRM](#) with your administrator credentials.

Create an administrator account if none exists, yet.

- b. If your users go directly to Salesforce to access services, then their single sign-on is SP-initiated from the Salesforce side. Salesforce provides a "My Domain" feature to facilitate SP-initiated single sign-on for desktop and device users.

When you have completed configuring Salesforce as a service provider, users can then browse to your domain at Salesforce, such as `https://openam.my.salesforce.com`, and be redirected to OpenAM to authenticate before being redirected to Salesforce.

- i. Select Administration Setup > Company Profile > My Domain.

- ii. Choose the domain name, and then register the domain.
  - iii. Wait until the domain is ready for testing to proceed.
- c. In Salesforce CRM, browse to Setup > Administration Setup > Security Controls > Single Sign-On Settings, and then click Edit for Single Sign-On Settings.
  - d. Select SAML Enabled.
  - e. Set the SAML Version to 2.0.
  - f. Copy the issuer name from the OpenAM page to the Issuer field on the Salesforce CRM page.
  - g. Copy or download the OpenAM verification certificate to a text file, such as OpenAMCert.pem or OpenAMCert.txt.
  - h. Upload the certificate file as Identity Provider Certificate on the Salesforce CRM page.
  - i. For SAML Identity Type in Salesforce CRM, choose Assertion contains the Federation ID from the User object.
  - j. For SAML Identity Location in Salesforce CRM, choose Identity is in an Attribute element.
  - k. If you require specific login or logout pages, enter them in the next two fields.
  - l. Enter the URL of your page specific error page if you have a page where you would like users redirected to when they encounter an error.
  - m. Copy the attribute name such as IDPEmail from the OpenAM page to the Attribute Name field on the Salesforce CRM page.
  - n. Salesforce uses an unspecified nameid-format value; therefore, your IdP configuration in OpenAM should reflect this. In the NameID Value Map setting, enter the following NameID Format user attribute mapping, and then click Add.

```
urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified=attribute
```

Here the *attribute* is the attribute you copied in [Step 5.m](#).

- o. Select the Entity ID corresponding to the "My Domain" that you set up.

- p. Save your work in Salesforce CRM.
  - q. Salesforce CRM displays a Salesforce Login URL.
    - Copy the Salesforce Login URL to the field provided on the OpenAM page.
  - r. Salesforce CRM returns to the Single Sign-On Settings form.
  - s. Click Download Metadata to download the Salesforce CRM SP metadata.
- After you complete the configuration, you must import the SP metadata you download in this step.
- t. In Salesforce CRM, browse to Administration Setup > Manage Users, and then click Users.
  - u. Add users as necessary, making sure the attribute chosen as the Federation ID matches the local attribute you mapped to the remote attribute in the previous page in OpenAM.
  - v. Click Finish to complete the process.
6. After you finish, import the metadata for Salesforce CRM as SP.
- a. Browse in OpenAM console to the Federation tab.
  - b. If the remote SP entity for Salesforce CRM is already in the Entity Providers list, delete the existing configuration.
  - c. Click Import Entity..., and then use the Import Entity Provider page to import the Salesforce CRM metadata.
    - Update the Realm Name to the appropriate realm.
    - Select the location where the metadata file is.
    - Enter the path for the metadata file.
    - If you have an extended data file, select the location where the file is.
    - If you have an extended data file, enter the path for the metadata file.

At this point, when a user browses to the Salesforce domain you set up, they should be redirected to OpenAM for authentication. Upon successful authentication, they should be logged in to Salesforce.

## 11.9 Using SAML 2.0 Single Sign-On & Single Logout

OpenAM SAML 2.0 Federation provides JSPs where you can direct users to do single sign-on (SSO) and single logout (SLO) across providers in a circle of trust. OpenAM has two JSPs for SSO and two JSPs for SLO, allowing you to initiate both processes either from the identity provider side, or from the service provider side.

SSO lets users sign in once and remain authenticated as they access services in the circle of trust.

SLO attempts to log a user out of all providers in the circle of trust.

The JSP pages are found under the context root where you deployed OpenAM, in `saml2/jsp/`.

### `spSSOInit.jsp`

Used to initiate SSO from the service provider side, so call this on the service provider not the identity provider. This is also mapped to the endpoint `spssoinit` under the context root.

Examples: <http://www.sp.example:8080/openam/saml2/jsp/spSSOInit.jsp>,  
<http://www.sp.example:8080/openam/spssoinit>

### `idpSSOInit.jsp`

Used to initiate SSO from the identity provider side, so call this on the identity provider not the service provider. This is also mapped to the endpoint `idpssoinit` under the context root.

Examples: <http://www.idp.example:8080/openam/saml2/jsp/idpSSOInit.jsp>,  
<http://www.idp.example:8080/openam/idpssoinit>

### `spSingleLogoutInit.jsp`

Used to initiate SLO from the service provider side, so call this on the service provider not the identity provider.

Example: <http://www.sp.example:8080/openam/saml2/jsp/spSingleLogoutInit.jsp>, <http://www.sp.example:8080/openam/SPSloInit>

### `idpSingleLogoutInit.jsp`

Used to initiate SLO from the identity provider side, so call this on the identity provider not the service provider.

Example: <http://www.idp.example:8080/openam/saml2/jsp/idpSingleLogoutInit.jsp>, <http://www.idp.example:8080/openam/IDPSloInit>

When you invoke these JSPs, there are several parameters to specify. Which parameters you can use depends on the JSP.

## **idpSSOInit.jsp Parameters**

### **metaAlias**

(Required) Use this parameter to specify the local alias for the provider, such as `metaAlias=/myRealm/idp`. This parameter takes the format `/realm-name/provider-name` as described in [MetaAlias](#). You do not repeat the slash for the top level realm, for example `metaAlias=/idp`.

### **spEntityID**

(Required) Use this parameter to indicate the remote service provider. Make sure you URL encode the value. For example, specify `spEntityID=http://www.sp.example:8080/openam` as `spEntityID=http%3A%2F%2Fwww.sp.example%3A8080%2Fopenam`.

### **affiliationID**

(Optional) Use this parameter to specify a SAML affiliation identifier.

### **binding**

(Optional) Use this parameter to indicate what binding to use for the operation. For example, specify `binding=HTTP-POST` to use HTTP POST binding with a self-submitting form. In addition to `binding=HTTP-POST`, you can also use `binding=HTTP-Artifact`.

### **NameIDFormat**

(Optional) Use this parameter to specify a SAML Name Identifier format identifier such as `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`, or `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`.

### **RelayState**

(Optional) Use this parameter to specify where to redirect the user when the process is complete. Make sure you URL encode the value. For example, `RelayState=http%3A%2F%2Fforgerock.com` takes the user to `http://forgerock.com`.

### **RelayStateAlias**

(Optional) Use this parameter to specify the parameter to use as the RelayState. For example, if your query string has `target=http%3A%2F%2Fforgerock.com&RelayStateAlias=target`, this is like setting `RelayState=http%3A%2F%2Fforgerock.com`.

## **spSSOInit.jsp Parameters**

### **idpEntityID**

(Required) Use this parameter to indicate the remote identity provider. Make sure you URL encode the value. For example, specify `idpEntityID=http://`

`www.idp.example:8080/openam as idpEntityID=http%3A%2Fwww.idp.example%3A8080%2Fopenam.`

### **metaAlias**

(Required) Use this parameter to specify the local alias for the provider, such as `metaAlias=/myRealm/sp`. This parameter takes the format `/realm-name/provider-name` as described in [MetaAlias](#). You do not repeat the slash for the top level realm, `metaAlias=/sp`.

### **affiliationID**

(Optional) Use this parameter to specify a SAML affiliation identifier.

### **AllowCreate**

(Optional) Use this parameter to indicate whether the identity provider can create a new identifier for the principal if none exists (`true`) or not (`false`).

### **AssertionConsumerServiceIndex**

(Optional) Use this parameter to specify an integer that indicates the location to which the Response message should be returned to the requester.

### **AuthComparison**

(Optional) Use this parameter to specify a comparison method to evaluate the requested context classes or statements. OpenAM accepts the following values: `better`, `exact`, `maximum`, and `minimum`.

### **AuthnContextClassRef**

(Optional) Use this parameter to specify authentication context class references. Separate multiple values with pipe characters (|).

### **AuthnContextDeclRef**

(Optional) Use this parameter to specify authentication context declaration references. Separate multiple values with pipe characters (|).

### **AuthLevel**

(Optional) Use this parameter to specify the authentication level of the authentication context that OpenAM should use to authenticate the user.

### **binding**

(Optional) Use this parameter to indicate what binding to use for the operation. For example, specify `binding=HTTP-POST` to use HTTP POST binding with a self-submitting form. In addition to `binding=HTTP-POST`, you can also use `binding=HTTP-Artifact`.

### **Destination**

(Optional) Use this parameter to specify a URI Reference indicating the address to which the request is sent.

### ForceAuthn

(Optional) Use this parameter to indicate whether the identity provider should force authentication (true) or can reuse existing security contexts (false).

### isPassive

(Optional) Use this parameter to indicate whether the identity provider should authenticate passively (true) or not (false).

### NameIDFormat

(Optional) Use this parameter to specify a SAML Name Identifier format identifier such as urn:oasis:names:tc:SAML:2.0:nameid-format:persistent, or urn:oasis:names:tc:SAML:2.0:nameid-format:transient.

### RelayState

(Optional) Use this parameter to specify where to redirect the user when the process is complete. Make sure you URL encode the value. For example, RelayState=http%3A%2F%2Fforgerock.com takes the user to http://forgerock.com.

### RelayStateAlias

(Optional) Use this parameter to specify the parameter to use as the RelayState. For example, if your query string has target=http%3A%2F%2Fforgerock.com&RelayStateAlias=target, this is like setting RelayState=http%3A%2F%2Fforgerock.com.

### reqBinding

(Optional) Use this parameter to indicate what binding to use for the authentication request. Valid values include urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect (default) and urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST.

### sunamcompositeadvice

(Optional) Use this parameter to specify a URL encoded XML blob that specifies the authentication level advice. For example, the following XML indicates a requested authentication level of 1. Notice the required : before the 1.

```
<Advice>
  <AttributeValuePair>
    <Attribute name="AuthLevelConditionAdvice"/>
    <Value>/:1</Value>
  </AttributeValuePair>
</Advice>
```

### **idpSingleLogoutInit.jsp Parameters**

#### **binding**

(Required) Use this parameter to indicate what binding to use for the operation. The full, long name format is required for this parameter to work.

The value must be one of the following.

- urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect (default)
- urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST
- urn:oasis:names:tc:SAML:2.0:bindings:SOAP

#### **Consent**

(Optional) Use this parameter to specify a URI that is a SAML Consent Identifier.

#### **Destination**

(Optional) Use this parameter to specify a URI Reference indicating the address to which the request is sent.

#### **Extension**

(Optional) Use this parameter to specify a list of Extensions as string objects.

#### **goto**

(Optional) Use this parameter to specify where to redirect the user when the process is complete. RelayState takes precedence over this parameter.

#### **logoutAll**

(Optional) Use this parameter to specify that the identity provider should send single logout requests to service providers without indicating a session index.

#### **RelayState**

(Optional) Use this parameter to specify where to redirect the user when the process is complete. Make sure you URL encode the value. For example, RelayState=http%3A%2F%2Forgerock.com takes the user to http://forgerock.com.

### **spSingleLogoutInit.jsp Parameters**

#### **binding**

(Required) Use this parameter to indicate what binding to use for the operation. The full, long name format is required for this parameter to work. For example, specify binding=urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST to use HTTP POST binding with a self-submitting form

rather than the default HTTP redirect binding. In addition, you can use binding=urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact.

**idpEntityID**

(Required for Fedlets) Use this parameter to indicate the remote identity provider. If the binding is not set, then OpenAM uses this parameter to find the default binding. Make sure you URL encode the value. For example, specify idpEntityID=http://www.sp.example:8080/openam as idpEntityID=http%3A%2Fwww.idp.example%3A8080%2Fopenam.

**NameIDValue**

(Required for Fedlets) Use this parameter to indicate the SAML Name Identifier for the user.

**SessionIndex**

(Required for Fedlets) Use this parameter to indicate the sessionIndex of the user session to terminate.

**Consent**

(Optional) Use this parameter to specify a URI that is a SAML Consent Identifier.

**Destination**

(Optional) Use this parameter to specify a URI Reference indicating the address to which the request is sent.

**Extension**

(Optional) Use this parameter to specify a list of Extensions as string objects.

**goto**

(Optional) Use this parameter to specify where to redirect the user when the process is complete. RelayState takes precedence over this parameter.

**RelayState**

(Optional) Use this parameter to specify where to redirect the user when the process is complete. Make sure you URL encode the value. For example, RelayState=http%3A%2F%2Fforgerock.com takes the user to http://forgerock.com.

**spEntityID**

(Optional, for Fedlets) Use this parameter to indicate the Fedlet entity ID. When missing, OpenAM uses the first entity ID in the metadata.

### **Example 11.1. SSO & SLO From the Service Provider**

The following URL takes the user from the service provider side to authenticate at the identity provider and then come back to the end user profile page at the service provider after successful SSO. Lines are folded to show you the query string parameters.

```
http://www.sp.example:8080/openam/saml2/jsp/spSSOInit.jsp?metaAlias=/sp
&idpEntityID=http%3A%2F%2Fwww.idp.example%3A8080%2Fopenam
&RelayState=http%3A%2F%2Fwww.sp.example%3A8080%2Fopenam%2Fidm%2FEndUser
```

The following URL initiates SLO from the service provider side, leaving the user at <http://forgerock.com>.

```
http://www.sp.example:8080/openam/saml2/jsp/spSingleLogoutInit.jsp?
&idpEntityID=http%3A%2F%2Fwww.idp.example%3A8080%2Fopenam
&RelayState=http%3A%2F%2Fforgerock.com
```

### Procedure 11.12. To Indicate Progress During SSO

During SSO log in, OpenAM presents users with a self-submitting form when access has been validated. This page is otherwise blank. If you want to present users with something to indicate that the operation is in progress, then customize the necessary templates.

1. Modify the templates to add a clue that SSO is in progress, such as an image.

Edit the source of the OpenAM Java Server Page, `saml2/jsp/autosubmitaccessrights.jsp`, under the file system directory where the OpenAM .war has been unpacked.

When you add an image or other presentation element, make sure that you retain the form and Java code as is.

2. Unpack OpenAM-12.0.0.war, and add your modified template files under `WEB-INF/classes/` where you unpacked the .war.

Also include any images you reference in the page.

3. Pack up your custom version of OpenAM, and then deploy it in your web container.

## 11.10 Configuring OpenAM For the ECP Profile

The SAML 2.0 Enhanced Client or Proxy (ECP) profile is intended for use when accessing services over devices like simple phones, medical devices, and set-top boxes that lack the capabilities needed to use the more widely used SAML 2.0 Web Browser SSO profile.

The ECP knows which identity provider to contact for the user, and is able to use the reverse SOAP (PAOS) SAML 2.0 binding for the authentication request and response. The PAOS binding uses HTTP and SOAP headers to pass information about processing SOAP requests and responses, starting with a PAOS HTTP header that the ECP sends in its initial request to the server. The PAOS messages

continue with a SOAP authentication request in the server's HTTP response to the ECP's request for a resource, followed by a SOAP response in an HTTP request from the ECP.

An enhanced client, such as a browser with a plugin or an extension, can handle these communications on its own. An enhanced proxy is an HTTP server such as a WAP gateway that can support the ECP profile on behalf of client applications.

OpenAM supports the SAML 2.0 ECP profile on the server side for identity providers and service providers. You must build the ECP.

By default an OpenAM identity provider uses the `com.sun.identity.saml2.plugins.DefaultIDPECPSessionMapper` class to find a user session for requests to the IDP from the ECP. The default session mapper uses OpenAM cookies as it would for any other client application. If for some reason you must change the mapping after writing and installing your own session mapper, you can change the class under Federation > Entity Providers > *idp-name* > IDP > Advanced > ECP Configuration.

By default an OpenAM service provider uses the `com.sun.identity.saml2.plugins.ECPIDPFinder` class to return identity providers from the list under Federation > Entity Providers > *sp-name* > SP > Advanced > ECP Configuration > Request IDP List. You must populate the list with identity provider entity IDs.

The endpoint for the ECP to contact on the OpenAM service provider is /SPECP as in `http://www.sp.example:8080/openam/SPECP`. The ECP provides two query string parameters to identify the service provider and to specify the URL of the resource to access.

#### `metaAlias`

This specifies the service provider, by default `metaAlias=/realm-name/sp`, as described in [MetaAlias](#).

#### `RelayState`

This specifies the resource the client aims to access such as `RelayState=http%3A%2F%2forgerock.org%2Findex.html`.

For example, the URL to access the service provider and finally the resource at `http://forgerock.org/index.html` could be `http://www.sp.example:8080/openam/SPECP?metaAlias=/sp&RelayState=http%3A%2F%2forgerock.org%2Findex.html`.

## 11.11 Managing Federated Accounts

Identity providers and service providers must be able to communicate about users. Yet in some cases the identity provider can choose to communicate a minimum of information about an authenticated user, with no user account maintained on the service provider side. In other cases the identity provider and service provider can choose to link user accounts in a persistent way, in a more

permanent way, or even in automatic fashion by using some shared value in the user's profiles such as an email address or by dynamically creating accounts on the service provider when necessary. OpenAM supports all these alternatives.

### 11.11.1 Using Transient Federation Identifiers

OpenAM allows you to link accounts using transient name identifiers, where the identity provider shares a temporary identifier with the service provider for the duration of the user session. Nothing is written to the user profile.

Transient identifiers are useful where the service is anonymous, and all users have similar access on the service provider side.

To use transient name identifiers, specify the name ID format `urn:oasis:names:tc:SAML:2.0:nameid-format:transient` when initiating single sign on.

The examples below work in an environment where the identity provider is `www.idp.example` and the service provider is `www.sp.example`. Both providers have deployed OpenAM on port 8080 under deployment URI `/openam`.

To initiate single sign on from the service provider, access the following URL with at least the query parameters shown.

```
http://www.sp.example:8080/openam/saml2/jsp/spSSOInit.jsp?  
idpEntityID=http%3A%2F%2Fwww.idp.example%3A8080%2Fopenam  
&metaAlias=/sp  
&NameIDFormat=urn:oasis:names:tc:SAML:2.0:nameid-format:transient
```

For a complete list of query parameters, see [spSSOInit.jsp Parameters](#).

To initiate single sign on from the identity provider, access the following URL with at least the query parameters shown.

```
http://www.idp.example:8080/openam/saml2/jsp/idpSSOInit.jsp?  
spEntityID=http%3A%2F%2Fwww.sp.example%3A8080%2Fopenam  
&metaAlias=/idp  
&NameIDFormat=urn:oasis:names:tc:SAML:2.0:nameid-format:transient
```

For a complete list of query parameters, see [idpSSOInit.jsp Parameters](#).

The accounts are only linked for the duration of the session. Once the user logs out for example the accounts are no longer linked.

### 11.11.2 Using Persistent Federation Identifiers

OpenAM lets you use persistent pseudonym identifiers to federate user identities, linking accounts on the identity provider and service provider with a SAML persistent identifier.

Persistent identifiers are useful for establishing links between otherwise unrelated accounts.

The examples below work in an environment where the identity provider is `www.idp.example` and the service provider is `www.sp.example`. Both providers have deployed OpenAM on port 8080 under deployment URI `/openam`.

To initiate single sign on from the service provider, access the following URL with at least the query parameters shown.

```
http://www.sp.example:8080/openam/saml2/jsp/spSSOInit.jsp?  
idpEntityID=http%3A%2F%2Fwww.idp.example%3A8080%2Fopenam  
&metaAlias=/sp  
&NameIDFormat=urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
```

For a complete list of query parameters, see [spSSOInit.jsp Parameters](#).

To initiate single sign on from the identity provider, access the following URL with at least the query parameters shown.

```
http://www.idp.example:8080/openam/saml2/jsp/idpSSOInit.jsp?  
spEntityID=http%3A%2F%2Fwww.sp.example%3A8080%2Fopenam  
&metaAlias=/idp  
&NameIDFormat=urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
```

For a complete list of query parameters, see [idpSSOInit.jsp Parameters](#).

On successful login, the accounts are persistently linked, with persistent identifiers stored in the user's accounts on the identity provider and the service provider.

### 11.11.3 Changing Federation of Persistently Linked Accounts

OpenAM implements the SAML 2.0 Name Identifier Management profile, allowing you to change a persistent identifier that has been set to federate accounts, and also to terminate federation for an account.

When user accounts are stored in an LDAP directory server, name identifier information is stored on the `sun-fm-saml2-nameid-info` and `sun-fm-saml2-nameid-infokey` attributes of a user's entry.<sup>1</sup>

You can retrieve the name identifier value on the IDP side by checking the value of `sun-fm-saml2-nameid-infokey`. For example, if the user's entry in the directory shows `sun-fm-saml2-nameid-infokey: http://www.idp.example:8080/openam|http://www.sp.example:8080/openam|XyFFEsr6Vixbnt0BSqIgllLFMGjR2`, then the name identifier on the IDP side is `XyFFEsr6Vixbnt0BSqIgllLFMGjR2`.

---

<sup>1</sup>These attribute types are configurable in the OpenAM console under Configuration > Global > SAMLv2 Service Configuration.

You can use this identifier to initiate a change request from the service provider as in the following example.

```
http://www.sp.example:8080/openam/saml2/jsp/spMNIRequestInit.jsp?  
idpEntityID=http%3A%2Fwww.idp.example%3A8080%2Fopenam  
&metaAlias=/sp  
&requestType>NewID  
&IDPProvidedID=XyffEsrr6Vixbnt0BSqIglLFMGjR2
```

If desired, you can substitute openam/SPMniInit for openam/saml2/jsp/  
spMNIRequestInit.jsp

You can also initiate the change request from the identity provider as in the following example.

```
http://www.idp.example:8080/openam/saml2/jsp/idpMNIRequestInit.jsp?  
spEntityID=http%3A%2Fwww.sp.example%3A8080%2Fopenam  
&metaAlias=/idp  
&requestType>NewID  
&IDPProvidedID=XyffEsrr6Vixbnt0BSqIglLFMGjR2
```

If desired, you can substitute openam/IDPMniInit for openam/saml2/jsp/  
idpMNIRequestInit.jsp

You can retrieve the name identifier value on the SP side by checking the value of sun-fm-saml2-nameid-info. For example, if the user's entry in the directory shows sun-fm-saml2-nameid-info: `http://www.sp.example:8080/openam| http://www.idp.example:8080/openam| ATo9TSA9Y2Ln7DDrAd03HFFH5jKD| http://www.idp.example:8080/openam| urn:oasis:names:tc:SAML:2.0:nameid-format:persistent| 9B10Py3m0ejv3fZYhlqxXmiGD24c| http://www.sp.example:8080/openam| SPRole| false`, then the name identifier on the SP side is 9B10Py3m0ejv3fZYhlqxXmiGD24c.

## **idpMNIRequestInit.jsp Parameters**

### **spEntityID**

(Required) Use this parameter to indicate the remote service provider. Make sure you URL encode the value. For example, specify `spEntityID=http://www.sp.example:8080/openam` as `spEntityID=http%3A%2Fwww.sp.example%3A8080%2Fopenam`.

### **metaAlias**

(Required) Use this parameter to specify the local alias for the provider, such as `metaAlias=/myRealm/idp`. This parameter takes the format `/realm-name/provider-name` as described in [MetaAlias](#). You do not repeat the slash for the top level realm, for example `metaAlias=/idp`.

### **requestType**

(Required) Type of manage name ID request, either `NewID` to change the ID, or `Terminate` to remove the information that links the accounts on the identity provider and service provider.

**SPProvidedID**

(Required if `requestType=NewID`) Name identifier in use as described above.

**affiliationID**

(Optional) Use this parameter to specify a SAML affiliation identifier.

**binding**

(Optional) Use this parameter to indicate what binding to use for the operation. The full, long name format is required for this parameter to work.

The value must be one of the following.

- `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST`
- `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect`
- `urn:oasis:names:tc:SAML:2.0:bindings:SOAP`

**relayState**

(Optional) Use this parameter to specify where to redirect the user when the process is complete. Make sure you URL encode the value. For example, `relayState=http%3A%2F%2Fforgerock.com` takes the user to `http://forgerock.com`.

## **spMNIRRequestInit.jsp Parameters**

**idpEntityID**

(Required) Use this parameter to indicate the remote identity provider. Make sure you URL encode the value. For example, specify `idpEntityID=http://www.idp.example:8080/openam` as `idpEntityID=http%3A%2F%2Fwww.idp.example%3A8080%2Fopenam`.

**metaAlias**

(Required) Use this parameter to specify the local alias for the provider, such as `metaAlias=/myRealm/sp`. This parameter takes the format `/realm-name/provider-name` as described in [MetaAlias](#). You do not repeat the slash for the top level realm, `metaAlias=/sp`.

**requestType**

(Required) Type of manage name ID request, either `NewID` to change the ID, or `Terminate` to remove the information that links the accounts on the identity provider and service provider.

**IDPProvidedID**

(Required if `requestType=NewID`) Name identifier in use as described above.

**affiliationID**

(Optional) Use this parameter to specify a SAML affiliation identifier.

**binding**

(Optional) Use this parameter to indicate what binding to use for the operation. The full, long name format is required for this parameter to work.

The value must be one of the following.

- urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST
- urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect
- urn:oasis:names:tc:SAML:2.0:bindings:SOAP

**relayState**

(Optional) Use this parameter to specify where to redirect the user when the process is complete. Make sure you URL encode the value. For example, relayState=http%3A%2F%2Fforgerock.com takes the user to http://forgerock.com.

You can terminate federation as described in [Section 11.11.4, “Terminating Federation of Persistently Linked Accounts”](#).

#### **11.11.4 Terminating Federation of Persistently Linked Accounts**

OpenAM lets you terminate account federation, where the accounts have been linked with a persistent identifier as described in [Section 11.11.2, “Using Persistent Federation Identifiers”](#).

The examples below work in an environment where the identity provider is www.idp.example and the service provider is www.sp.example. Both providers have deployed OpenAM on port 8080 under deployment URI /openam.

To initiate the process of terminating account federation from the service provider, access the following URL with at least the query parameters shown.

```
http://www.sp.example:8080/openam/saml2/jsp/spMNIRestartInit.jsp?  
idpEntityID=http%3A%2F%2Fwww.idp.example%3A8080%2Fopenam  
&metaAlias=/sp  
&requestType=Terminate
```

To initiate the process of terminating account federation from the identity provider, access the following URL with at least the query parameters shown.

```
http://www.idp.example:8080/openam/saml2/jsp/idpMNIRestartInit.jsp?  
spEntityID=http%3A%2F%2Fwww.sp.example%3A8080%2Fopenam  
&metaAlias=/idp  
&requestType=Terminate
```

## **11.11.5 Configuring How Remote Accounts Map To Local Accounts**

---

OpenAM lets you configure the service provider to link an account based on an attribute value from the identity provider. When you know the user accounts on both the identity provider and the service provider share a common attribute value, such as an email address or other unique user identifier, you can use this method to link accounts without user interaction. See [Procedure 11.13, “To Map Accounts Based on an Attribute Value”](#).

OpenAM also lets you map users on the identity provider temporarily to a single anonymous user account on the service provider, in order to exchange attributes about the user without a user-specific account on the service provider. This approach can be useful when the service provider either needs no user-specific account to provide a service, or when you do not want to retain a user profile on the service provider but instead you make authorization decisions based on attribute values from the identity provider. See [Procedure 11.14, “To Map Remote Accounts to a Single Service Provider Account”](#).

OpenAM further allows you to use attributes from the identity provider to create accounts dynamically on the service provider. When using this method, you should inform the user and obtain consent to create the account if necessary. See [Procedure 11.15, “To Map Accounts With Dynamic Service Provider Account Creation”](#).

### **Procedure 11.13. To Map Accounts Based on an Attribute Value**

The following steps demonstrate how to map accounts based on an attribute value that is the same in both accounts.

Perform the following steps on the hosted identity provider(s), and again on the hosted service provider(s).

1. Login to the OpenAM console as administrator.
2. Browse to Federation > *hosted-provider-name* > Assertion Processing.
3. If the attribute to use when linking accounts is not yet included in the attribute map, add the attribute mapping, and then Save your work.
4. On the hosted service provider, under Auto Federation, select Enabled and enter the local attribute name in the Attribute field, and then Save your work.

### **Procedure 11.14. To Map Remote Accounts to a Single Service Provider Account**

The following steps demonstrate how to auto-federate using a single anonymous user account on the service provider.

## Configuring How Remote Accounts Map To Local Accounts

---

Perform the following steps on the hosted identity provider(s), and again on the hosted service provider(s).

1. Login to the OpenAM console as administrator.
2. Browse to Federation > *hosted-provider-name* > Assertion Processing.
3. If you want to get attributes from the identity provider and the attributes are not yet in the attribute map, add the attribute mappings, and then Save your work.
4. On the hosted service provider, under Transient User, set the single account to which to map all users, such as anonymous, and then Save your work.
5. After completing configuration on the providers, use transient identifiers to federate as described in [Section 11.11.1, “Using Transient Federation Identifiers”](#).

### **Procedure 11.15. To Map Accounts With Dynamic Service Provider Account Creation**

The following steps demonstrate how to map accounts with dynamic creation of missing accounts on the service provider side.

1. Set up a mapping based on an attribute value as described in [Procedure 11.13, “To Map Accounts Based on an Attribute Value”](#). The attributes you map from the identity provider are those that the service provider sets on the dynamically created accounts.
2. On the service provider console, browse to Access Control > *realm-name* > Authentication > All Core Settings..., and Dynamic or Dynamic with User Alias, which are described in [Hints For the Core Authentication Module](#), and then Save your work.
3. To test your work, create a user on the identity provider, log out of the console, and initiate SSO logging in as the user you created.

To initiate SSO, browse to one of the OpenAM SAML 2.0 JSPs with the appropriate query parameters. The following is an example URL for service provider initiated SSO.

```
http://www.sp.example:8080/openam/saml2/jsp/spSSOInit.jsp?  
idpEntityID=http%3A%2F%2Fwww.idp.example%3A8080%2Fopenam  
&metaAlias=/sp
```

On success, check <http://www.sp.example:8080/openam/idm/EndUser> to see the new user account.

## 11.11.6 Linking Federated Accounts in Bulk

If you manage both the identity provider and service provider, you can link accounts out-of-band, in bulk. You make permanent connections for a list of identity provider and service provider by using the **ssoadm** bulk federation commands.

Before you can run the bulk federation commands, first establish the relationship between accounts, set up the providers as described in [Section 11.2, “Setting Up SAML 2.0 SSO”](#), and install the **ssoadm** command as described in [To Set Up Administration Tools](#).

To understand the relationships between accounts, consider an example where the identity provider is at `idp.example.org` and the service provider is at `sp.example.com`. A demo user account has the Universal ID, `id=demo,ou=user,dc=example,dc=org`, on the identity provider. That maps to the Universal ID, `id=demo,ou=user,dc=com`, on the service provider.

The **ssoadm** command then needs a file that maps local user IDs to remote user IDs, one per line, separated by the vertical bar character `|`. Each line of the file appears as follows.

```
local-user-ID|remote-user-ID
```

In the example, starting on the service provider side, the line for the demo user reads as follows.

```
id=demo,ou=user,dc=example,dc=com|id=demo,ou=user,dc=example,dc=org
```

All the users' accounts mapped in your file must exist at the identity provider and the service provider when you run the commands to link them.

Link the accounts using the **ssoadm** bulk federation commands.

1. Prepare the data with the **ssoadm do-bulk-federation** command.

The following example starts on the service provider side.

```
$ cat /tmp/user-map.txt
id=demo,ou=user,dc=example,dc=com|id=demo,ou=user,dc=example,dc=org
$ ssoadm \
  do-bulk-federation \
  --metaalias /sp \
  --remoteentityid http://idp.example.org:8080/openam \
  --useridmapping /tmp/user-map.txt \
  --nameidmapping /tmp/name-map.txt \
  --adminid amadmin \
  --password-file /tmp/pwd.txt \
  --spec saml2
```

```
Bulk Federation for this host was completed. To complete the federation, name Id mapping file should be loaded to remote provider.
```

2. Copy the name ID mapping output file to the other provider.

```
$ scp /tmp/name-map.txt openam@idp.example.org:/tmp/name-map.txt  
openam@idp.example.org's password:  
name-map.txt          100%  177      0.2KB/s   00:00
```

3. Import the name ID mapping file with the **ssoadm import-bulk-fed-data** command.

The following example is performed on the identity provider side.

```
$ ssoadm \  
import-bulk-fed-data \  
--adminid amadmin \  
--password-file /tmp/pwd.txt \  
--metaalias /idp \  
--bulk-data-file /tmp/name-map.txt  
  
Bulk Federation for this host was completed.
```

At this point the accounts are linked.

### 11.11.7 Authentication & Linked Accounts

In a SAML 2.0 federation where accounts are durably linked, authentication is required only on the identity provider side.

Authentication is also required however on the service provider side when the OpenAM account mapper on the service provider is not able to map the user identified in the SAML assertion from the identity provider to a local user account.

This can happen for example the first time accounts are linked as described in [Section 11.11.2, “Using Persistent Federation Identifiers”](#), after which the persistent identifier establishes the mapping.

This also happens when transient identifiers are used to map accounts. When accounts are linked as described in [Section 11.11.1, “Using Transient Federation Identifiers”](#), then the service provider must locally authenticate the user for every SAML assertion received. This is because the identifier used to link the accounts is transient: it does not provide a durable means to link the accounts.

## 11.12 Using SAML 2.0 Artifacts

By default OpenAM transmits SAML messages by value. This makes it possible to access the SAML messages in the user agent. You can instead request that OpenAM transmit SAML messages by reference using SAML artifacts, which are small values that reference a SAML message. Providers then communicate directly to resolve artifacts, rather than sending the messages through the user agent.

When initiating single sign-on using `idpSSOInit.jsp` or `spSSOInit.jsp` for example, add `binding=HTTP-Artifact` to the list of query parameters. The following example works in an environment where the identity provider is `www.idp.example` and the service provider is `www.sp.example`. Both providers have deployed OpenAM on port 8080 under deployment URI `/openam`.

```
http://www.sp.example:8080/openam/saml2/jsp/spSSOInit.jsp?
  idpEntityID=http%3A%2F%2Fwww.idp.example%3A8080%2Fopenam
  &metaAlias=/sp
  &binding=HTTP-Artifact
```

## 11.13 SAML 2.0 & Policy Agents

You can use policy agents in a SAML 2.0 Federation deployment.

### Procedure 11.16. To Use a Policy Agent with a SAML 2.0 Service Provider

The following procedure applies when OpenAM is configured as an IDP in one domain, and the desired policy agent protects resources on behalf of a second OpenAM server configured as an SP on a second domain.

1. Install the policy agent.

The basic process for installing policy agents is available in the [Web Policy Agent User's Guide](#) and the [Java EE Policy Agent User's Guide](#).

2. Replace the given OpenAM Login URL and OpenAM Logout URLs with SAML 2.0 URLs described in [Section 11.9, “Using SAML 2.0 Single Sign-On & Single Logout”](#).

The following steps explain how to do this for web policy agents.

- If you have configured the Web policy agents to store their properties centralized on an OpenAM server, navigate to the URL for the OpenAM console. Select Access Control > *Realm Name* > Agents > Web > *Agent Name* > OpenAM Services.

For the Web Agent, under the OpenAM Services tab, in the Agent Logout URL section, set up a list of application logout URLs. In the Logout Redirect URL text box, enter an appropriate URL to redirect the user after logout.

- Alternatively, if the Web policy agents are set up to store properties on local systems, find the `OpenSSOAgentConfiguration.properties` file in the `/path/to/agent/config/` directory.

You can specify OpenAM Login and Logout URLs with the `com.sun.identity.agents.config.login.url` and `com.sun.identity.agents.config.logout.url` attributes, respectively.



---

## Chapter 12

# Managing OAuth 2.0 Authorization

This chapter covers OpenAM support for the OAuth 2.0 authorization framework. The chapter begins by showing where OpenAM fits into the OAuth 2.0 authorization framework, and then shows how to configure the functionality.

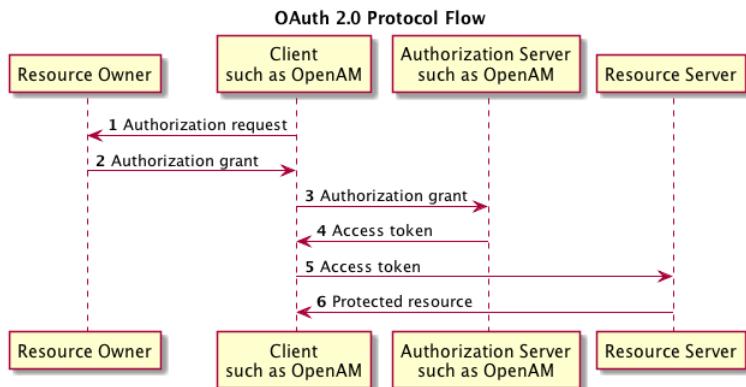
## 12.1 About OAuth 2.0 Support in OpenAM

RFC 6749, [The OAuth 2.0 Authorization Framework](#), provides a standard way for *resource owners* to grant *client* applications access to the owners' web-based resources. The canonical example involves a user (resource owner) granting access to a printing service (client) to print photos that the user has stored on a photo-sharing server.

The section describes how OpenAM supports the OAuth 2.0 authorization framework in terms of the roles that OpenAM plays.<sup>1</sup> The following sequence diagram indicates the primary roles OpenAM can play in the OAuth 2.0 protocol flow.

---

<sup>1</sup>Read [RFC 6749](#) to understand the authorization framework itself.



## 12.1.1 OpenAM as OAuth 2.0 Authorization Server

OpenAM can function as an OAuth 2.0 *authorization server*. In this role, OpenAM authenticates resource owners and obtains their authorization in order to return access tokens to clients.

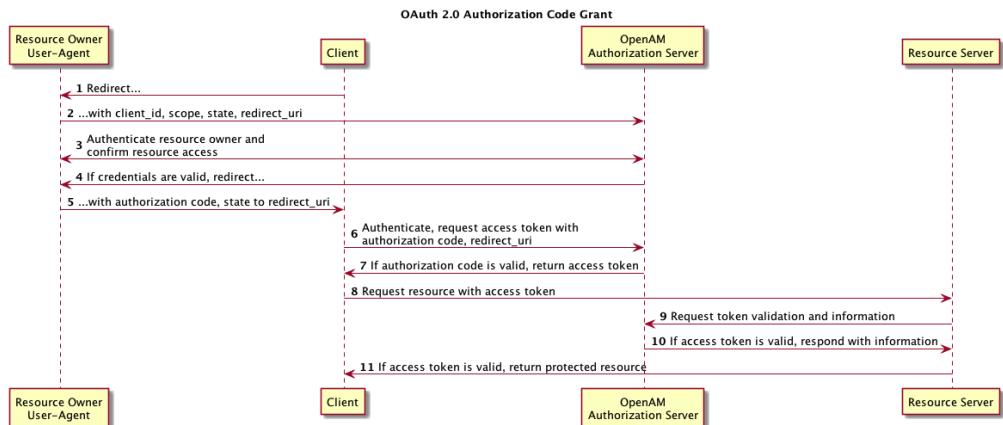
When using OpenAM as authorization server, you can register clients in OpenAM Console alongside policy agent profiles under the OAuth 2.0 Client tab. OpenAM supports both confidential and public clients.

OpenAM supports the four main grants for obtaining authorization described in RFC 6749: the authorization code grant, the implicit grant, the resource owner password credentials grant, and the client credentials grant. See RFC 6749 for details on the authorization grant process, and for details on how clients should make authorization requests and handle authorization responses. OpenAM also supports the [SAML 2.0 Bearer Assertion Profiles for OAuth 2.0](#), described in the Internet-Draft.

### 12.1.1.1 OAuth 2.0 Authorization Grant

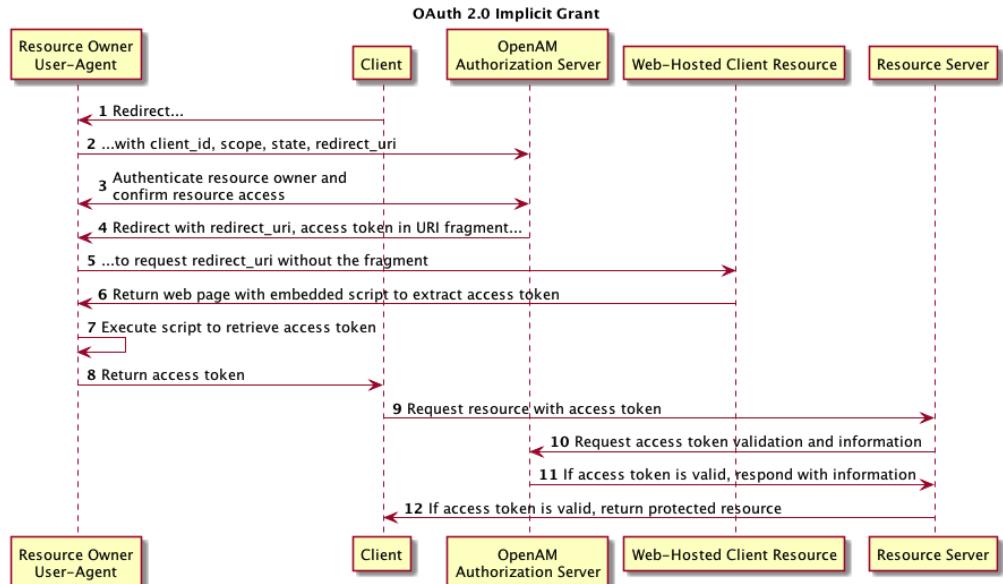
The authorization code grant starts with the client, such as a web-based service, redirecting the resource owner's user-agent to the OpenAM authorization service. After authenticating the resource owner and obtaining the resource owner's authorization, OpenAM redirects the resource owner's user-agent back to the client with an authorization code that the client uses to request the access token. The following sequence diagram outlines a successful process from initial client redirection through to the client accessing the protected resource.

## OpenAM as OAuth 2.0 Authorization Server



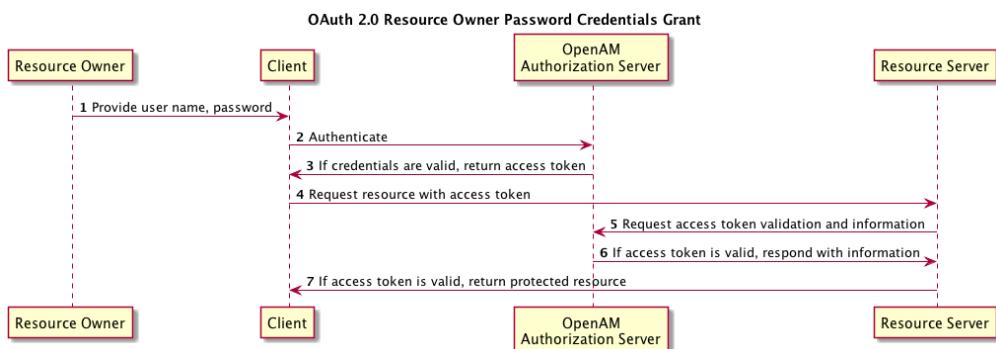
### 12.1.1.2 OAuth 2.0 Implicit Grant

The implicit grant is designed for clients implemented to run inside the resource-owner user agent. Instead of providing an authorization code that the client must use to retrieve an access token, OpenAM returns the access token directly in the fragment portion of the redirect URI. The following sequence diagram outlines the successful process.



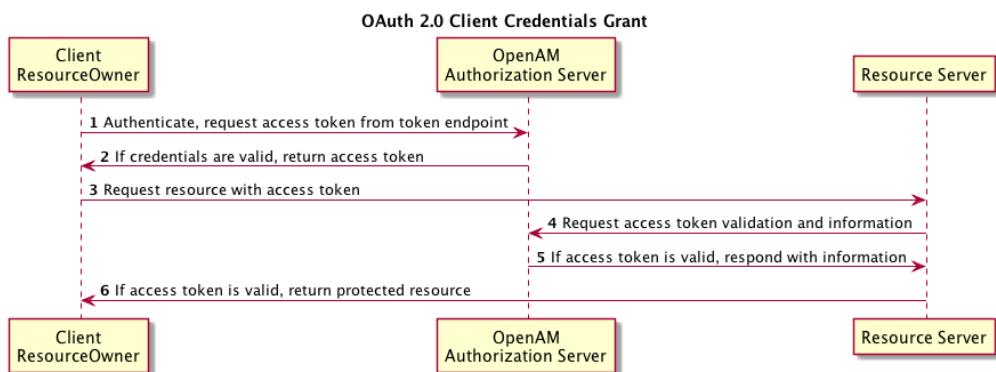
### 12.1.1.3 OAuth 2.0 Resource Owner Password Credentials Grant

The resource owner password credentials grant lets the client use the resource owner's user name and password to get an access token directly. Although this grant might seem to conflict with an original OAuth goal of not having to share resource owner credentials with the client, it can make sense in a secure context where other authorization grant types are not available, such as a client that is part of a device operating system using the resource owner credentials once and thereafter using refresh tokens to continue accessing resources. The following sequence diagram shows the successful process.



### 12.1.1.4 OAuth 2.0 Client Credentials Grant

The client credentials grant uses client credentials as an authorization grant. This grant makes sense when the client is also the resource owner, for example. The following sequence diagram shows the successful process.



### 12.1.1.5 JWT Bearer Profile

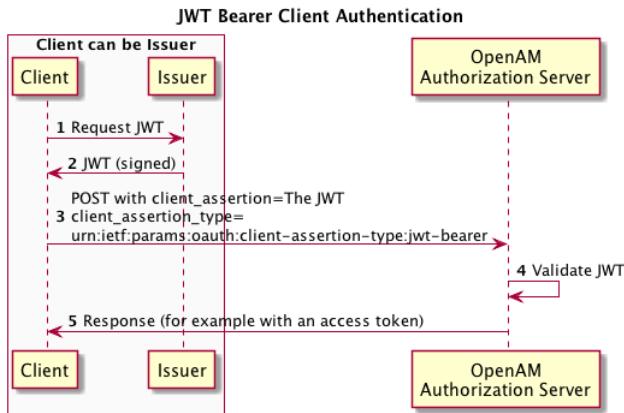
The Internet-Draft, [JSON Web Token \(JWT\) Profile for OAuth 2.0 Client Authentication and Authorization Grants](#) describes a means to use a JWT for

## OpenAM as OAuth 2.0 Authorization Server

client authentication or to use a JWT to request an access token. When clients are also resource owners, the profile allows clients to issue JWTs to obtain access tokens rather than use the resource owner password credentials grant.

OpenAM implements both features of the profile. Both involve HTTP POST requests to the access token endpoint.

When the client bearing the JWT uses it for authentication, then in the POST data the client sets `client_assertion_type` to `urn:ietf:params:oauth:client-assertion-type:jwt-bearer` and `client_assertion` to the JWT string.



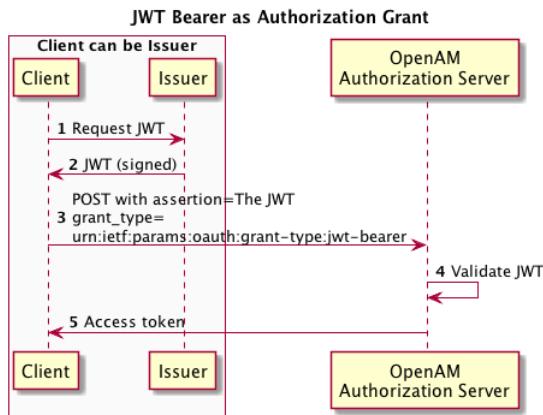
The HTTP POST to OpenAM looks something like the following, where the assertion value is the JWT.

```
POST /openam/oauth2/access_token HTTP/1.1
Host: openam.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
code=362ad374-735c-4f69-aa8e-bf384f8602de&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A
client-assertion-type%3Ajwt-bearer&
client_assertion=eyJhbGciOiJIUzI1NiBhaIwic3ViIjogImP3...
```

When the client bearing the JWT uses it as an authorization grant, then in the POST data the client sets `grant_type` to `urn:ietf:params:oauth:grant-type:jwt-bearer` and `assertion` to the JWT string.

## OpenAM as OAuth 2.0 Authorization Server



The HTTP POST to OpenAM looks something like the following, where the assertion value is the JWT. This listing does not show the client credentials, which must be provided, for example as form parameters, a JWT token, or an authorization header.

```
POST /openam/oauth2/access_token HTTP/1.1
Host: openam.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-bearer&
assertion=eyJhbGciOiJIUzI1NiBha...  
eyAiYXnIjogIlJTmjU2IiB9.eyJpc3ViIjogImp3...
```

In both profiles, OpenAM must be able to validate the JWT.

For validation, the JWT must include the following claims:

- "iss" (issuer) whose value identifies the JWT issuer
- "sub" (subject) whose value identifies the principal who is the subject of the JWT

For client authentication, the "sub" value must be the same as the value of the "client\_id".

- "aud" (audience) whose value identifies the authorization server that is the intended audience of the JWT

When the JWT is used for authentication, this is the OpenAM access token endpoint.

- "exp" (expiration) whose value specifies the time of expiration

Also for validation, the issuer must digitally sign the JWT or apply a keyed message digest. When the issuer is also the client, the client can sign the JWT by using a private key, and include the public key in its profile registered with OpenAM.

A sample Java-based client is [available online](#).

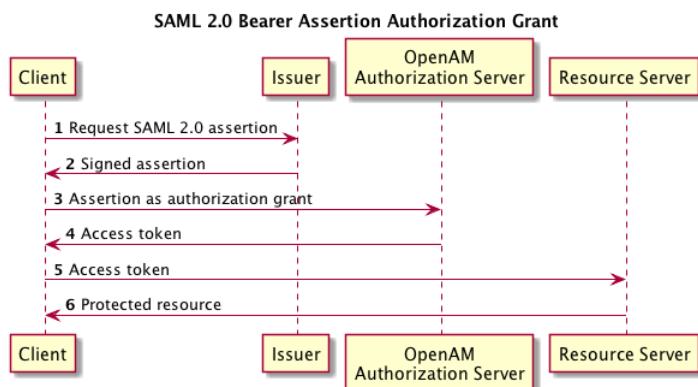
### 12.1.1.6 SAML 2.0 Bearer Assertion Profiles

The Internet-Draft, [SAML 2.0 Bearer Assertion Profiles for OAuth 2.0](#), describes a means to use SAML 2.0 assertions to request access tokens and to authenticate OAuth 2.0 clients.

At present OpenAM implements the profile to request access tokens.

In both profiles, the issuer must sign the assertion. The client communicates the assertion over a channel protected with transport layer security, by performing an HTTP POST to the OpenAM's access token endpoint. OpenAM as OAuth 2.0 authorization server uses the issuer ID to validate the signature on the assertion.

In the profile to request an access token, the OAuth 2.0 client bears a SAML 2.0 assertion that was issued to the resource owner on successful authentication. A valid assertion in this case is equivalent to an authorization grant by the resource owner to the client. OAuth 2.0 clients must make it clear to the resource owner that by authenticating to the identity provider who issues the assertion, they are granting the client permission to access the protected resources.



The HTTP POST to OpenAM to request an access token looks something like this:

```
POST /openam/oauth2/access_token HTTP/1.1
Host: openam.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Asaml2-bearer&
assertion=PHNhbwXw0l...[base64url encoded assertion]...ZT4&
client_id=[ID registered with OpenAM]
```

If OpenAM is already a SAML 2.0 service provider, you can configure OpenAM as OAuth 2.0 authorization server as well, and set an adapter class name in the service provider configuration that lets OpenAM POST the assertion from the service provider to the authorization server. See [Section 12.4.2, “Configuring OpenAM as Both SAML 2.0 Service Provider & OAuth 2.0 Authorization Server”](#) for details.

### 12.1.1.7 OpenAM OAuth 2.0 Endpoints

In addition to the standard authorization and token endpoints described in RFC 6749, OpenAM also exposes a token information endpoint for resource servers to get information about access tokens so they can determine how to respond to requests for protected resources. OpenAM as authorization server exposes the following endpoints for clients and resource servers.

`/oauth2/authorize`

Authorization endpoint defined in RFC 6749, used to obtain an authorization grant from the resource owner

Example: `https://openam.example.com:8443/openam/oauth2/authorize`

`/oauth2/access_token`

Token endpoint defined in RFC 6749, used to obtain an access token from the authorization server

Example: `https://openam.example.com:8443/openam/oauth2/access_token`

`/oauth2/tokeninfo`

Endpoint not defined in RFC 6749, used to validate tokens, and to retrieve information such as scopes

Given an access token, a resource server can perform an HTTP GET on `/oauth2/tokeninfo?access_token=token-id` to retrieve a JSON object indicating `token_type`, `expires_in`, `scope`, and the `access_token` ID.

Example: `https://openam.example.com:8443/openam/oauth2/tokeninfo`

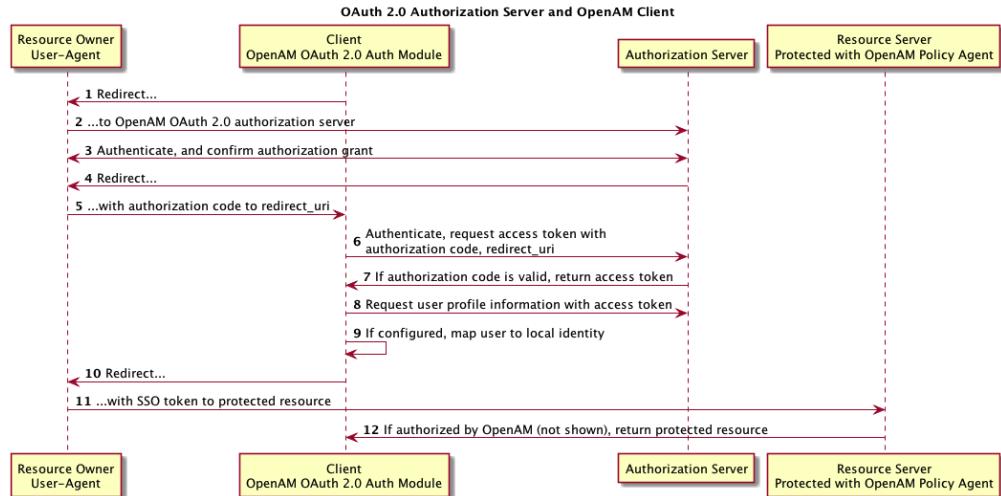
For examples, see the *Developer's Guide* section, [OAuth 2.0 Authorization](#). This section of the *Developer's Guide* also covers OAuth 2.0 token administration and client administration endpoints that are specific to OpenAM.

### 12.1.2 OpenAM as OAuth 2.0 Client & Resource Server Solution

OpenAM can function as an OAuth 2.0 client for installations where the web resources are protected by OpenAM. To configure OpenAM as an OAuth 2.0 client, you set up an OpenAM OAuth 2.0 / OpenID Connect authentication module instance, and then integrate the authentication module into your authentication chains as necessary.

## Using Your Own Client & Resource Server

When OpenAM functions as an OAuth 2.0 client, OpenAM provides an OpenAM SSO session after successfully authenticating the resource owner and obtaining authorization. This means the client can then access resources protected by policy agents. In this respect the OpenAM OAuth 2.0 client is just like any other authentication module, one that relies on an OAuth 2.0 authorization server to authenticate the resource owner and obtain authorization. The following sequence diagram shows how the client gains access to protected resources in the scenario where OpenAM functions as both authorization server and client for example.



As the OAuth 2.0 client functionality is implemented as an OpenAM authentication module, you do not need to deploy your own resource server implementation when using OpenAM as an OAuth 2.0 client. Instead, use policy agents or OpenIG to protect resources.

To configure OpenAM as an OAuth 2.0 client, see the section [Hints for the OAuth 2.0 / OpenID Connect Authentication Module](#).

### 12.1.3 Using Your Own Client & Resource Server

OpenAM returns bearer tokens as described in RFC 6750, [The OAuth 2.0 Authorization Framework: Bearer Token Usage](#). Notice in the following example JSON response to an access token request that OpenAM returns a refresh token with the access token. The client can use the refresh token to get a new access token as described in RFC 6749.

```
{  
  "expires_in": 599,  
  "token_type": "Bearer",
```

```
        "refresh_token": "f6dcf133-f00b-4943-a8d4-ee939fc1bf29",
        "access_token": "f9063e26-3a29-41ec-86de-1d0d68aa85e9"
    }
```

In addition to implementing your client, the resource server must also implement the logic for handling access tokens. The resource server can use the /oauth2/tokeninfo endpoint to determine whether the access token is still valid, and to retrieve the scopes associated with the access token.

The default OpenAM implementation of OAuth 2.0 scopes assumes that the space-separated (%20 when URL encoded) list of scopes in an access token request correspond to names of attributes in the resource owner's profile.

To take a concrete example, consider an access token request where scope=mail %20cn and where the resource owner is the default OpenAM demo user. (The demo user has no email address by default, but you can add one, such as demo@example.com to the demo user's profile.) When the resource server performs an HTTP GET on the token information endpoint, /oauth2/tokeninfo?access\_token=token-id, OpenAM populates the mail and cn scopes with the email address (demo@example.com) and common name (demo) from the demo user's profile. The result is something like the following token information response.

```
{
    "mail": "demo@example.com",
    "scope": [
        "mail",
        "cn"
    ],
    "cn": "demo",
    "realm": "/",
    "token_type": "Bearer",
    "expires_in": 577,
    "access_token": "f9063e26-3a29-41ec-86de-1d0d68aa85e9"
}
```

OpenAM is designed to allow you to plug in your own scopes implementation if the default implementation does not do what your deployment requires. See [Customizing OAuth 2.0 Scope Handling](#) for an example.

## 12.2 Configuring the OAuth 2.0 Authorization Service

You configure the OAuth 2.0 authorization service for a particular realm, starting from the Common Tasks page of the OpenAM console.

### Procedure 12.1. To Set Up the OAuth 2.0 Authorization Service

Follow the steps in this procedure to set up the service with the Common Tasks wizard.

When you create the service with the Common Tasks wizard, the wizard also creates a standard policy in the top level realm (/) to protect the authorization endpoint. In this configuration OpenAM serves the resources to protect, and no separate application is involved. OpenAM therefore acts both as the policy decision point and also as the policy enforcement point that protects the OAuth 2.0 authorization endpoint.

There is no requirement to use the wizard or to create the policy in the top level realm. However if you create the OAuth 2.0 authorization service without the wizard, then you must set up the policy independently as well. The policy must appear in an application of type iPlanetAMWebAgentService, which is the default in the OpenAM policy editor. When configuring the policy allow all authenticated users to perform HTTP GET and POST requests on the authorization endpoint. The authorization endpoint is described in the *Developer's Guide* section, [OAuth 2.0 Client & Resource Server Endpoints](#). For details on creating policies, see the chapter on [Defining Authorization Policies](#).

1. In the OpenAM console, select Common Tasks > Configure OAuth2.
2. On the Configure OAuth2 page, enter the Realm for the authorization service.
3. If necessary, adjust the lifetimes for authorization codes (10 minutes is the recommended setting in RFC 6749), access tokens, and refresh tokens.
4. Select Issue Refresh Tokens unless you do not want the authorization service to supply a refresh token when returning an access token.
5. Select Issue Refresh Tokens on Refreshing Access Tokens if you want the authorization service to supply a refresh token when refreshing an access token.
6. If you want to use the default scope implementation, whereby scopes are taken to be resource owner profile attribute names, then keep the default setting.

If you have a custom scope validator implementation, put it on the OpenAM classpath, and provide the class name as Scope Implementation Class. For an example, see the *Developer's Guide* chapter, [Customizing OAuth 2.0 Scope Handling](#).

7. Click Create to complete the process.

To access the authorization server configuration in OpenAM Console, browse to Access Control > *Realm Name* > Services, and then click OAuth2 Provider.

As mentioned at the outset of this procedure, the wizard sets up a policy in the top level realm to protect the authorization endpoint. The policy appears in the iPlanetAMWebAgentService application. Its name is OAuth2ProviderPolicy.

8. If your provider has a custom response type plugin, put it on the OpenAM classpath, and then add the custom response types and the plugin class names to the list of Response Type Plugins.
9. If you use an external identity repository where resource owners log in not with their user ID, but instead with their mail address or some other profile attribute, then complete this step.

The following steps describe how to configure OpenAM authentication so OAuth 2.0 resource owners can log in using their email address, stored on the LDAP profile attribute, `mail`. Adapt the names if you use a different LDAP profile attribute, such as `cn`.

- a. When configuring the data store for the LDAP identity repository, make sure that you select Load schema when saved, and that you set the Authentication Naming Attribute to `mail`. You can find the data store configuration under Access Control > *Realm Name* > Data Stores.
- b. Add the `mail` profile attribute name to the list of attributes that can be used for authentication.

To make the change, browse to Access Control > *Realm Name* > Services > OAuth2 Provider, add the profile attributes to the list titled User Profile Attribute(s) the Resource Owner is Authenticated On, and then click Save.

- c. Create an LDAP authentication module to use with the external directory.
  - i. In OpenAM Console under Access Control > *Realm Name* > Authentication > Module Instances, create a module to access the LDAP identity repository, such as `LDAPAuthUsingMail`.
  - ii. In the Attribute Used to Retrieve User Profile field, set the attribute to `mail`.
  - iii. In the Attributes Used to Search for a User to be Authenticated list, remove the default `uid` attribute and add the `mail` attribute.
  - iv. Click Save.
- d. Create an authentication chain to include the module such as `authUsingMail`.
  - i. When creating the authentication chain, choose the `LDAPAuthUsingMail` module in the Instance drop-down list, and set the criteria to REQUIRED.
  - ii. Click Save.

- e. Set Organization Authentication Configuration to use the new chain, `authUsingMail`, and then click Save.

At this point OAuth 2.0 resource owners can authenticate using their email address rather than their user ID.

10. Set a multi-valued, string syntax profile attribute where OpenAM can store a resource owner's decisions to authorize clients without further interaction in the Saved Consent Attribute Name field.

You are not likely to find a standard profile attribute for this. For evaluation purposes only, you might try an unused existing profile attribute such as `description`.

When moving to production, however, use a dedicated, multi-valued, string syntax profile attribute that clearly is not used for other purposes. For example, you might call the attribute `oAuth2SavedConsent`.

Adding a profile attribute involves updating the identity repository to support use of the attribute, updating the AMUser Service for the attribute, and optionally allowing users to edit the attribute. The process is described in the *Developer's Guide* chapter, [Customizing Profile Attributes](#), which demonstrates adding a custom attribute when using OpenDJ directory services to store user profiles.

11. Save your changes.

You can further adjust the authorization server configuration after you create it in the OpenAM console under Access Control > *Realm Name* > Services > OAuth2 Provider.

You can adjust global defaults in the OpenAM console under Configuration > Global > OAuth2 Provider.

## 12.3 Registering OAuth 2.0 Clients With the Authorization Service

You register an OAuth 2.0 client with the OpenAM OAuth 2.0 authorization service by creating and configuring an OAuth 2.0 Client agent profile.

At minimum you must have the client identifier and client password in order to register your OAuth 2.0 client.

### Procedure 12.2. To Create an OAuth 2.0 Client Agent Profile

- Use either of these two facilities.

- In the OpenAM console, access the client registration endpoint at / oauth2/registerClient.jsp.

The full URL depends on where you deployed OpenAM. For example, <https://openam.example.com:8443/openam/oauth2/registerClient.jsp>.

The Register a Client page lets you quickly create and configure an OAuth 2.0 client in a simple web page without inline help.

- In the OpenAM console under Access Control > *Realm Name* > Agents > OAuth 2.0 Client > Agent, click New, then provide the client identifier and client password, and finally click Create to create the profile.

This page requires that you perform additional configuration separately.

### Procedure 12.3. To Configure an OAuth 2.0 Client Agent Profile

After initially registering or creating a client agent profile as necessary.

1. In the OpenAM console, browse to Access Control > *Realm Name* > Agents > OAuth 2.0 Client > Agent > *Client Name* to open the Edit *Client Name* page.
2. Adjust the configuration as needed using the inline help for hints, and also the documentation section [Configuring OAuth 2.0 & OpenID Connect 1.0 Clients](#).

Examine the client type option. An important decision to make at this point is whether your client is a confidential client or a public client. This depends on whether your client can keep its credentials confidential, or whether its credentials can be exposed to the resource owner or other parties. If your client is a web-based application running on a server, such as the OpenAM OAuth 2.0 client, then you can keep its credentials confidential. If your client is a user-agent based client, such as a JavaScript client running in a browser, or a native application installed on a device used by the resource owner, then yours is a public client.

3. When finished, Save your work.

## 12.4 Managing OAuth 2.0 Tokens

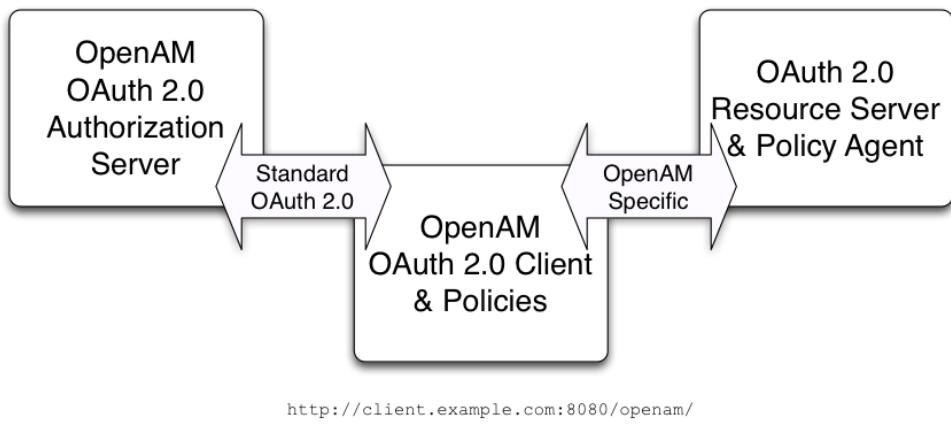
OpenAM exposes a RESTful API that lets administrators read, list, and delete OAuth 2.0 tokens. OAuth 2.0 clients can also manage their own tokens. See the *Developer's Guide* section on the [OAuth 2.0 Token Administration Endpoint](#) for details.

### 12.4.1 Configuring OpenAM as Authorization Server & Client

This section takes a high-level look at how to set up OpenAM both as an OAuth 2.0 authorization server and also as an OAuth 2.0 client in order to protect resources on a resource server by using an OpenAM policy agent.

<http://authz.example.com:8080/openam/>

<http://www.example.com:8080/examples/>



The example in this section uses three servers, <http://authz.example.com:8080/openam> as the OAuth 2.0 authorization server, <http://client.example.com:8080/openam> as the OAuth 2.0 client, which also handles policy, <http://www.example.com:8080/> as the OAuth 2.0 resource server protected with an OpenAM policy agent where the resources to protect are deployed in Apache Tomcat. The two OpenAM servers communicate using OAuth 2.0. The policy agent on the resource server communicates with OpenAM as policy agents normally do, using OpenAM specific requests. The resource server in this example does not need to support OAuth 2.0.

The high-level configuration steps are as follows.

1. On the OpenAM server that you will configure to act as an OAuth 2.0 client, configure a policy agent profile, and the policy used to protect the resources.

On the web server or application container that will act as an OAuth 2.0 resource server, install and configure the OpenAM policy agent.

Make sure that you can access the resources when you log in through an authentication module that you know to be working, such as the default DataStore authentication module.

In this example, you would try to access <http://www.example.com:8080/examples/>. The policy agent should redirect you to the OpenAM login page.

After you log in successfully as a user with access rights to the resource, OpenAM should redirect you back to <http://www.example.com:8080/examples/>, and the policy agent should allow access.

Fix any problems you have in accessing the resources before you try to set up access through the OAuth 2.0 / OpenID Connect authentication module.

2. Configure one OpenAM server as an OAuth 2.0 authorization service, which is described in [Section 12.2, “Configuring the OAuth 2.0 Authorization Service”](#).
3. Configure the other OpenAM server with the policy agent profile and policy as an OAuth 2.0 client, by setting up an OAuth 2.0 / OpenID Connect authentication module according to the section [\*Hints for the OAuth 2.0 / OpenID Connect Authentication Module\*](#).
4. On the authorization server, register the OAuth 2.0 / OpenID Connect authentication module as an OAuth 2.0 client, which is described in [Section 12.3, “Registering OAuth 2.0 Clients With the Authorization Service”](#).
5. Log out and access the protected resources to see the process in action.

### **Example 12.1. Web Site Protected With OAuth 2.0**

This example pulls everything together (except security considerations), using OpenAM servers both as the OAuth 2.0 authorization server, and also as the OAuth 2.0 client, with an OpenAM policy agent on the resource server requesting policy decisions from OpenAM as OAuth 2.0 client. In this way any server protected by a policy agent that is connected to an OpenAM OAuth 2.0 client can act as an OAuth 2.0 resource server.

1. On the OpenAM server that will be configured as an OAuth 2.0 client, set up an OpenAM policy agent and policy in the top-level realm, `/`, to protect resources.

See the [\*Web Policy Agent User's Guide\*](#) or the [\*Java EE Policy Agent User's Guide\*](#) for instructions on installing a policy agent. This example relies on the Apache Tomcat Java EE policy agent, configured to protect resources in Apache Tomcat at <http://www.example.com:8080/>.

The policies for this example protect the Apache Tomcat examples under <http://www.example.com:8080/examples/>, allowing GET and POST operations by all authenticated users. For more information on creating policies, see [\*Configuring Policies\*](#).

After setting up the policy agent and the policy, you can make sure everything is working by attempting to access a protected resource, in this case <http://www.example.com:8080/examples/>. The policy agent should redirect you to

OpenAM to authenticate with the default authentication module, where you can login as user `demo` password `changeit`. After successful authentication, OpenAM redirects your browser back to the protected resource and the policy agent lets you get the protected resource, in this case the Tomcat examples top page.

#### **Apache Tomcat Examples**

- [Servlets examples](#)
- [JSP Examples](#)
- [WebSocket Examples](#)

2. On the OpenAM server to be configured as an OAuth 2.0 authorization server, configure OpenAM's OAuth 2.0 authorization service as described in [Section 12.2, “Configuring the OAuth 2.0 Authorization Service”](#).

The authorization endpoint to protect in this example is at `http://authz.example.com:8080/openam/oauth2/authorize`.

3. On the OpenAM server to be configured as an OAuth 2.0 client, configure an OpenAM OAuth 2.0 / OpenID Connect authentication module instance for the top-level realm.

Under Access Control > / (Top-Level Realm) > Authentication > Module Instances, click New. Name the module `OAuth2`, and select the OAuth 2.0 type, then click OK to save your work.

Then click Authentication > Module Instances > OAuth2 to open the OAuth 2.0 client configuration page. This page offers numerous options. The key settings for this example are the following.

##### **Client Id**

This is the client identifier used to register your client with OpenAM's authorization server, and then used when your client must authenticate to OpenAM.

Set this to `myClientID` for this example.

##### **Client Secret**

This is the client password used to register your client with OpenAM's authorization server, and then used when your client must authenticate to OpenAM.

Set this to `password` for this example. Make sure you use strong passwords when you actually deploy OAuth 2.0.

##### **Authentication Endpoint URL**

In this example, `http://authz.example.com:8080/openam/oauth2/authorize`.

This OpenAM endpoint can take additional parameters. In particular you must specify the realm if the OpenAM OAuth 2.0 provider is configured for a subrealm rather than / (Top-Level Realm).

For example, if the OAuth 2.0 provider is configured for the realm / customers, then use the following URL: <http://authz.example.com:8080/openam/oauth2/authorize?realm=/customers>

The /oauth2/authorize endpoint can also take module and service parameters. Use either as described in [Authenticating To OpenAM](#), where module specifies the authentication module instance to use or service specifies the authentication chain to use when authenticating the resource owner.

#### Access Token Endpoint URL

In this example, [http://authz.example.com:8080/openam/oauth2/access\\_token](http://authz.example.com:8080/openam/oauth2/access_token).

This OpenAM endpoint can take additional parameters. In particular you must specify the realm if the OpenAM OAuth 2.0 provider is configured for a subrealm rather than / (Top-Level Realm).

For example, if the OAuth 2.0 provider is configured for the realm / customers, then use the following URL: [http://authz.example.com:8080/openam/oauth2/access\\_token?realm=/customers](http://authz.example.com:8080/openam/oauth2/access_token?realm=/customers)

#### User Profile Service URL

In this example, <http://authz.example.com:8080/openam/oauth2/tokeninfo>.

#### Scope

In this example, cn.

The demo user has common name demo by default, so by setting this to cn|Read your user name, OpenAM can get the value of the attribute without the need to create additional subjects, or to update existing subjects. The description, Read your user name, is shown to the resource owner in the consent page.

#### OAuth2 Access Token Profile Service Parameter name

Identifies the parameter that contains the access token value, which in this example is access\_token.

#### Proxy URL

The client redirect URL, which in this example is <http://client.example.com:8080/openam/oauth2c/OAuthProxy.jsp>.

Account Mapper

In this example, `org.forgerock.openam.authentication.modules.oauth2.DefaultAccountMapper`.

Account Mapper Configuration

In this example, `cn=cn`.

Attribute Mapper

In this example, `org.forgerock.openam.authentication.modules.oauth2.DefaultAttributeMapper`.

Attribute Mapper Configuration

In this example, `cn=cn`.

Create account if it does not exist

In this example, disable this functionality.

OpenAM can create local accounts based on the account information returned by the authorization server.

4. On the OpenAM server configured to act as an OAuth 2.0 authorization server, register the OAuth 2.0 / OpenID Connect authentication module as an OAuth 2.0 confidential client, which is described in [Section 12.3, “Registering OAuth 2.0 Clients With the Authorization Service”](#).

Under Access Control > / (Top-Level Realm) > Agents > OAuth 2.0 Client > Agents > `myClientID`, adjust the following settings.

Client type

In this example, `confidential`. OpenAM protects its credentials as an OAuth 2.0 client.

Redirection URIs

In this example, `http://client.example.com:8080/openam/oauth2c/OAuthProxy.jsp`.

Scopes

In this example, `cn`.

5. Before you try it out, on the OpenAM server configured to act as an OAuth 2.0 client, you must make the following additional change to the configuration.

Your OpenAM OAuth 2.0 client authentication module is not part of the default chain, and therefore OpenAM does not call it unless you specifically request the OAuth 2.0 client authentication module.

To cause the policy agent to request your OAuth 2.0 client authentication module explicitly, browse in OpenAM console to your *policy agent profile*

*configuration*, in this case Access Control > / (Top-Level Realm) > Agents > J2EE > Agents > *Agent Name* > OpenAM Services > OpenAM Login URL, and add `http://client.example.com:8080/openam/UI/Login?module=OAuth2`, moving it to the top of the list.

Save your work.

This ensures that the policy agent directs the resource owner to OpenAM with the instruction to authenticate using the OAuth2 authentication module.

## 6. Try it out.

First make sure you are logged out of OpenAM, for example by browsing to the logout URL, in this case `http://client.example.com:8080/openam/UI/Logout`.

Next attempt to access the protected resource, in this case `http://www.example.com:8080/examples/`.

If everything is set up properly, the policy agent redirects your browser to the login page of OpenAM with `module=OAuth2` among other query string parameters. After you authenticate, for example as user `demo`, password `changeit`, OpenAM presents you with an authorization decision page.

### **OAuth authorization page**

#### **Application requesting scope**

:

**The following private info is requested**

Save Consent:

When you click Allow, the authorization service creates an SSO session, and redirects the client back to the resource, thus allowing the client to access the protected resource. If you configured an attribute on which to store the saved consent decision, and you choose to save the consent decision for this authorization, then OpenAM can use that saved decision to avoid prompting you for authorization next time the client accesses the resource, but only ensure that you have authenticated and have a valid session.

### **Apache Tomcat Examples**

- [Servlets examples](#)
- [JSP Examples](#)
- [WebSocket Examples](#)

## 12.4.2 Configuring OpenAM as Both SAML 2.0 Service Provider & OAuth 2.0 Authorization Server

As described in [Section 12.1.1.6, “SAML 2.0 Bearer Assertion Profiles”](#), OpenAM as OAuth 2.0 authorization server can handle the profile where a SAML 2.0 assertion borne by the client functions as an authorization grant to get an access token. This lets a client get an access token when a resource owner completes SAML 2.0 Web Single Sign-On.

You can configure OpenAM as both SAML 2.0 service provider and OAuth 2.0 authorization server, using an built-in adapter class to POST assertions returned to the service provider to the access token endpoint of the authorization server. This allows clients to send a resource owner to the identity provider for SAML 2.0 web SSO, get an assertion at the service provider, and retrieve an access token from the authorization server. In other words, once this scenario is configured, the client must only direct the resource owner to start web SSO as described in [Using SAML 2.0 Single Sign-On & Single Logout](#), and then retrieve the access token on success or handle the error condition on failure.

### Procedure 12.4. To Get an Access Token From SAML 2.0 Web SSO

For this scenario to work, the following conditions must be met.

- The client must make the resource owner understand that by authenticating to the SAML 2.0 identity provider the resource owner grants the client access to the protected resources. OpenAM does not present the resource owner with an authorization decision.
- The SAML 2.0 identity provider issuing the assertion must sign the assertion, and must correctly handle the name ID for the subject.
- OpenAM as relying party must request that assertions are signed, must verify the signatures on assertions, must correctly handle name IDs from the issuer, and must use the built-in `org.forgerock.openam.oauth2.saml2.core.OAuth2Saml2GrantSPAdapter` adapter class in the service provider configuration to POST assertions to the OAuth 2.0 authorization service.
- The OAuth 2.0 authorization service and SAML 2.0 service provider must be configured together on the same OpenAM server.
- An OAuth 2.0 client configuration on OpenAM with the same name as the service provider entity ID must be set up on OpenAM.
- The OAuth 2.0 client initiating the process must be able to consume the access token and to handle errors if necessary.

Follow these steps. The test configuration hints in this procedure let you prepare configuration to test with the demo user created in OpenAM by default.

Configuring OpenAM as  
Both SAML 2.0 Service  
Provider & OAuth 2.0

- 
1. Make sure the SAML 2.0 identity provider signs assertions and that name IDs are correctly configured to map resource owner accounts.

When configuring OpenAM as a hosted identity provider follow these steps.

- a. Make sure the Signing Key is properly configured on setup.

For a test configuration, select the test certificate shown in the Common Tasks > Create Hosted Service Provider wizard.

- b. Make sure name IDs are properly configured.

For a test configuration, in the OpenAM console under Federation > Entity Providers > *IDP name* > NameID Value Map, add `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified=cn` and then Save your work.

For more detail on configuring OpenAM as a SAML 2.0 identity provider, see [Configuring Identity Providers](#).

2. Configure OpenAM as service provider.

- a. Set up a hosted service provider in OpenAM console under Common Tasks > Create Hosted Service Provider, keeping track of the name, such as `https://www.sp.example:8443/openam`, and selecting Use default attribute mapping from Identity Provider.

For details on configuring OpenAM as a SAML 2.0 service provider, see [Configuring Service Providers](#).

- b. Under Federation > Entity Providers > *SP name* > Assertion Content > Request/Response Signing, check Assertions Signed.
- c. For a test configuration, in Federation > Entity Providers > *SP name* > Assertion Content > NameID Format List, remove all but `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`, and then Save your work.
- d. In Federation > Entity Providers > *SP name* > Assertion Processing > Adapter, add `org.forgerock.openam.oauth2.saml2.core.OAuth2Saml2GrantSPAdapter`, and then Save your work.

This is the adapter class that POSTs the SAML 2.0 assertion to the OAuth 2.0 access token endpoint.

- e. Use the wizard under Common Tasks > Register Remote Identity Provider to import the identity provider metadata.

Configuring OpenAM as  
Both SAML 2.0 Service  
Provider & OAuth 2.0

- 
3. Make sure the identity provider imports the metadata for your service provider.

If your service provider is at `https://www.sp.example:8443/openam`, then the metadata can be accessed at `https://www.sp.example:8443/openam/saml2/jsp/exportmetadata.jsp`.

4. On the service provider OpenAM server, set up the OAuth 2.0 authorization server as described in [Section 12.2, “Configuring the OAuth 2.0 Authorization Service”](#).

For a test configuration, set the realm to `/`, and accept the defaults.

5. On the service provider and authorization server OpenAM server, set up an OAuth 2.0 client profile with the same name as the service provider under Access Control > *realm* > Agents > OAuth 2.0 Client > New...

For example, if the service provider name is `https://www.sp.example:8443/openam`, then that is also the name of the OAuth 2.0 client profile.

You can make additional changes to the client profile if necessary. See [Section 12.3, “Registering OAuth 2.0 Clients With the Authorization Service”](#) for details.

6. Test your configuration.

- Logout of all OpenAM servers.
- Initiate SAML 2.0 Web SSO.

For example, if your identity provider is at `https://www.idp.example:8443/openam` with meta alias `/idp` and your service provider is at `https://www.sp.example:8443/openam`, then browse to the following URL (without line breaks or spaces).

```
http://www.idp.example:8443/openam/saml2/jsp/idpSSOInit.jsp  
?metaAlias=/idp&spEntityID=http://www.sp.example:8443/openam
```

For other configurations, see [Using SAML 2.0 Single Sign-On & Single Logout](#).

- Login to the identity provider.

For OpenAM, login with user name `demo` and password `changeit`.

- Login to the service provider.

For OpenAM, login with user name `demo` and password `changeit`.

- See the resulting access token on successful login.

The result looks something like this, all on one line.

```
{  
  "expires_in": 59,  
  "token_type": "Bearer",  
  "access_token": "f0f731e0-6013-47e3-9c07-da598157a85f"  
}
```

### 12.4.3 User Consent Management

Users of OAuth 2.0 clients can now manage their OAuth 2.0 tokens on their user pages via the OpenAM console. For example, the user logs in to the OpenAM console as demo, and then clicks the Dashboard link on the Profile page. In the Authorized Apps section, the users can view their OAuth 2.0 tokens or remove them by clicking the Revoke Access link, effectively removing their consent to the application.

**Figure 12.1. OAuth2 Self-Service**

The screenshot shows a web browser window titled 'OpenAM - Dashboard'. The URL is 'openam.example.com:8080/openam/XUI/#dashboard/'. The page header includes the FORGEROCK logo, a navigation bar with 'Dashboard' selected, and links for 'demo', 'Profile', 'Change Password', and 'Log out'. The main content area has a heading 'My Applications' followed by the message 'You have no applications assigned to you'. Below this is a heading 'Authorized Apps' with a table listing two applications: 'Calendar' and 'Photo'. The table columns are 'Application', 'Scope', 'Expiry Date', and 'Revoke Access'. Both rows show 'Indefinitely' in the 'Expiry Date' column and a 'Revoke Access' link in the last column. At the bottom of the page, there is footer text: 'info@forgerock.com, or phone +47-2108-1746.' and 'Copyright © 2010-14 ForgeRock AS, all rights reserved.'

| Application | Scope             | Expiry Date  |                               |
|-------------|-------------------|--------------|-------------------------------|
| Calendar    | cn,openid,profile | Indefinitely | <a href="#">Revoke Access</a> |
| Photo       | cn                | Indefinitely | <a href="#">Revoke Access</a> |

## 12.5 Security Considerations

OAuth 2.0 messages involve credentials and access tokens that allow the bearer to retrieve protected resources. Therefore, do not let an attacker capture requests or responses. Protect the messages going across the network.

RFC 6749 includes a number of [Security Considerations](#), and also requires Transport Layer Security (TLS) to protect sensitive messages. Make sure you read the section covering *Security Considerations*, and that you can implement them in your deployment.

Also, especially when deploying a mix of other clients and resource servers, take into account the points covered in the Internet-Draft, [\*OAuth 2.0 Threat Model and Security Considerations\*](#), before putting your service into production.



---

## Chapter 13

# Managing OpenID Connect 1.0 Authorization

This chapter covers OpenAM support for OpenID Connect 1.0. [OpenID Connect](#) 1.0 is an authentication layer built on OAuth 2.0. OpenID Connect 1.0 is a specific implementation of OAuth 2.0 where the identity provider that runs the authorization server also holds the protected resource that the third-party application aims to access. This resource is the *User Info*, information about the authenticated end user expressed in a standard format. In this way OpenID Connect 1.0 allows relying parties both to verify the identity of the end user and also to obtain user information using REST. This contrasts with OAuth 2.0, which only defines the authorization mechanism.

The names used in OpenID Connect 1.0 differ from those used in OAuth 2.0. In OpenID Connect 1.0, the key entities are the following.

- The *end user* (OAuth 2.0 resource owner) whose user information the application needs to access.

The end user wants to use an application through existing identity provider account without signing up to and creating credentials for yet another web service.

- The *Relying Party* (RP) (OAuth 2.0 client) needs access to the end user's protected user information.

For example, an online mail application needs to know which end user is accessing the application in order to present the correct inbox.

As another example, an online shopping site needs to know which end user is accessing the site in order to present the right offerings, account, and shopping cart.

- The *OpenID Provider* (OP) (OAuth 2.0 authorization server and also resource server) that holds the user information and grants access.

OpenAM can play this role in an OpenID Connect deployment.

The OP effectively has the end user's consent to providing the RP with access to some of its user information. As OpenID Connect 1.0 defines unique identification for an account (subject identifier + issuer identifier), the RP can use this as a key to its own user profile.

In the case of the online mail application, this key could be used to access the mailboxes and related account information. In the case of the online shopping site, this key could be used to access the offerings, account, shopping cart and so forth. The key makes it possible to serve users as if they had local accounts.

In OpenID Connect the relying party can verify claims about the identity of the end user, and log the user out at the end of a session. OpenID Connect also makes it possible to discover the OpenID Provider for an end user, and to register relying party client applications dynamically. OpenID connect services are built on OAuth 2.0, JSON Web Token (JWT), WebFinger and Well-Known URIs.

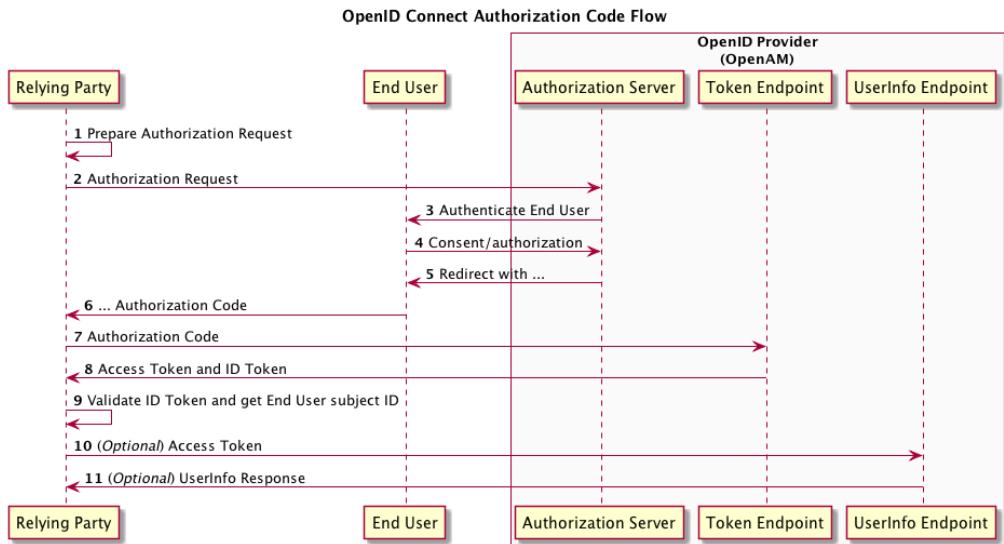
## 13.1 About OpenID Connect 1.0 Support in OpenAM

In its role as OpenID Provider, OpenAM lets OpenID Connect relying parties (clients) discover its capabilities, handles both dynamic and static registration of OpenID Connect relying parties, responds to relying party requests with authorization codes, access tokens, and user information according to the Authorization Code and Implicit flows of OpenID Connect, and manages sessions.

This section describes how OpenAM fits into the OpenID Connect picture in terms of the roles that it plays in the authorization code and implicit flows, provider discovery, client registration, and session management.

### 13.1.1 OpenID Connect Authorization Code Flow

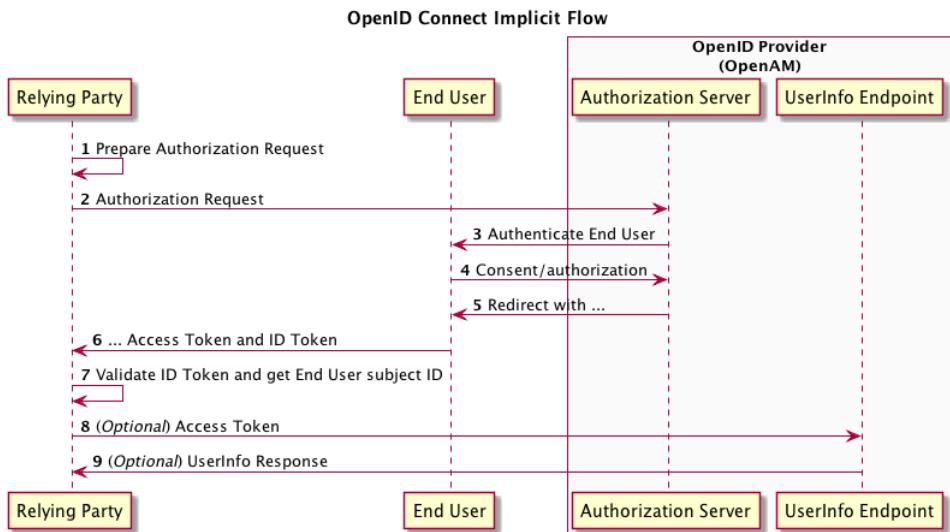
The OpenID Connect Authorization Code Flow specifies how the relying party interacts with the OpenID Provider, in this case OpenAM, based on use of the OAuth 2.0 authorization grant. The following sequence diagram shows successful processing from the authorization request, through grant of the authorization code, access token, and ID token, and optional use of the access token to get information about the end user.



In addition to what OAuth 2.0 specifies, OpenID Connect uses an ID token so the relying party can validate claims about the end user. It also defines how to get user information such as profile, email, address, and phone details from the UserInfo endpoint with a valid access token.

### 13.1.2 OpenID Connect Implicit Flow

The OpenID Connect Implicit Flow specifies how the relying party interacts with the OpenID Provider, in this case OpenAM, based on use of the OAuth 2.0 implicit grant. The following sequence diagram shows successful processing from the authorization request, through grant of the access and ID tokens, and optional use of the access token to get information about the end user.



As for the Authorization Code Flow, the Implicit Flow specifies an ID token so that the relying party can validate claims about the end user. It also defines how to get user information such as profile, email, address, and phone details from the UserInfo endpoint with a valid access token.

### 13.1.3 OpenID Connect Discovery

OpenID Connect defines how a relying party can discover the OpenID Provider and corresponding OpenID Connect configuration for an end user. The discovery mechanism relies on WebFinger to get the information based on the end user's identifier. The server returns the information in JSON Resource Descriptor (JRD) format.

### 13.1.4 OpenID Connect Relying Party Registration

OpenID Connect relying parties register OAuth 2.0 client profiles with OpenAM. Relying parties can register with OpenAM as a provider both statically, as for other OAuth 2.0 clients, and also dynamically as specified by OpenID Connect Discovery. To allow dynamic registration, you register an initial OAuth 2.0 client that other relying parties can use to get access tokens for registration.

You can also enable OpenID Connect relying parties to register dynamically without having to provide an access token. For details, see the documentation on the advanced server property, `org.forgerock.openam.openidconnect.allow.open.dynamic.registration`, in the *OpenAM Reference* section, [Servers > Advanced](#).

Take care to limit or throttle dynamic registration if you enable this capability on production systems.

### 13.1.5 OpenID Connect Session Management

OpenID Connect lets the relying party track whether the end user is logged in at the provider, and also initiate end user logout at the provider. The specification has the relying party monitor session state using an invisible iframe and communicate status using the HTML 5 postMessage API.

## 13.2 Configuring OpenAM As OpenID Provider

You can configure OpenAM's OAuth 2.0 authorization service to double as an OpenID Provider.

First, follow the instructions for [Configuring the OAuth 2.0 Authorization Service](#), making sure that the Response Type Plugins list includes at least the default plugin classes.

Next, configure the OpenID Connect specific options.

- The optional Remote JSON Web Key URL field allows you to set a URL to a [JSON Web Key set](#) with the public key(s) for the provider.

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.

- The Subject Types supported map allows you to support pairwise subject types as described in the OpenID Connect core specification section concerning [Subject Identifier Types](#).
- The ID Token Signing Algorithms supported list allows you to change the list of algorithms used sign ID Tokens.
- The Supported Claims list allows you to restrict the claims supported by OpenAM's userinfo endpoint.
- The Alias of ID Token Signing Key alias allows you to set the key pair alias for the key used to sign ID Tokens when using a signing algorithm that involves asymmetric keys.

For instructions on changing the key pair, see [To Change the Signing Key for Federation](#).

- The Allow Open Dynamic Client Registration checkbox enables relying parties to register without using an access token.

- The Generate Registration Access Tokens checkbox has OpenAM generate Registration Access Tokens for dynamic client registration when Allow Open Dynamic Client Registration is enabled. This allows the client to view and update its registration.

Finally, if your provider is part of a GSMA Mobile Connect deployment, see [Procedure 13.3, “Configuring OpenAM as an OP for Mobile Connect”](#).

## 13.3 Configuring OpenAM For OpenID Connect Discovery

In order to allow relying parties to discover the OpenID Provider for an end user, OpenAM supports OpenID Connect Discovery 1.0. In addition to discovering the OpenID Provider for an end user, the relying party can also request the OpenID Provider configuration.

OpenAM as OpenID Provider exposes two endpoints for discovery:

```
/well-known/webfinger  
/well-known/openid-configuration
```

A relying party needs to be able to discover the OpenID provider for an end user. In this case you should consider redirecting requests to URIs at the server root, such as `http://www.example.com/.well-known/webfinger` and `http://www.example.com/.well-known/openid-configuration`, to these Well-Known URIs in OpenAM's space.

Discovery relies on [WebFinger](#), a protocol to discover information about people and other entities using standard HTTP methods. WebFinger uses [Well-Known URIs](#), which defines the path prefix `/well-known/` for the URLs defined by OpenID Connect Discovery.

Unless you deploy OpenAM in the root context of a container listening on port 80 on the primary host for your domain, relying parties need to find the right `host:port/deployment-uri` combination to locate the well-known endpoints. Therefore you must manage the redirection to OpenAM. If you are using WebFinger for something else than OpenID Connect Discovery, then you probably also need proxy logic to route the requests.

To retrieve the OpenID Provider for an end user, the relying party needs the following.

`host`

The server where the relying party can access the WebFinger service.

Notice that this is a host name rather than a URL to the endpoint, which is why you might need to redirect relying parties appropriately as described above.

### resource

Identifies the end user that is the subject of the request.

The relying party must percent-encode the resource value when using it in the query string of the request, so when using the "acct" URI scheme and the resource is `acct:user@example.com`, then the value to use is `acct%3Auser%40example.com`.

### rel

URI identifying the type of service whose location is requested.

In this case `http://openid.net/specs/connect/1.0/issuer`, which is `http%3A%2F%2Fopenid.net%2Fspecs%2Fconnect%2F1.0%2Fissuer`.

Ignoring the question of redirection, you can test the endpoint for the demo user account (output lines folded to make them easier to read).

```
$ curl \
"https://openam.example.com:8443/openam/.well-known/webfinger"\
?resource=acct%3Ademo%40example.com\
&rel=http%3A%2F%2Fopenid.net%2Fspecs%2Fconnect%2F1.0%2Fissuer"
{
  "subject": "acct:demo@example.com",
  "links": [
    {
      "rel": "http://openid.net/specs/connect/1.0/issuer",
      "href": "https://openam.example.com:8443/openam"
    }
  ]
}
```

This shows that the OpenID Provider for the OpenAM demo user is indeed the OpenAM server.

The relying party can also discover the OpenID provider configuration. Ignoring the question of redirection, you can test this (output lines folded to make them easier to read).

```
$ curl https://openam.example.com:8443/openam/.well-known/openid-configuration
{
  "response_types_supported": [
    "token id_token",
    "code token",
    "code token id_token",
    "token",
    "code id_token",
    "code",
    "id_token"
  ],
  "id_token_signed_response_alg": "RS256",
  "id_token_encrypted_response_alg": "RSA1_256",
  "id_token_encrypted_response_enc": "A128CBC-HS256"
}
```

## Registering OpenID Connect Relying Parties

```
"registration_endpoint": "https://openam.example.com:8443/openam/oauth2/connect/register",
"token_endpoint": "https://openam.example.com:8443/openam/oauth2/access_token",
"end_session_endpoint": "https://openam.example.com:8443/openam/oauth2/connect/endSession",
"version": "3.0",
"userinfo_endpoint": "https://openam.example.com:8443/openam/oauth2/userinfo",
"subject_types_supported": [
    "public"
],
"issuer": "https://openam.example.com:8443/openam",
"jwks_uri": "https://openam.example.com:8443/openam/oauth2/connect/jwk_uri
?realm=",
"id_token_signing_alg_values_supported": [
    "HS256",
    "HS512",
    "RS256",
    "HS384"
],
"check_session_iframe": "https://openam.example.com:8443/openam/oauth2/connect/checkSession",
"claims_supported": [
    "phone",
    "email",
    "address",
    "openid",
    "profile"
],
"authorization_endpoint": "https://openam.example.com:8443/openam/oauth2/authorize"
}
```

When the OpenID Provider is configured in a subrealm, then relying parties can get the configuration by passing the realm as a query string parameter, as in <https://openam.example.com:8443/openam/.well-known/openid-configuration?realm=realm-name>.

### 13.4 Registering OpenID Connect Relying Parties

OpenID Connect relying parties can register with OpenAM both statically through OpenAM console for example, and also dynamically using OpenID Connect 1.0 Dynamic Registration.

#### Procedure 13.1. To Register a Relying Party With OpenAM Console

Registering a relying party by using the OpenAM console consists of first creating an OAuth 2.0 Client agent profile, and then editing the profile for the settings pertinent to OpenID Connect 1.0.

1. In the OpenAM console under Access Control > *Realm Name* > Agents > OAuth 2.0 Client > Agent, click New..., then provide the client identifier and client password, and finally click Create to create the profile.
2. Follow the hints in the section, [\*Configuring OAuth 2.0 & OpenID Connect 1.0 Clients\*](#) to edit the profile to match the relying party configuration.

In order to read and edit the relying party profile dynamically later without using OpenAM console, be sure to set an access token in the Access Token field.

### Procedure 13.2. To Register a Relying Party Dynamically

For dynamic registration you need the relying party profile data, and an access token to write the configuration to OpenAM by HTTP POST. To obtain the access token, register an initial client statically after creating the provider, as described in [Procedure 13.1, “To Register a Relying Party With OpenAM Console”](#). Relying parties can then use that client to obtain the access token needed to perform dynamic registration.

#### Tip

As described in [Section 13.1.4, “OpenID Connect Relying Party Registration”](#), You can allow relying parties to register without having an access token by setting the advanced server property, org.forgerock.openam.openidconnect.allow.open.dynamic.registration, to true. When using that setting in production systems, take care to limit or throttle dynamic registration.

On successful registration, OpenAM responds with information including an access token to allow the relying party subsequently to read and edit its profile.

1. Register an initial OAuth 2.0 client statically with a client ID such as masterClient and client secret such as password.
2. Obtain an access token using the client you registered.

For example, if you created the client as described in the previous step, and OpenAM administrator amadmin has password password, you can use the OAuth 2.0 resource owner password grant as in the following example.

```
$ curl \
--request POST \
--user "masterClient:password" \
--data "grant_type=password&username=amadmin&password=password" \
https://openam.example.com:8443/openam/oauth2/access_token
{
  "expires_in": 59,
  "token_type": "Bearer",
  "refresh_token": "26938cd0-6870-4e31-ade9-df31afc37ee1",
  "access_token": "515d6551-4512-4279-98b6-c0ef3f03a722"
}
```

3. HTTP POST the relying party registration profile to the /oauth2/connect/register endpoint, using bearer token authorization with the access token you obtained from OpenAM.

For an example written in JavaScript, see the registration page in the examples [available online](#). Successful registration shows a response that includes the client ID and client secret. Lines are folded in the following example.

```
{  
    "issued_at": 1392364349,  
    "expires_at": 0,  
    "client_secret": "7f446ca9-3f1f-48fb-bf8c-150b9e643f29",  
    "redirect_uris": [  
        "https://openam.example.com:8443/openid/cb-basic.html",  
        "https://openam.example.com:8443/openid/cb-implicit.html"  
    ],  
    "registration_access_token": "515d6551-4512-4279-98b6-c0ef3f03a722",  
    "client_id": "6e4abd50-3f03-41dc-b807-c6705c3e45d7",  
    "registration_client_uri":  
        "https://openam.example.com:8443/openam/oauth2/connect/register  
        ?client_id=6e4abd50-3f03-41dc-b807-c6705c3e45d7"  
}
```

## 13.5 Managing User Sessions

OpenID Connect Session Management 1.0 allows the relying party to manage OpenID Connect sessions, making it possible to know when the end user should be logged out.

As described in the [OpenID Connect Session Management 1.0](#) specification, OpenAM's OpenID Provider exposes both a "check\_session\_iframe" URL that allows the relying party to receive notifications when the end user's session state changes at the provider, and also an "end\_session\_endpoint" URL to which to redirect an end user for logout.

When registering your relying party that uses session management, you set the OAuth 2.0 client agent profile properties Post Logout Redirect URI and Client Session URI, described in [Configuring OAuth 2.0 & OpenID Connect 1.0 Clients](#). The Post Logout Redirect URI is used to redirect the end user user-agent after logout. The Client Session URI is the relying party URI where OpenAM sends notifications when the end user's session state changes.

## 13.6 Relying Party Examples

OpenID Connect Authorization Code Flow and Implicit Flow define how clients interact with the provider to obtain end user authorization and profile

information. Although you can run the simple example relying parties that are mentioned in this section without setting up Transport Layer Security, do not deploy relying parties in production without securing the transport.

Code for the relying party examples shown here is [available online](#). Clone the example project to deploy it in the same web container as OpenAM. Edit the configuration at the outset of the .js files in the project, register a corresponding profile for the example relying party as described in [Section 13.4, “Registering OpenID Connect Relying Parties”](#), and browse the deployment URL to see the initial page.

### **OpenID Connect Client Profiles**

[OpenID Connect 1.0](#) defines two client profiles.

#### *Basic Client Profile*



The Basic Client Profile is designed for web-based relying parties that use the OAuth 2.0 Authorization Code grant type, such as server-side clients that can protect their client credentials.

Try the [Basic Client Profile](#).

#### *Implicit Client Profile*

The Implicit Client Profile is designed for relying parties that use the OAuth 2.0 Implicit grant type, such as browser-based clients written in JavaScript.

Try the [Implicit Client Profile](#).

The examples provided here are both written in JavaScript. Neither aims to protect anything, but instead to show you the steps that each kind of client follows, and the responses from OpenAM as OpenID Connect Provider.

### **OpenID Connect Dynamic Registration**

OpenID Connect 1.0 defines mechanisms both to discover the provider configuration, and also to register client applications dynamically.

See the [example registration page](#) for details.

### 13.6.1 Authorization Code Flow Example

OpenID Connect Authorization Code Flow is designed for web-based relying parties that use the OAuth 2.0 Authorization Code grant type. This grant type makes it possible for the relying party to get the access code by using the authorization code directly, without passing through the end user's browser. To protect its client secret (password), part of the relying party must run on a server.

In the example, the Basic Client Profile Start Page describes the prerequisite configuration, which must be part of the relying party profile that is stored in the OpenAM realm where you set up the OpenID Provider. In OpenAM console, check that the OAuth 2.0 client profile matches the settings described.

#### Basic Client Profile Start Page

Try OpenAM as an OpenID Connect provider using the [Basic Client Profile](#).



OpenID Connect Basic Client Profile 1.0 is designed for web-based relying parties that use the OAuth 2.0 Authorization Code grant type. This grant type makes it possible for the client to get the access code by using the authorization code directly, without passing through the end user's browser. To protect its client secret (password), part of the client must run on the server.

Note: This example is not designed to protect the client secret, but instead to show the results at each step in the process.

---

#### Prerequisite Configuration

OpenAM should be running and configured as an OpenID Connect Provider in the same container as this application.

Current settings in `basic.js`:

```
OpenAM URI      /openam
client_id       myClientID
client_secret   password
redirect_uri    http://openam.example.com:8088/openid/cb-basic.html
```

In OpenAM, create an OAuth 2.0 agent using the `client_id`, `client_secret`, and `redirect_uri`, and then edit the configuration to add the scopes "openid" and "profile".

After you have configured everything, log out of OpenAM. Then click the link to start the authorization process.

[Start authorization](#)

## Authorization Code Flow Example

---

Logout of OpenAM, and click the link at the bottom of the page to request authorization. The link sends an HTTP GET request asking for `openid profile` scopes to the OpenID Provider authorization URI.

If everything is configured correctly, OpenAM's OpenID Provider has you authenticate as an end user, such as the demo user with username `demo` and password `changeit`, and grant (Allow) the relying party access to your profile.

If you successfully authenticate and allow the example relying party access to your profile, OpenAM returns an authorization code to the example relying party. The example relying party then uses the authorization code to request an access token and an ID token. It shows the response to that request. It also validates the ID token signature using the default (HS256) algorithm, and decodes the ID token to validate its content and show it in the output. Finally it uses the access token to request information about the end user who authenticated, and displays the result.

**Basic Client Profile Response Page**Concerning signature validation, see [common.js](#).**Authorization Code**

af2e8ee5-6603-46c7-a29e-3a27cae2e467

**Token Response**

```
{
  "scope": "openid profile",
  "expires_in": 59,
  "token_type": "Bearer",
  "refresh_token": "874acf87-ce77-4e66-af1d-b44adefafa9d7",
  "id_token": "eyAidHlwIjogImp3dCIsICJhbGciOiAiSFMyNTYiLCaiY3R5IjogIkpxVCIfQ.ey
Aidg9rZW50ZW1lIjogImlkX3Rva2VuIiwgImV4cCI6IDE0MTMxOTg0MzYsICJzdWIiOiAiZGVtbyIsIC
JhenAiOiaiAibXldbG1lbR5cgUiOiaiSlduUVg9rZW4iLCACmVhbg0iAiLyIsIC
Jpc3MiOiaiAHR0cDovL29wZW5hbSSleGFTcGx1LmNvbT04MDg4L29wZWShbsIsICJhdQioibBICJteU
NsawVudELEIiBdLCAiawFOIjogMTqxMzE5zgznIwgf0aC16IDE0MTMxOTc4MzYgfQ.TF6JJwIdgAb
9Vxhq0gwC91_YjoCrUlXnbkHZTv5yN0",
  "access_token": "76b56892-a9f9-4ade-acf1-3c375ff6bee4a"
}
```

**Decoded ID Token Header**

```
{
  "typ": "jwt",
  "alg": "HS256",
  "cty": "JWT"
}
```

**Decoded ID Token Content**

```
{
  "tokenName": "id_token",
  "exp": 1413198436,
  "sub": "demo",
  "azp": "myClientID",
  "tokenType": "JWTToken",
  "realm": "/",
  "iss": "http://openam.example.com:8088/openam",
  "aud": [
    "myClientID"
  ],
  "iat": 1413197836,
  "ath": 1413197836
}
```

Notice that in addition to the standard payload, the ID token indicates the end user's OpenAM realm, in this case "realm": "/".

### 13.6.2 Implicit Flow Example

OpenID Connect Implicit Flow is designed for relying parties that use the OAuth 2.0 Implicit grant type. This grant type is designed for relying parties implemented in a browser. Rather than protect a client secret, the client profile must register a protected redirect URI in advance with the OpenID Provider.

In the example, the Implicit Client Profile Start Page describes the prerequisite configuration, which must be part of the relying party profile that is stored in

the OpenAM realm where you set up the OpenID Provider. In OpenAM console, check that the OAuth 2.0 client profile matches the settings described. If you have already configured the agent profile for the Authorization Code Flow example then you still need to add the redirect URI for the Implicit Flow.

### Implicit Client Profile Start Page

This example tries OpenAM as an OpenID Connect provider using the [Implicit Client Profile](#).



OpenID Connect Implicit Client Profile 1.0 is designed for relying parties that use the OAuth 2.0 Implicit grant type. This grant type is designed for clients implemented in a browser. Rather than protect a client secret, the client profile must register a protected redirect URI in advance with the OpenID Provider.

---

### Prerequisite Configuration

OpenAM should be running and configured as an OpenID Connect Provider in the same container as this application.

Current settings for this example:

```
OpenAM URI      /openam
client_id       myClientID
redirect_uri    http://openam.example.com:8088/openid/cb-implicit.html
```

In OpenAM, create an OAuth 2.0 agent using the `client_id`, then edit the configuration to add the `redirect_uri`, and scopes "openid" and "profile".

After you have configured everything, log out of OpenAM. Then click the link to start the authorization process.

#### [Start authorization](#)

Logout of OpenAM, and click the link at the bottom of the page to request authorization. The link sends an HTTP GET request asking for `id_token` token response types and `openid profile` scopes to the OpenID Provider authorization URI.

If everything is configured correctly, OpenAM's OpenID Provider has you authenticate as an end user, such as the demo user with username `demo` and password `changeit`, and grant (Allow) the relying party access to your profile.

If you successfully authenticate and allow the example relying party access to your profile, OpenAM returns the access token and ID token directly in the fragment (after #) of the redirect URI. The relying party does not get an authorization code. The relying party shows the response to the request. It also validates the ID token signature using the default (HS256) algorithm, and decodes the ID token to validate its content and show it in the output. Finally the relying party uses the access token to request information about the end user who authenticated, and displays the result.

### Implicit Client Profile Response Page

Concerning signature validation, see [common.js](#).



### Response From Provider

```
{  
    "state": "af0ifjlsdkj",  
    "token_type": "Bearer",  
    "expires_in": "59",  
    "id_token": "eyAidHlwIjogImp3CisICJhbGciOiAisFMyNTYiilCaiY3R5iJogIkpxVC1gfQ.ey  
AiIdG9rZW50YWl1IjogImlkX3Rva2VuIiwgImV4cC16DE0MTMxoTg1NzUsICJzdWIoiiaiZGVbyIsIC  
JhenAiOiaibXLDb6llbnRJRCIsICJ0b2tlbR5cGUioiaiSlduUVG9rZW4iLCaihm9uY2UiOiaibioUz  
ZfV3pBMk1qIiwgInJ1YWxtIjogIi8iLCaiaxXnZIjogImh0dHA6Ly9vcGVuYWouZXhbXBsZS5jb206OD  
A4OC9vcgCVuYWOiLCaIYXvKfjogWyaiBx1DbG1lbnRJRC1gKSwgImhdCIIDE0MTMxoTc5NzUsICJhdG  
giOiaxNDEzMTk3OTc1LCaih3BzIjogIkFRSUM1d0y0TFk0U2ZjeXNTSFJhdVVwaXBaF1BLdmIlal1R0nO  
ZNb1NYUm1RWnNB1pBQUpUU1FBQ01ERUFbBESMOUJNM016TTBNemcxT1RNek9UY3pNREk0TORjMs0iIH  
0.RAdkQikbpS9mrr9hCKOUwesUGjrCfy2B6RTLT3ucU",  
    "access_token": "308eec43-bdf2-4f72-a6b8-ac072cce526c"  
}
```

### Decoded ID Token Header

```
{  
    "typ": "jwt",  
    "alg": "HS256",  
    "cty": "JWT"  
}
```

### Decoded ID Token Content

```
{  
    "tokenName": "id_token",  
    "exp": 1413198575,  
    "sub": "demo",  
    "azp": "myclientID",  
    "tokenType": "JWTToken",  
    "nonce": "n-096_WzA2Mj",  
    "realm": "/",  
    "iss": "http://openam.example.com:8088/openam",  
    "aud": [  
        "myClientID"  
    ],  
    "iat": 1413197975,  
    "ath": 1413197975,  
    "ops": "AQIC5wMjLY4SfcyeSHRGUUpipl0PKvb5kTh7FMnSXRMQZsA.*AAJTSQACMDEAA1NLABM3M  
zM0MzglOTMzMDI40Dc1*"  
}
```

As for the Authorization Code Flow example, the ID Token indicates the end user's OpenAM realm and OpenAM token ID in addition to the standard information.

## 13.7 Security Considerations

As for other OAuth 2.0 applications, you must protect messages going across the network. OpenID Connect 1.0 requires Transport Layer Security (TLS). The chapter on [Managing Certificates](#) includes some discussion of protecting traffic in the container where OpenAM runs. Also see the documentation for your web application container.

Also take into account the points developed in the section on [Security Considerations](#) in the OpenID Connect Messages draft specification.

## 13.8 Using OpenAM with Mobile Connect

[GSMA Mobile Connect](#) is an application of OpenID Connect (OIDC). Mobile Connect builds on OIDC to facilitate use of mobile phones as authentication devices independently of the service provided and independently of the device used to consume the service. Mobile Connect thus offers a standard way for Mobile Network Operators to act as general-purpose identity providers, providing a range of levels of assurance and profile data to Mobile Connect-compliant Service Providers.

This section includes an overview, as well as the following.

- [Table 13.1, “Authorization Request Parameters”](#)
- [Table 13.2, “ID Token Properties”](#)
- [Procedure 13.3, “Configuring OpenAM as an OP for Mobile Connect”](#)

In a Mobile Connect deployment OpenAM can play the OpenID Provider role, implementing the Mobile Connect Profile as part of the Service Provider - Identity Gateway interface.

OpenAM can also play the Authenticator role as part of the Identity Gateway - Authenticators interface. In this role, OpenAM serves to authenticate users at the appropriate Level of Assurance (LoA). In Mobile Connect, LoAs represent the authentication level achieved. A Service Provider can request LoAs without regard to the implementation, and the Identity Gateway includes a claim in the ID Token that indicates the LoA achieved.

In OpenAM, Mobile Connect LoAs map to an authentication mechanism. Service Providers acting as OpenID Relying Parties (RP) request an LoA by using the "acr\_values" field in an OIDC authentication request. In OIDC, "acr\_values" specifies Authentication Context Class Reference values. The RP sets "acr\_values" as part of the OIDC Authentication Request. OpenAM returns the corresponding "acr" claim in the Authentication Response as the value of the ID Token "acr" field.

OpenAM as OP supports LoAs 1 (low - little or no confidence), 2 (medium - some confidence, as in single-factor authentication), and 3 (high - high confidence, as in multi-factor authentication), though out of the box it does not include support for 4, which involves digital signatures.

As Mobile Connect OP, OpenAM supports mandatory request parameters, and a number of optional request parameters.

**Table 13.1. Authorization Request Parameters**

| Request Parameter | Support   | Description   |
|-------------------|-----------|---|
| response_type     | Supported | OAuth 2.0 grant type to use. Set this to code for the authorization grant.  |
| client_id         | Supported | Set this to the client identifier.  |
| scope             | Supported | Space delimited OAuth 2.0 scope values.<br><br>Required: openid   |
|                   |           | Optional: profile, email, address, phone, offline_access  |
| redirect_uri      | Supported | OAuth 2.0 URI where the authorization request callback should go. Must match the redirect_uri in the client profile that you registered with OpenAM.                      |
| state             | Supported | Value to maintain state between the request and the callback. Required for Mobile Connect.  |
| nonce             | Supported | String value to associate the client session with the ID Token. Optional in OIDC, but required for Mobile Connect.  |
| display           | Supported | String value to specify the user interface display.   |
| login_hint        | Supported | String value indicating the the ID to use for login.  |
|                   |           | When provided as part of the OIDC Authentication Request, the login_hint is set as the value of a cookie named oidcLoginHint, which is an HttpOnly cookie (only sent over |

| <b>Request Parameter</b> | <b>Support</b> | <b>Description</b>   |
|--------------------------|----------------|--|
| acr_values               | Supported      | HTTPS). Authentication modules can then retrieve the cookie's value. Authentication Context class Reference values used to communicate acceptable LoAs.  |
| dtbs                     | Not supported  | When the OIDC relying party on the server provider supplies acr_values in the authorization request, OpenAM uses the OP configuration to map the values to authentication chains. It runs through the list of acr_values in order, attempting to use the first authentication chain that matches. OpenAM then returns the authentication chain used as the value of the ID token "acr" claims property. In this way the relying part on the service provider can determine the LoA achieved during authentication.<br><br>Data To Be Signed<br><br>At present OpenAM does not support LoA 4. |

As Mobile Connect OP, OpenAM responds to a successful authorization request with a response containing all the required fields, and also the optional "expires\_in" field. OpenAM supports the mandatory ID Token properties, though the relying party is expected to use the "expires\_in" value, rather than specifying max\_age as a request parameter.

**Table 13.2. ID Token Properties**

| <b>Request Parameter</b> | <b>Support</b> | <b>Description</b>  |
|--------------------------|----------------|---|
| iss                      | Supported      | Issuer identifier   |
| sub                      | Supported      | Subject identifier  |
| aud                      | Supported      | By default OpenAM returns the identifier from the user profile.<br><br>Audience, an array including the client_id |

| <b>Request Parameter</b> | <b>Support</b> | <b>Description</b>  |
|--------------------------|----------------|---|
| exp                      | Supported      | Expiration time in seconds since the epoch  |
| iat                      | Supported      | Issued at time in seconds since the epoch   |
| nonce                    | Supported      | The nonce supplied in the request   |
| at_hash                  | Supported      | Base64url encoding of the SHA-256 hash of the "access_token" value  |
| acr                      | Supported      | Authentication Context class Reference for the LoA achieved<br><br>For example, if the request specifies acr_values=loa-3 loa-2 and OpenAM achieves LoA 2, then the ID token includes "acr": "loa-2". |
| amr                      | Supported      | Authentication Methods Reference to indicate the authentication method<br><br>OpenAM maps these to authentication modules.  |
|                          |                | Suggested values include the following: OK, DEV_PIN, SIM_PIN, UID_PWD, BIOM, HDR, OTP   |
| azp                      | Supported      | Authorized party identifier, which is the client_id   |

In addition to the standard OIDC user information returned with userinfo, OpenAM as OP for Mobile Connect returns the "updated\_at" property, representing the time last updated as seconds since the epoch.

### Procedure 13.3. Configuring OpenAM as an OP for Mobile Connect

You configure OpenAM as an OpenID Provider for Mobile Connect by changing the OAuth2 Provider configuration.

In OpenAM console, the OAuth2 Provider configuration is under Access Control > *Realm Name* > Services > OAuth2 Provider for the configuration in a specific realm, and under Configuration > Global > OAuth2 Provider for the inherited global settings.

1. Configure OpenAM as an OpenID Provider as described in [Section 13.2, "Configuring OpenAM As OpenID Provider"](#).

2. For the OpenID Connect acr\_values to Auth Chain Mapping, configure the mapping between acr\_values in the authorization request and OpenAM authentication chains.

For example, if the relying party request includes acr\_values=loa-3 loa-2 and the map includes [loa-2]=ldapService, and [loa-3]=msisdnAndHotpChain, then the authentication chain for the request is msisdnPlusHotpChain.

The **ssoadm** attribute is forgerock-oauth2-provider-loa-mapping.

3. For the OpenID Connect default acr claim, set the "acr" claim value to return in the ID Token when falling back to the default authentication chain.

The **ssoadm** attribute is forgerock-oauth2-provider-default-acr.

4. For the OpenID Connect id\_token amr values to Auth Module mappings, set the "amr" values to return in the ID Token after successfully authenticating with specified authentication modules.

For example, you could set [UID\_PWD]=LDAP to return "amr": [ "UID\_PWD" ] in the ID Token after authenticating with the LDAP module.

The **ssoadm** attribute is forgerock-oauth2-provider-amr-mappings.

5. Configure the identity Data Store attributes used to return "updated\_at" values in the ID Token.

For Mobile Connect clients, the user info endpoint returns "updated\_at" values in the ID Token. If the user profile has never been updated "updated\_at" reflects creation time.

The "updated\_at" values are read from the profile attributes you specify. When using OpenDJ directory server as an identity Data Store, the value is read from the `modifyTimestamp` attribute, or the `createTimestamp` attribute for a profile that has never been modified.

The **ssoadm** attribute for Modified Timestamp attribute name is forgerock-oauth2-provider-modified-attribute-name.

The **ssoadm** attribute for Created Timestamp attribute name is forgerock-oauth2-provider-created-attribute-name.

In addition, you must also add these attributes to the list of LDAP User Attributes for the data store. Otherwise the attributes are not returned when OpenAM reads the user profile. To edit the list in OpenAM console, browse to Access Control > *Realm Name* > Data Stores > *Data Store Name* > LDAP User Attributes.

## Using OpenAM with Mobile Connect

---

A simple, non-secure GSMA Mobile Connect relying party example is [available online](#).

---

## Chapter 14

# Managing SAML 1.x Single Sign-On

SAML 1.x describes an XML and SOAP based framework that allows online trusted partners to exchange security information. In particular, SAML 1.x defines mechanisms for browser based web single sign-on (SSO) across independent organizations that work together to permit SSO for access to resources.

## Important

Although not strictly compatible with SAML 1.x, SAML 2.0 extends SAML 1.x to several additional use cases and also clarifies how partners share metadata with each other. Unless you are integrating with an existing SAML 1.x deployment consider using SAML 2.0, or an alternative such as OAuth 2.0 or OpenID Connect 1.0, instead.

See the following chapters for more information: [Managing SAML 2.0 Federation](#), [Managing OAuth 2.0 Authorization](#), and [Managing OpenID Connect 1.0 Authorization](#).

## 14.1 About SAML 1.x

SAML 1.x was defined in response to several technical problems.

- Web SSO solutions often use SSO session cookies. Browsers do not return cookies across domains. For instance, browsers do not return cookies set by

servers in the example.com domain to servers in the example.net domain. SAML 1.x works around this limitation of HTTP cookies.

- Before SAML 1.x was defined, there were already proprietary SSO solutions, but the solutions did not interoperate well across domains.

SAML 1.x specifies a standard, cross-domain, interoperable SSO mechanism that works together with proprietary SSO services in a particular domain.

- Before SAML 1.x was defined, there was not an easy way to communicate security attributes across organization boundaries.

SAML 1.x simplifies the communication of security attributes between different organizations.

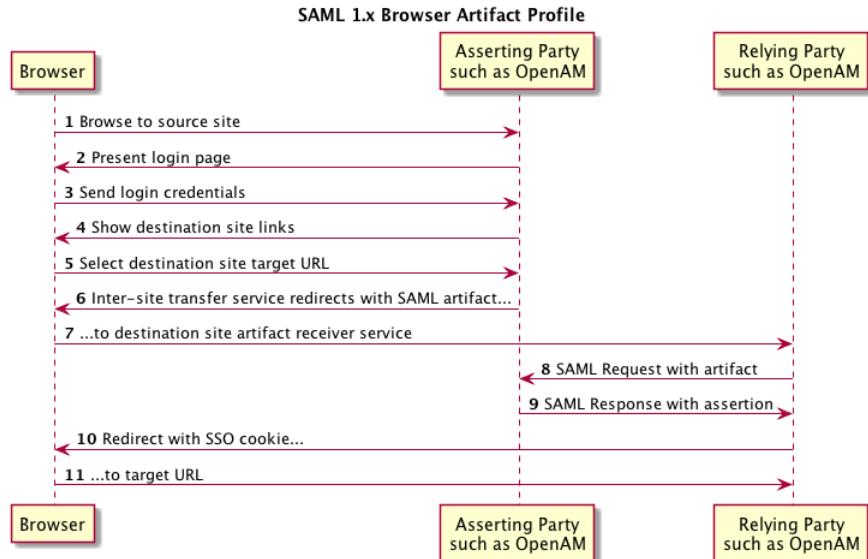
In SAML 1.x, business partners can play two roles. The *asserting party*, also known as the SAML authority and whose domain is the Source site, authenticates users and asserts information about them. The *relying party*, whose domain is known as the Destination site, consumes assertions and uses information from the assertion to decide whether to allow access to resources.

In the Web Browser SSO Profiles for SAML 1.x, the user generally starts by authenticating with the asserting party and then selecting a relying party link to browse. Alternatively, the "Destination-Site-First" scenario can start with the user browsing to the relying party's site and being redirected to the asserting party's site to authenticate.

The SAML 1.x *Inter-site Transfer Service* is a service that redirects the authenticated user from the asserting party's site to the appropriate service on the relying party's site. The Inter-site Transfer Service also handles artifact and redirect generation. How this service transfers the user to the relying party's site depends on how the asserting party and the relying party exchange messages.

The asserting party and relying party can exchange messages either by reference, where the asserting party sends an *artifact* (a base64 encoded reference to the assertion) as a query string parameter value, or by value, where the asserting party directs the user's browser to HTTP POST the assertion to the relying party.

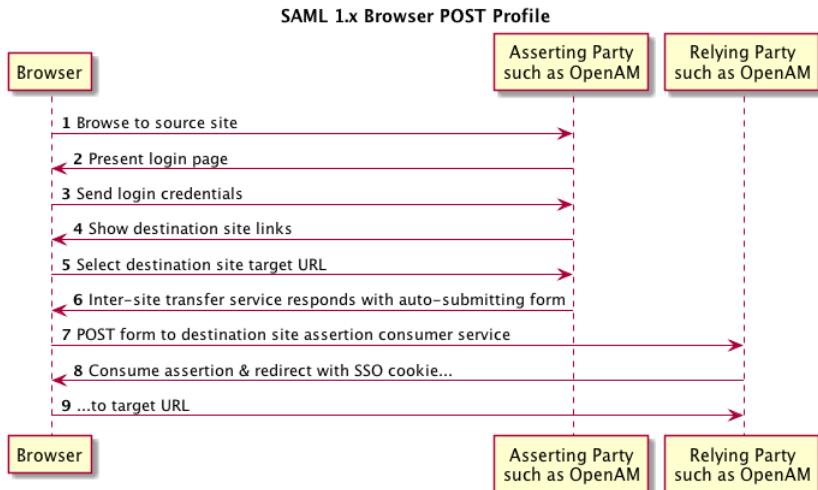
When the asserting party and relying party use artifacts, the Inter-site Transfer Service redirects the user's browser to the relying party's Artifact Receiver Service with the artifact as the value of a query string parameter. The relying party retrieves the artifact from the query string, and then sends a SAML Request to the Responder Service at the asserting party. The asserting party replies with a SAML Response containing one or more assertions.

**Figure 14.1. SAML 1.x Web SSO Browser Artifact Profile**

See section 4.1.1 of the [SAML 1.1 technical overview](#) for more detail.

When the assertion is sent using the Browser/POST Profile, the Inter-site Transfer Service responds to the user's browser with an auto-submitting form containing the SAML response. The browser then submits the SAML response as form data by HTTP POST to the relying party's Assertion Consumer Service. The relying party's Assertion Consumer Service then processes the assertion.

**Figure 14.2. SAML 1.x Web SSO Browser POST Profile**



See section 4.2.1 of the [SAML 1.1 technical overview](#) for more detail.

The Assertion Consumer Service at the relying party validates the digital signature on the SAML response, and redirects the browser to the target URL of the resource that the user is attempting to access. The server providing that resource uses the relying party's authorization decision capabilities to establish whether the user can access the resource. If so, the resource is returned to the user's browser. If the relying party is using OpenAM for example, then the relying party sets an OpenAM SSO token based on the SAML response, and this token is used to track the user's session for authorization.

Organizations working together to achieve SAML 1.x web SSO are called *trusted partners* in this context. Trusted partners agree on which services they provide, which web SSO profiles they implement, and how information is exchanged in the assertions, including profile attribute values. Once the trusted partners have reached agreement on how they interact, you can collect information about your partners' configurations and configure OpenAM to match your organization's part of the agreement.

## 14.2 Gathering Configuration Information

Before you can configure OpenAM to allow web SSO with trusted partners, you must first gather information about the agreement itself, as well as information for your site and for your partners sites.

This section lists the data that you must collect.

- SAML protocol version to use (1.1 or 1.0; default: 1.1)
- Assertion version to use (1.1 or 1.0; default: 1.1)
- Which trusted partners play which roles (asserting party, relying party)
- Domain names of partner sites (such as `example.com`, `example.net`)
- Whether assertions are exchanged by SAML artifact or by HTTP POST

If assertions are exchanged by artifact, also gather this information:

- SAML artifact parameter name (default: `SAMLart`)
- Artifact timeout
- URL to the relying party endpoint that receives the artifact (such as `https://rp.example.com/openam/SAMLAwareServlet`)
- Relying party hosts that consume artifacts (by IP addresses, DNS names, or certificate aliases)
- URL to the asserting party endpoint that responds to SAML requests (such as `https://ap.example.net/openam/SAMLSOAPReceiver`)
- Authentication credentials to connect to the asserting party endpoint, if any (such as the username and password for HTTP Basic authentication)
- Asserting party signing certificate alias

If assertions are exchanged by HTTP POST, also gather this information:

- URL to the relying party endpoint that consumes the form data in the POST assertion (such as `https://rp.example.com/openam/SAMLP0STProfileServlet`)
  - Asserting party host:port issuing assertions
  - Asserting party signing certificate alias
- Whether the relying party sends SOAP query requests to the asserting party, for example to get authorization decisions

If the relying party queries the asserting party, also gather this information:

- Relying party hosts that consume artifacts (by IP addresses, DNS names, or certificate aliases)
- How to get SSO information, and to map partner actions to authorization decisions

- Asserting party host:port issuing assertions
- Asserting party signing certificate alias
- Target specifier parameter name (default: TARGET)
- Assertion timeout
- Whether to digitally sign assertions, requests, responses
- Partners' public key certificates used for HTTPS
- Partners' public key certificates used for message signing (unless included on the KeyInfo element of signed messages)
- Partners' Site IDs (base64-encoded ID, such as XARFfsIAXeLX8BEWNIJg9Q8r0PE=)
- What NameID formats are used to exchange names (such as urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress)
- How attributes map from an assertion to an OpenAM profile (for example, urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress|EmailAddress=mail)

For more information about your own public key certificates, see [Section 14.3, “Preparing To Secure SAML 1.x Communications”](#).

For your own Site ID, see the following procedure.

#### **Procedure 14.1. To Generate a Site Identifier For an OpenAM Site**

Trusted partners should ask you for a Site ID. OpenAM generates a SAML 1.x Site ID value at configuration time. This Site ID value corresponds to the server. To find this in OpenAM Console, see Federation > SAML 1.x Configuration > Local Site Properties > Site Identifiers, and then click your server URL.

If you have multiple servers in an OpenAM Site set up behind a load balancer, you can generate a Site ID, and then use it for all the servers in your site.

- Generate a Site ID for your site, using the primary site URL.

This example is for an asserting party where the site load balancer host is ap.example.net. The command is bundled with OpenAM server, shown with lines folded to fit on the printed page.

```
$ cd /path/to/tomcat/webapps/openam/WEB-INF/lib/
$ java \
  -cp forgerock-util-1.3.5.jar:openam-shared-12.0.0.jar:\
    openam-federation-library-12.0.0.jar com.sun.identity.saml.common.SAMLsiteID \
```

`https://ap.example.net/openam  
9BAg4UmVS6IbjccsSj9gAFYGO9Y=`

## 14.3 Preparing To Secure SAML 1.x Communications

SAML communications are secured using Public Key Infrastructure (PKI). Communications should be protected over the network by HTTPS, and relying parties requesting assertions should use SSL or TLS mutual authentication to identify each other, meaning they should be able to trust each others' certificates. Furthermore, when an asserting party works through the user's browser to post an assertion to the relying party, then the asserting party must digitally sign the SAML response.

A certificate can be trusted when the signer's certificate is trusted, or when the certificate itself is trusted. Trusted partners must either use public key certificates signed by a well-known Certificate Authority (CA), or share their self-signed or private CA signing certificates.

### Procedure 14.2. To Configure Keys For Protecting SAML 1.x Communications

1. See the chapter *Managing Certificates* for instructions on handling your own key pairs.  
For specific instructions on changing signing keys, see the procedure *To Change the Signing Key for Federation*.
2. If necessary, share signing certificates with trusted partners.
3. Import public key certificates shared by trusted partners into your OpenAM key store.

## 14.4 Configuring SAML 1.x For Your Site

After you have gathered configuration information and prepared to secure SAML 1.x communications you can configure SAML 1.x for your site.

### Tip

When you enter SAML 1.x configuration data, OpenAM Console escapes these special characters by default: & < > " ' / . If instead you have already escaped these characters in the data that you plan to enter in OpenAM Console, then set the advanced configuration property com.sun.identity.saml.escapeattributevalue to false under Configuration > Servers and Sites > Default Server Settings > Advanced, and then restart

OpenAM or the container in which it runs to prevent OpenAM Console from escaping the characters for you.

- [Procedure 14.3, “To Configure Asserting Party Local Site Properties”](#)
- [Procedure 14.4, “To Configure Relying Party Local Site Properties”](#)

### **Procedure 14.3. To Configure Asserting Party Local Site Properties**

Using the configuration information you have gathered complete the following steps.

1. Login to OpenAM Console as administrator, amadmin, browse to Federation > SAML 1.x Configuration, and then click Local Site Properties.
2. If the target specifier query string parameter is something other than the standard default TARGET, set it in the Target Specifier field.
3. If instead of the default server Site Identifier, you use a Site Identifier for the OpenAM Site, click New in the Site Identifiers table, and then add the information for the OpenAM Site, including the Site ID that you generated.
4. Target URLs let you configure URLs for which HTTP POST is always used.

When the TARGET specified matches a URL in the Target URLs list, then the asserting party sends the response to the relying party by HTTP POST of an auto-submitting form returned to the browser.

5. If necessary, set the Default Protocol Version.
6. In the Assertion section, change the values if necessary.

Remove Assertion: Yes means that assertions are deleted from memory after they are used, rather than deleted only when they expire.

7. In the Artifact section, change the values if necessary.
8. In the Signing section, for an asserting party using the HTTP POST profile, check at least Sign SAML Assertion.

By default OpenAM signs messages using the certificate with alias test.

Check other options as required by your trusted partners.

9. In the Attribute Query section, if relying parties issue attribute queries, then set the default list of profile attributes to return.
10. In the NameID Format section, map SAML NameID formats to local OpenAM user profile attributes.

This allows OpenAM to map a remote user to a local user profile.

11. In the Attribute Map section, if the parties exchange attributes, then map the SAML attributes requested by relying parties to local OpenAM user profile attributes.
12. Save your work.

#### **Procedure 14.4. To Configure Relying Party Local Site Properties**

Using the configuration information you have gathered complete the following steps.

1. Login to OpenAM Console as administrator, amadmin, browse to Federation > SAML 1.x Configuration, and then click Local Site Properties.
2. If the target specifier query string parameter is something other than the standard default TARGET, set it in the Target Specifier field.
3. If instead of the default server Site Identifier, you use a Site Identifier for the OpenAM Site, click New in the Site Identifiers table, and then add the information for the OpenAM Site, including the Site ID that you generated.
4. Ignore the Target URLs table for a relying party.
5. If necessary, set the Default Protocol Version.
6. In the Assertion section, change the values if necessary.
7. In the Artifact section, change the values if necessary.
8. Ignore the Signing section for relying parties, unless trusted partners require that your site signs SAML requests.

By default OpenAM signs messages using the certificate with alias test.

9. Ignore the Attribute Query section for relying parties.
10. In the NameID Format section, map SAML NameID formats to local OpenAM user profile attributes.

This allows OpenAM to map a remote user to a local user profile when not all the partners are using OpenAM user IDs.

11. In the Attribute Map section, if the parties exchange attributes, then map the SAML attributes requested by relying parties to local OpenAM user profile attributes.

12. Save your work.

## 14.5 Configuring SAML 1.x Trusted Partners

After you have gathered configuration information and if necessary imported public key certificates from trusted partners you can configure SAML 1.x information for the partners.

- [Procedure 14.5, “To Configure a Trusted Relying Party”](#)
- [Procedure 14.6, “To Configure a Trusted Asserting Party”](#)

### Procedure 14.5. To Configure a Trusted Relying Party

OpenAM Console refers to the relying party as the Destination, because the relying party's site is the destination site.

1. Login to OpenAM Console as administrator, amadmin, browse to Federation > SAML 1.x Configuration, and then click New in the Trusted Partners table.
2. Under Destination, select the SAML profiles used with the relying party.
3. In the Common Settings section, set at least a name for the partner configuration, enter the partner's Site ID as the Source ID, and specify the fully qualified domain, optionally with the port number, of the relying party in the Target field. The value in the target field is matched to TARGET parameter values, so it should correspond to the real domain (and optionally port number) in the URLs of resources to access at the relying party's site.

Optionally set a custom site attribute mapper, a custom name identifier mapper, and the SAML Version to use with the partner.

You must also set one or more values in the host list for the partner to identify all hosts from the partner site that can send requests. OpenAM rejects SAML requests from hosts not specified in this list.

4. In the Destination section, if the SAML Artifact profile is used with the relying party, set the SAML URL to the relying party's endpoint that receives the artifact and contacts your asserting party.

If the SAML POST profile is used with the relying party, set the Post URL to the relying party's endpoint that consumes the assertion in the HTTP POST form data and redirects the user's browser to the target at the relying party's site.

If the relying party makes SAML SOAP query requests, optionally set custom attribute or action mappers.

If the relying party signs requests, then either requests include the certificate for the signing key in the KeyInfo element, or OpenAM must find the signing certificate elsewhere. If the relying party provides the signing certificate separately, import the signing certificate into OpenAM's `keystore.jks` file, and set the alias for the signing certificate here in the configuration.

Set the issuer to a host:port combination corresponding to the relying party server issuing the requests.

5. Save your work.

#### **Procedure 14.6. To Configure a Trusted Asserting Party**

OpenAM Console refers to the asserting party as the Source, because the asserting party's site is the source site.

1. Login to OpenAM Console as administrator, amadmin, browse to Federation > SAML 1.x Configuration, and then click New in the Trusted Partners table.
2. Under Source, select the SAML profiles used with the asserting party.
3. In the Common Settings section, set at least a name for the partner configuration and enter the partner's Site ID as the Source ID.

Optionally set a custom account mapper. By default OpenAM maps accounts based on the NameID format configuration for your site.

If the asserting party signs assertions (or other messages) and you have imported the signing certificate into OpenAM's key store (also used as a trust store), then enter the signing certificate alias. If instead the asserting party includes the signing certificate in the KeyInfo element of signed messages, then you can leave the alias blank.

4. In the Source section, if the SAML Artifact profile is used with the asserting party, set the SOAP URL to the asserting party endpoint that responds to requests such as `https://ap.example.net/openam/SAMLSOAPReceiver`.

If the asserting party requires authentication to the SOAP URL, then configure the settings appropriately.

If the SOAP URL is accessed over HTTP, choose None or Basic. If the SOAP URL is accessed over HTTPS, choose SSL/TLS or SSL/TLS with Basic.

Basic means HTTP Basic authentication (with username and password). For HTTP Basic authentication, the authentication at this level is performed by the application server container, not OpenAM. Therefore if the asserting party runs OpenAM and wants to enforce HTTP Basic authentication, the

asserting party administrator must set up the container to handle HTTP Basic authentication for the SOAP URL.

Set the SAML Version as necessary.

If the SAML POST profile is used with the asserting party, set the Issuer to the issuer name, such as a host:port combination.

5. Save your work.

## 14.6 Testing SAML 1.x Web SSO

You can try SAML 1.x Web SSO using OpenAM by following the procedures in this section.

- [Procedure 14.7, “To Prepare the OpenAM Servers”](#)
- [Procedure 14.8, “To Prepare to Test the Asserting Party”](#)
- [Procedure 14.9, “To Prepare to Test the Relying Party”](#)
- [Procedure 14.10, “To Try SAML 1.x Web SSO”](#)

### Procedure 14.7. To Prepare the OpenAM Servers

1. Install two separate OpenAM servers, one to act as asserting party, the other to act as relying party.

How you do this in practice is up to you.

You can for example set up two separate OpenAM servers on a single host by adding aliases for the hosts in your hosts file, and by using separate containers that listen on different ports.

For example if your host is a laptop, you can add the aliases to the loopback address as in the following example line from an /etc/hosts file.

```
127.0.0.1 localhost ap.example.net rp.example.com
```

Then run one application server to listen on port 8080, and another to listen on port 9080.

Deploy and configure OpenAM server with the default configuration at <http://ap.example.net:8080/ap> for the asserting party and at <http://rp.example.com:9080/rp> for the relying party. This allows you to use the default configuration for both servers.

See the [Installation Guide](#) for instructions.

The procedures in this section use those example URLs to represent the OpenAM servers acting as asserting and relying parties.

2. On the asserting party server, login to OpenAM Console as administrator, browse to Federation > SAML 1.x Configuration, and then click Local Site Properties.

Click the server's instance ID in the Site Identifiers table.

Record the asserting party Site ID for later use.

3. On the relying party server, login to OpenAM Console as administrator, browse to Federation > SAML 1.x Configuration, and then click Local Site Properties.

Click the server's instance ID in the Site Identifiers table.

Record the asserting party Site ID for later use.

### **Procedure 14.8. To Prepare to Test the Asserting Party**

Follow these steps to configure the asserting party OpenAM server.

1. Login to OpenAM Console as administrator, browse to Federation > SAML 1.x Configuration, and then click Local Site Properties.
2. On the Local Site Properties page for the asserting party server, select Sign SAML Response.

The asserting party thus signs SAML responses with the private key for the default test certificate.
3. Save your work, and then click Back to Federation.
4. Click New in the Trusted Partners table to add the relying party as a trusted partner.
5. In the Destination area of the Select trusted partner type and profile page, select Artifact and Post (not SOAP Query), and then click Next.
6. Apply the following settings, adjusted for the host names you use.

If a field is not mentioned, accept the defaults.

Under Common Settings, use these settings:

Name: rp.example.com:9080  
Source ID: relying party Site ID that you recorded  
Target: rp.example.com:9080

Under Destination > Artifact, use these settings:

SOAP URL: http://rp.example.com:9080/rp/SAMLAwareServlet  
Host List: rp.example.com

Under Source > Post, set Post URL: http://rp.example.com:9080/rp/  
SAMLPOSTProfileServlet

7. Click Finish to save your work.

#### **Procedure 14.9. To Prepare to Test the Relying Party**

Follow these steps to configure the relying party OpenAM server.

1. Login to OpenAM Console as administrator, browse to Federation > SAML 1.x Configuration, and then click New in the Trusted Partners table to add the asserting party as a trusted partner.
2. In the Source area of the Select trusted partner type and profile page, select Artifact and Post, and then click Next.
3. Apply the following settings, adjusted for the host names you use.

If a field is not mentioned, accept the defaults.

Under Common Settings, use these settings:

Name: ap.example.net:8080  
Source ID: asserting party Site ID that you recorded  
Signing Certificate Alias: test

Under Source > Artifact, set SOAP URL: http://ap.example.net:8080/ap/  
SAMLSOAPReceiver

Under Source > Post, set Issuer: ap.example.net:8080

Click Finish to save your work.

#### **Procedure 14.10. To Try SAML 1.x Web SSO**

Once you have successfully configured both parties, try SAML 1.x Web SSO.

1. Log out of OpenAM Console on both servers.

2. Try Web SSO using the SAML Artifact profile.
  - a. Simulate the OpenAM administrator browsing the asserting party's site, and selecting a link to the OpenAM Console on the relying party's site.

The URL to simulate this action is something like `http://ap.example.net:8080/ap/SAMLAwareServlet?TARGET=http://rp.example.com:9080/rp`.

OpenAM requires that you authenticate.
  - b. Login as OpenAM demo user, `demo` with default password `changeit`, on the asserting party server.
  - c. Notice that you are redirected to OpenAM Console on the relying party server, and that you are successfully logged in as the `demo` user.
  - d. Log out of OpenAM Console on both servers.
3. Try Web SSO using the SAML HTTP POST profile.
  - a. Simulate the OpenAM administrator browsing the asserting party's site, and selecting a link to the OpenAM Console on the relying party's site.

The URL to simulate this action is something like `http://ap.example.net:8080/ap/SAMLPOSTProfileServlet?TARGET=http://rp.example.com:9080/rp`.

OpenAM requires that you authenticate.
  - b. Login as OpenAM administrator, `amadmin`, on the asserting party server.
  - c. Notice that you are redirected to OpenAM Console on the relying party server, and that you are successfully logged in as `amadmin`.



---

## Chapter 15

# The RESTful Security Token Service

This chapter specifies the configuration requirements associated with the RESTful Security Token Service (REST-STS). Like previous implementations of STS, this implementation secures identity tokens and messages.

The REST-STS is inspired by the WS-Trust STS. The REST-STS supports one basic operation: token transformation. This chapter describes how you can configure REST-STS token transformations through the OpenAM UI.

Different Identity Relationship Management (IRM) services require different token types. Given the variety of token types in use today, it can be helpful to have a service that you can configure to transform tokens.

## 15.1 About the REST-STS

### Note

Do not use the WSP and WSC agent UI tools described in [Configuring Policy Agent Profiles](#).

The OpenAM implementation of REST-STS supports a variety of token transformations, as well as different attributes from SAML2 assertions, along with different encryption contexts.

The REST-STS UI currently includes transformations from three types of tokens:

- UNT: Username Token. May be associated with usernames and/or associated passwords.
- OpenID Connect: OpenID Connect token.
- OpenAM: OpenAM session token.

At this time, the ForgeRock REST-STS supports conversions from these types of tokens to a SAML2 token. For the conversion, you can configure OpenAM to invalidate the interim OpenAM session.

## 15.2 Configuring the Security Token Service

To access the REST-STS configuration menu for your realm, navigate to Access Control > *Realm Name* > STS. When you see the *Realm Name* - REST STS Instances menu, click Add. You should see an excerpt of the REST Security Token Service menu in the following screen shot.

The screenshot shows the 'update rest sts instance' configuration page. At the top, there are two tabs: 'General Configuration' and 'Deployment Configuration'. Below these tabs, there are two dropdown menus: 'Keystore Configuration' and 'Issued SAML2 Token Configuration'. On the right side of the page, there are 'Save' and 'Back' buttons. The main section is titled 'General Configuration'. It contains a field 'The issuer name:' with the value 'ForgeRock'. Below this, there is a table with three columns: 'Supported Token Transforms:', 'X->Y', and 'Description'. The 'Supported Token Transforms:' column lists several options: 'UNT->SAML2;invalidate interim OpenAM session', 'UNT->SAML2;don't invalidate interim OpenAM session', 'OPENIDCONNECT->SAML2;invalidate interim OpenAM session', and 'OPENIDCONNECT->SAML2;don't invalidate interim OpenAM session'. The 'X->Y' column shows 'X' as 'Unt' and 'Y' as 'Saml2'. The 'Description' column provides a detailed explanation of the transforms. At the bottom of the page, there is a link 'Back to top'.

### Deployment Configuration

Deployment Url Element: sts-test  
STS endpoint Url will be composed of rest-sts/realm/uriElement

The REST STS configuration menu is subdivided into four sections:

### 15.2.1 REST STS General Configuration

At the top of the REST STS configuration menu, you can set up the General Configuration for the system.

You will first add a name for the REST-STS token issuer. You can pick any name; the REST-STS includes that name with the **issuer** label for each token.

You can then select from one of the noted types of token transformations. Be aware, you should not select both "invalidate" and "don't invalidate" options for the same token transformation.

### 15.2.2 REST STS Deployment Configuration

In the Deployment Configuration section, you can assign a specific endpoint for the REST STS, and set up the mapping for each token transformation.

OpenAM includes what you enter in the **Deployment Url Element** text box in the REST-STS endpoint, as well as the name of the REST-STS instance. As an example, assume that you've set the **Deployment Url Element** to `instance1`, on a realm named `NewRealm`.

In that case, the STS Endpoint URI would be `rest-sts/NewRealm/instance1`, and the REST STS Instance would be `NewRealm/instance1`.

Under Authentication Target Mappings, you can configure two or three pieces of information related to each input token. The default options shown in this field are known as a tuple, an ordered list of elements. Each element is separated with a pipe character: "|".

Authentication target type

Whether the target is a module or service

Name of the authentication module or service

Two default options are included: the `ldapService` service for usernames, and `OPENIDCONNECT` module to identify elements of the OpenAM RESTful authentication context that should be consumed to validate that OpenID Connect token.

The `OPENIDCONNECT` module supports the transformation of a token from OpenID Connect to SAML2.

Authentication context information (optional)

Defines STS authentication context information about the token to be consumed. In the case shown in the default, the name of the token to be consumed is `oidc_id_token`.

### 15.2.3 REST STS Keystore Configuration

The REST STS can use the same keystore as is used for OpenAM. While the version of Java that you use includes its own version of the **keytool** command, the location where you set up the keystore file may vary.

For example, you may use the procedure [To Set Up OpenAM With HTTPS on Tomcat](#), if you use Tomcat as the container for OpenAM.

The following describes the purpose of each entry under Keystore Configuration:

#### Keystore Path

You can set this to the location of the keystore file of your choice. For reference, the default OpenAM keystore file is `keystore.jks`, and you can find it in the OpenAM configuration directory, as described in the following installation procedure: [To Deploy OpenAM](#)

#### Keystore Password

Enter the password for the keystore file, normally `keystore.jks`. The default password for that file packaged with OpenAM is `changeit`.

#### Encryption Key Alias

Alias used for the encryption key; you may use that alias with the **keytool** command.

#### Encryption Key Password

Enter the password for the encryption key.

#### Signature Key Alias

The REST-STS uses a signature key to verify X.509 tokens. Enter the desired alias for that signature key.

#### Signature Key Password

Enter the password for the signature key.

### 15.2.4 REST STS SAML2 Token Configuration

This REST-STS configuration UI supports token transformations from username, Open ID Connect, and OpenAM tokens to SAML2 format. Configure the options shown in this part of the REST-STS UI menu to customize how OpenAM will create those SAML2 tokens.

Several options require you to include the specific classes, either in a custom .jar file or classes configured in files in the WEB-INF/classes directory.

#### NameIdFormat

Specify the name identifier format for the SAML2 token that you want REST-STS to generate.

**Token Lifetime (Seconds)**

Enter the desired lifetime for the SAML2 token that you want REST-STS to generate. The default is 10 minutes.

**Custom Conditions Provider Class Name (optional)**

If you need to customize the conditions for the issued SAML2 token, specify the class name of the implementation in the associated text box.

Implement the specified class name with the `org.forgerock.openam.sts.tokengeneration.saml2.statements.ConditionsProvider` interface.

**Custom Subject Provider Class Name (optional)**

If you need to define a custom provider for the SAML2 token, enter the class name of the provider here.

Implement the specified class name with the `org.forgerock.openam.sts.tokengeneration.saml2.statements.SubjectProvider` interface.

**Custom Authentication Statements Class Name (optional)**

If you need to create a custom authentication statement for the SAML2 token, enter the class name of the statement here.

Implement the specified class name with the `org.forgerock.openam.sts.tokengeneration.saml2.statements.AuthenticationStatementsProvider` interface.

**Custom Attribute Statements Class Name (optional)**

If you need to define a custom attribute statement for the SAML2 token, enter the class name of the statement here.

Implement the specified class name with the `org.forgerock.openam.sts.tokengeneration.saml2.statements.AttributeStatementsProvider` interface.

**Custom Authorization Decision Statements Class Name (optional)**

If you need to set up a custom authorization decision statement for the SAML2 token, enter the class name of the statement here.

Implement the specified class name with the `org.forgerock.openam.sts.tokengeneration.saml2.statements.AuthzDecisionStatementsProvider` interface.

**Custom Attribute Mapper Class Name (optional)**

If you need to define a custom attribute mapper for the SAML2 token, enter the class name of the mapper here.

Implement the specified class name with the `org.forgerock.openam.sts.tokengeneration.saml2.statements.AttributeMapper` interface.

#### Custom Authentication Context Class Name (optional)

If you need to note a custom AuthContext mapper for the SAML2 token, enter the class name of the context here.

If you need to include an AuthnContext mapping, implemented by the `org.forgerock.openam.sts.token.provider.AuthnContextMapperImpl` class, enter the class name of the following interface: `org.forgerock.openam.sts.tokengeneration.saml2.statements.AuthnContextMapper`.

#### Canonicalization Algorithm

For more information on the two options shown see the W3C Recommendation document entitled: [Exclusive XML Canonicalization](#).

#### Sign Assertion

If checked, this implementation of the REST-STS will sign the assertion of the associated token.

#### Assertion audiences

If you need to restrict the audience for the newly created SAML2 tokens, enter appropriate assertion audiences here. Values so entered will be used to populate the Audience based on the AudienceRestriction element, as defined in section 4.1.4.2 of the OASIS document entitled: [Profiles for the OASIS Security Assertion Markup Language \(SAML\) v2.0](#).

#### Attribute Mappings

You should configure mappings between SAML2 attribute names (Map keys) and local OpenAM attributes (Map values).

The DefaultAttributeMapper looks at profile attributes in configured data stores, or in session properties.

The map keys will define the names of the attributes included in Assertion Attribute statements.

The data pulled from the subject's directory entry or session state corresponding to the map value will define the value corresponding to the name of the attribute.

Map keys can be set up in the following format: [NameFormatURI] |  
SAML\_ATTRIBUTE\_NAME.

If the attribute value is enclosed in quotes, that quoted value will be included in the attribute without mapping. Binary attributes should be followed by '`;binary`'

Beyond that, examples for appropriate mappings include:

```
EmailAddress=mail
```

## REST STS SAML2 Token Configuration

---

Address=postaladdress

urn:oasis:names:tc:SAML:2.0:attrname-format:uri|urn:mace:dir:attribute-def:cn=cn

partnerID="staticPartnerIDValue"

urn:oasis:names:tc:SAML:2.0:attrname-format:uri|nameID="staticNameIDValue"

photo=photo;binary

urn:oasis:names:tc:SAML:2.0:attrname-format:uri|photo=photo;binary



---

## Chapter 16

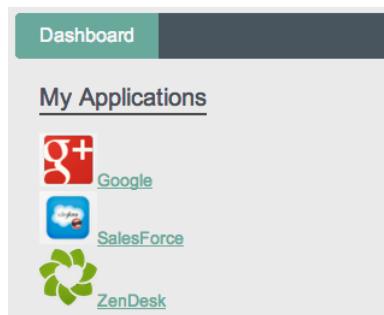
# Configuring the Dashboard Service

This chapter shows how to configure the OpenAM Dashboard service.

### 16.1

### About 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.



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 can be enabled by logging in to OpenAM Console as administrator and making sure that XUI is enabled under Configuration > Authentication > Core > XUI Interface > Enabled.

Once the Dashboard service is configured for a user, the user can access their Dashboard after logging in through the XUI under /XUI/#dashboard/. For example, the full URL depending on the deployment might be at <https://openam.example.com:8443/openam/XUI/#dashboard/>.

## 16.2 Setting Up 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.

### Procedure 16.1. To Add Applications from the Dashboard

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. Login to the OpenAM console as OpenAM Administrator, amadmin.
2. On the Configuration tab > Global > Dashboard 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 2.0 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.

## 16.3 Configuring Dashboard Service for a Realm

### Procedure 16.2. 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 Configuration > Global > Dashboard.

1. On the Access Control > *Realm Name* > Services, click Add....
2. Select the Dashboard service, then click Next.
3. Add or remove the applications you would like to appear on the Dashboard service for the realm.
4. Click Finish when you are done.

## 16.4 Adding Applications to a User's Dashboard

### Procedure 16.3. To Add an Application to a User's Dashboard

Use the following steps to add an application to a user's Dashboard.

1. On the Access Control > *Realm Name* > Subjects, 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 16.4. Removing User Access to an Application**

You may need to remove an application from user's Dashboard, but you don't want to entirely delete the user. The following steps walk you through removing an application from a user's Dashboard.

1. On the Access Control > *Realm Name* > Subjects, 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.

---

## **Chapter 17**

# Configuring REST APIs

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 17.1, “Configure Versioning Behavior by using the Web-based Console”](#)
- [Procedure 17.2, “Configure Versioning Behavior by using 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.

---

### *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 17.1. Configure Versioning Behavior by using the Web-based Console**

1. Login as OpenAM administrator, amadmin.
2. Click Configuration > Global > 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'.

The screenshot shows the 'REST APIs' configuration page under 'Global Attributes'. It has three radio buttons for 'Default Version': 'Latest' (selected), 'Oldest', and 'None'. A tooltip for 'Default Version' explains it as 'The API version to use when the REST request does not specify a desired version'. Below that is a 'Warning Header' section with a checked checkbox labeled 'Enabled', which is described as 'Whether to include a warning header in the response to a request which fails to include the Accept-API-Version header'. At the bottom are 'Save', 'Reset', and 'Back to Service Configuration' buttons.

4. Optionally, enable 'Warning Header' to include warning messages in the headers of responses to requests.
5. Save your work.

## **Procedure 17.2. Configure Versioning Behavior by using 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.

---



---

## **Chapter 18**

# Backing Up and Restoring OpenAM 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 chapter 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 & Restoring Data\*](#).

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.

This chapter aims to cover the following uses of backup data.

1. Recovery from server failure.
  - [Procedure 18.1, “To Back Up All Server Configuration Data”](#)
  - [Procedure 18.2, “To Restore All Server Configuration Data”](#)
2. Recovery from serious administrative error.
  - [Procedure 18.3, “To Export Only Configuration Data”](#)
  - [Procedure 18.4, “To Restore Configuration Data After Serious Error”](#)

### **Procedure 18.1. 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.

Use this procedure when the following statements are true.

- OpenAM stores configuration data in the embedded OpenDJ directory server.

The embedded OpenDJ directory server files are co-located with other OpenAM configuration files.

- 
- If OpenAM 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.

For deployments with long-lived sessions, you can limit the extent of re-authentication by using session failover. For details, see the *Installation Guide* chapter, [Setting Up OpenAM Session Failover](#).

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 & Restoring Data](#).

## Important

Understand the explanation in the introductory paragraphs of this chapter before you proceed. Directory backup validity depends on replication purge delay, which by default for OpenDJ directory server is 3 days.

Also do not restore configuration data from a backup of a different release of OpenAM. The structure of configuration data can change from release to release.

Follow these steps for each OpenAM server that you want to back up.

1. Stop OpenAM or the container in which it runs.
2. Backup 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 +i5F-%m-%S`.zip \
  openam .openamcfg/AMConfig_path_to_tomcat_webapps_openam_ \
  --exclude openam/openam/debug/* openam/openam/log/* openam/openam/stats* \
  openam/opensds/logs/* openam/opensds/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 18.2. To Restore All Server Configuration Data**

This procedure restores all the configuration data for a server, where the configuration data has been backed up as described in [Procedure 18.1, “To Back Up All Server Configuration Data”](#). This procedure applies when rebuilding a failed server.

Use this procedure when the following statements are true.

- OpenAM stores configuration data in the embedded OpenDJ directory server.

The embedded OpenDJ directory server files are co-located with other OpenAM configuration files.

- If OpenAM 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.

For deployments with long-lived sessions, you can limit the extent of re-authentication by using session failover. For details, see the *Installation Guide* chapter, [Setting Up OpenAM Session Failover](#).

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 & Restoring Data](#).

### **Important**

Understand the explanation in the introductory paragraphs of this chapter before you proceed. Directory backup validity depends on replication purge delay, which by default for OpenDJ directory server is 3 days.

Also do not restore configuration data from a backup of a different release of OpenAM. The structure of configuration data can change from release to release.

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 18.3. 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 18.4. 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 OpenAM 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 18.3, “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 18.3, “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 *OpenDJ Administration Guide* section on [Updating the Directory](#).

---

## Chapter 19

# Managing Certificates

This chapter shows you how to handle certificates used to protect network communication and for authentication.

In theory, you should not have to concern yourself with certificates when working with OpenAM. OpenAM core services and Java EE policy agents depend on the certificates installed for use with the web application container in which they run. OpenAM web policy agents depend on the certificates installed for use with the web server. Theoretically, each certificate has been signed by a well-known Certificate Authority (CA), whose certificate is already installed in the Java CA certificates trust store (`$JAVA_HOME/jre/lib/security/cacerts`, default password `changeit`) and in browsers, and so is recognized by other software used without you having to configure anything.

In practice, you might not have the budget for CA signed certificates in your lab or test environment, where you might constantly be installing new configurations, using and throwing away certificates for experiments and repeated tests. In the lab, therefore, you set up OpenAM to use self-signed certificates that you generate at no cost, but that are not recognized, and therefore not trusted out of the box.

How you configure the containers where OpenAM and your applications run to use self-signed certificates depends on your web application server or web server software. Yet, the basic principles apply.

- First, your container requires its own certificate for setting up secure connections.

- Second, the clients connecting must be able to trust the container certificate. Generally this means that clients must recognize the container certificate because they have a copy of the public certificate stored somewhere the client trusts.
- Third, if you use certificate authentication in OpenAM, OpenAM must also be able to find a copy of the client's public certificate to trust the client, most likely by finding a match with the certificate stored in the client profile from the identity repository. How you include client certificates in their identity repository entries depends on your identity repository more than it depends on OpenAM.

Some client applications let you trust certificates blindly. This can be helpful when working in your lab or test environment with self-signed certificates. For example, you might want to use HTTPS with the OpenAM RESTful API without having the client recognize the self-signed server certificate.

```
$ curl \
"https://openam.example.com:8443/openam/identity/authenticate
?username=bjensen&password=hifalutin"
curl: (60) Peer certificate cannot be authenticated with known CA certificates
More details here: http://curl.haxx.se/docs/sslcerts.html

curl performs SSL certificate verification by default, using a "bundle"
of Certificate Authority (CA) public keys (CA certs). If the default
bundle file isn't adequate, you can specify an alternate file
using the --cacert option.
If this HTTPS server uses a certificate signed by a CA represented in
the bundle, the certificate verification probably failed due to a
problem with the certificate (it might be expired, or the name might
not match the domain name in the URL).
If you'd like to turn off curl's verification of the certificate, use
the -k (or --insecure) option.
$ curl \
--insecure \
"https://openam.example.com:8443/openam/identity/authenticate
?username=bjensen&password=hifalutin"
token.id=AQIC5wM2LY4SfczMax8jegpSiaigB96N0WyllLilsd0PUMjY.*AAJTSQACMDE.*
```

### Procedure 19.1. To Set Up OpenAM With HTTPS on Tomcat

The container where you install OpenAM requires a certificate in order to set up secure connections. The following steps demonstrate one way to set up Tomcat with an HTTPS connector, using the Java **keytool** command to manage the certificate and key stores. Once Tomcat can do HTTPS, you deploy OpenAM as you normally would, over HTTPS.

1. Stop Tomcat.
2. Create a certificate and store it in a new key store.

```

$ cd /path/to/tomcat/conf/
$ keytool -genkey -alias openam.example.com -keyalg RSA -keystore keystore.jks
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 <openam.example.com>
(RTURN if same as keystore password):

```

3. Uncomment the SSL connector configuration in Tomcat's `conf/server.xml`, specifying your key store file and password.

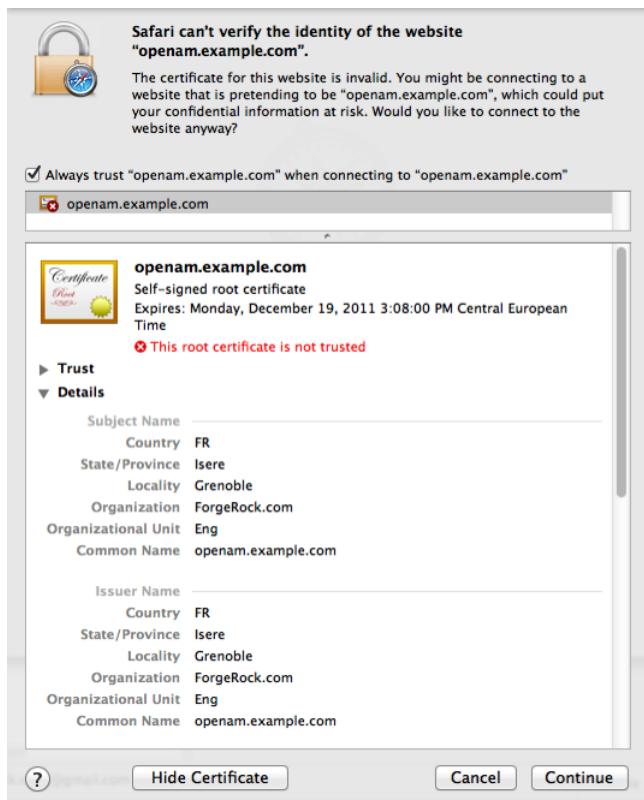
```

<!-- Define a SSL HTTP/1.1 Connector on port 8443
   This connector uses the JSSE configuration, when using APR, the
   connector should be using the OpenSSL style configuration
   described in the APR documentation -->
<!--
-->
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
           maxThreads="150" scheme="https" secure="true"
           keystoreFile="/path/to/tomcat/conf/keystore.jks"
           keystorePass="changeit"
           clientAuth="false" sslProtocol="TLS" />

```

4. Start Tomcat.
5. Verify that you can connect to Tomcat on port 8443 over HTTPS.

Your browser does not trust the certificate, because the certificate is self-signed, not signed by any of the CAs your browser knows.



You recognize the Subject and Issuer of your certificate, and so can choose to trust the certificate, effectively saving it into your browser's trust store.

6. Deploy and configure OpenAM as you normally would.

### Procedure 19.2. To Share Self-Signed Certificates

When you use a self-signed certificate for your container, clients connecting must be able to trust the container certificate. Your browser makes this an easy, but manual process. For other client applications, you must import the certificate into the trust store used by the client. By default, Java applications can use the \$JAVA\_HOME/jre/lib/security/cacerts store. The default password is changeit.<sup>1</sup> The steps that follow demonstrate how to import a self-signed certificate into the Java cacerts store.

<sup>1</sup>Alternatively, you can specify the trust store for a Java application, such as -Djavax.net.ssl.trustStore=/path/to/truststore.jks -Djavax.net.ssl.trustStorePassword=changeit.

- 
1. Export the certificate from the key store.

```
$ cd /path/to/tomcat/conf/
$ keytool \
  -exportcert \
  -alias openam.example.com \
  -file openam.crt \
  -keystore keystore.jks
Enter keystore password:
Certificate stored in file <openam.crt>
```

2. Import the certificate into the trust store.

```
$ keytool \
  -importcert \
  -alias openam.example.com \
  -file openam.crt \
  -trustcacerts \
  -keystore $JAVA_HOME/jre/lib/security/cacerts
Enter keystore password:
Owner: CN=openam.example.com, OU=Eng, O=ForgeRock.com, L=Grenoble, ST=Isere,
C=FR
Issuer: CN=openam.example.com, OU=Eng, O=ForgeRock.com, L=Grenoble, ST=Isere,
C=FR
Serial number: 4e789e40
Valid from: Tue Sep 20 16:08:00 CEST 2011 until: Mon Dec 19 15:08:00 CET 2011
Certificate fingerprints:
  MD5: 31:08:11:3B:15:75:87:C2:12:08:E9:66:00:81:61:8D
  SHA1: AA:90:2F:42:0A:F4:A9:A5:0C:90:A9:FC:69:FD:64:65:D9:78:BA:1D
  Signature algorithm name: SHA1withRSA
  Version: 3
Trust this certificate? [no]: yes
Certificate was added to keystore
```

### Procedure 19.3. To Change the Signing Key for Federation

Digital signatures are constructed as follows.

- The signer computes a hash of the data to sign, and encrypts the hash using a private key to get the signature.
- The signer then attaches the signature to the data, and exchanges the message with the recipient.
- To validate the digital signature on the message, the recipient decrypts the signature using the public key certificate that corresponds to the private key of the signer.

- 
- The recipient computes the hash of the data, and then checks that the decrypted signature, in other words the decrypted hash, matches the computed hash.
  - Parties signing requests, responses, or assertions must therefore share the public key certificates for signing keys. The certificates can either be shared in advance and imported into the trusted partners' trust stores, then referenced in the configuration by their trust store aliases, or can be shared in each signed message.

The following steps cover how to change the signing key for an identity provider. This procedure involves creating a self-signed certificate in a new key store file, and also preparing encrypted password files so that OpenAM can access the key store and the private key.

1. If you do not already have the new signing key in your key store, generate a new key and key store.

The following example starts an interactive **keytool** session that requests information needed to generate a new key valid for two years, and puts it in a key store named `keystore.jks`. You can perform this step in a temporary location, and then move the files generated once you have completed your work.

Keep track of the passwords you enter here, as you use them in the next step.

```
$ cd /tmp
$ keytool \
  -genkeypair \
  -alias newkey \
  -keyalg RSA \
  -keysize 1024 \
  -validity 730 \
  -storetype JKS \
  -keystore keystore.jks
Enter keystore password:
Re-enter 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):  
Re-enter new password:
```

Self-signed keys are not automatically recognized by other entities. You must also share the self-signed key as described in [Procedure 19.2, “To Share Self-Signed Certificates”](#).

2. Using the passwords you entered in the previous step, prepare the password files to encrypt.

Create two files, each containing only a password in clear text. You can create the files in the same directory as the key store.

- `keypass.cleartext` contains the clear text key password for the private key you generated.
- `storepass.cleartext` contains the clear text key store password.

3. If you have not already done so, install the administration tools as described in [To Set Up Administration Tools](#).
4. Prepare encrypted password files for use by OpenAM.

```
$ ./ampassword --encrypt keypass.cleartext > .keypass  
$ ./ampassword --encrypt storepass.cleartext > .storepass
```

Remove the `*.cleartext` files after preparing the encrypted versions.

5. Replace the default OpenAM key store and password files with the ones that you have created.

The following example works with an installation of OpenAM where the deployment URI is `/openam`.

```
$ cp keystore.jks .keypass .storepass ~/openam/openam/
```

6. Restart OpenAM, or the container where it runs, so that OpenAM can use the new key store and encrypted password files.
7. Login to OpenAM console as administrator, and then set the new signing key in one of two ways.
  - a. If you have not yet configured your identity provider, select Common Tasks > Create Hosted Identity Provider, and then follow the instructions, selecting your key from the Signing Key drop-down list.

- 
- b. If you have already configured your identity provider, browse to Federation > *provider-name* > Assertion Content > Signing and Encryption, and then edit the signing key certificate alias.
- Save your work.
- 8. Share updated metadata with other entities in your circle of trust as described in [\*Setting Up SAML 2.0 SSO\*](#).

---

## Chapter 20

# Monitoring OpenAM Services

This chapter covers how to monitor OpenAM services to ensure appropriate performance and service availability.

## 20.1 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 OpenAM console as OpenAM administrator, and browse to Configuration > System > 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.

### 20.1.1 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 AQICMBCKlw6G3vzK3TYRbtTpNYAagVIPNP
```

Or:

```
$ cat openam/openam/opensso_mon_auth
demo AQICvSe+tX Eg8TUUT8ekzHb8IRzVSvm1Lc2u
```

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.

**MBean View**

[Project OpenDMKopendmk-1.0-b02]

- MBean Name:**  
SUN\_OPENSSO\_SERVER\_MIB/ssoServerServerTable:ssoServerEntry.serverHostName=openam-ter.example.com,ssoServerEntry.serverPort=8080
- MBean Java Class:** com.sun.identity.monitoring.SsoServerServerEntryImpl

Reload Period in seconds:

[Back to Agent View](#) Reload[Unregister](#)**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                      |

**20.1.2 JMX Monitoring**

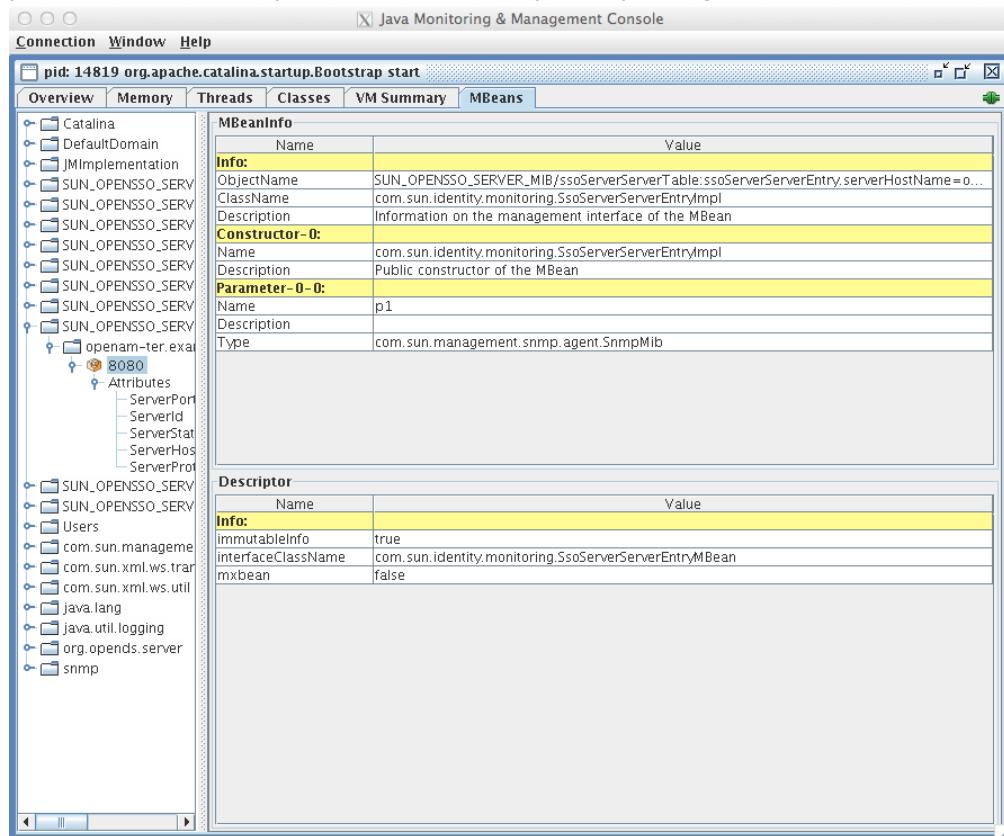
You can configure OpenAM to allow you to listen for Java Management eXtension (JMX) clients, by default on port 9999. Either use the OpenAM console page under Configuration > System > Monitoring and make sure both Monitoring Status and Monitoring RMI interface status are both set to Enabled, or use the **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.



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.

### 20.1.3 SNMP Monitoring

You can configure OpenAM to allow you to listen on port 8085 for SNMP monitoring. 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-snmp-enabled=true
```

## 20.2 Monitoring CTS Tokens

The [OpenAM Core Token Service](#) (CTS) provides persistent and highly available token storage for a several components within OpenAM, including user sessions, OAuth 2.0 and SAML 2.0 tokens.

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 [SNMP Monitoring](#).

### 20.2.1 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 the OpenAM Reference chapter on the [Core Token Service \(CTS\) Object Identifiers \(OIDs\)](#).

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.36733.1.2.3.3.1.1.1
```

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  $2^{64} - 1$ .

## 20.2.2 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 OpenAM Console under Configuration > System > Monitoring. For details, see the system configuration reference section, [Monitoring](#).)

Interface Stability: [Evolving](#)

As described in [Section 20.2.1, “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 20.2.1, “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 20.1. OIDs Used in SNMP Monitoring For Policy Evaluation**

| <b>OID</b>                          | <b>Description</b>   | <b>Syntax</b>   |
|-------------------------------------|--|-----------------|
| enterprises.36733.1.2.2.1.<br>1.1   | Cumulative number of policy evaluations for specific resources (self)      | Counter64       |
| enterprises.36733.1.2.2.1.<br>1.2   | Average rate of policy evaluations for specific resources (self)           | Counter64       |
| enterprises.36733.1.2.2.1.<br>1.3   | Minimum rate of policy evaluations for specific resources (self)           | Counter64       |
| enterprises.36733.1.2.2.1.<br>1.4   | Maximum rate of policy evaluations for specific resources (self)           | Counter64       |
| enterprises.36733.1.2.2.1.<br>2.1   | Cumulative number of policy evaluations for a tree of resources (subtree)  | Counter64       |
| enterprises.36733.1.2.2.1.<br>2.2   | Average rate of policy evaluations for a tree of resources (subtree)       | Counter64       |
| enterprises.36733.1.2.2.1.<br>2.3   | Minimum rate of policy evaluations for a tree of resources (subtree)       | Counter64       |
| enterprises.36733.1.2.2.1.<br>2.4   | Maximum rate of policy evaluations for a tree of resources (subtree)       | Counter64       |
| enterprises.36733.1.2.2.1.<br>2.1.1 | Average length of time to evaluate a policy for a specific resource (self) | Counter64       |
| enterprises.36733.1.2.2.1.<br>2.1.2 | Slowest evaluation time for a specific resource (self)                     | SnmpAdminString |

## SNMP Monitoring For Sessions

---

| OID                             | Description   | Syntax          |
|---------------------------------|---|-----------------|
| enterprises.36733.1.2.2.1.2.2.1 | Average length of time to evaluate a policy for a tree of resources (subtree) | Counter64       |
| enterprises.36733.1.2.2.1.2.2.2 | Slowest evaluation time for a tree of resources (subtree)                     | SnmpAdminString |
| enterprises.36733.1.2.2.1.3.1   | Slowest individual policy evaluation time overall                             | SnmpAdminString |

### 20.2.3 SNMP Monitoring For Sessions

You can monitor session statistics over SNMP. OpenAM records statistics for up to a configurable number of recent sessions. (You can configure the number in OpenAM Console under Configuration > System > Monitoring. For details, see the system configuration reference section, [Monitoring](#).)

Interface Stability: [Evolving](#)

As described in [Section 20.2.1, “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 20.2.1, “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 20.2. OIDs Used in SNMP Monitoring For Sessions**

| OID                         | Description  | Syntax    |
|-----------------------------|--|-----------|
| enterprises.36733.1.2.1.1.1 | Total number of current internal sessions            | Counter64 |
| enterprises.36733.1.2.1.1.2 | Average time it takes to refresh an internal session | Counter64 |
| enterprises.36733.1.2.1.1.3 | Average time it takes to logout an internal session  | Counter64 |
| enterprises.36733.1.2.1.1.4 | Average time it takes to destroy an internal session | Counter64 |

| <b>OID</b>                  | <b>Description</b>   | <b>Syntax</b> |
|-----------------------------|--|---------------|
| enterprises.36733.1.2.1.1.5 | Average time it takes to set a property on an internal session     | Counter64     |
| enterprises.36733.1.2.1.2.1 | Total number of current remote sessions                            | Counter64     |
| enterprises.36733.1.2.1.2.2 | Average time it takes to refresh a remote session                  | Counter64     |
| enterprises.36733.1.2.1.2.3 | Average time it takes to logout a remote session                   | Counter64     |
| enterprises.36733.1.2.1.2.4 | Average time it takes to destroy a remote session                  | Counter64     |
| enterprises.36733.1.2.1.2.5 | Average time it takes to set a property on a remote session        | Counter64     |
| enterprises.36733.1.2.1.3.1 | Total number of sessions currently in the Core Token Service (CTS) | Counter64     |
| enterprises.36733.1.2.1.3.2 | Average time it takes to refresh a CTS session                     | Counter64     |
| enterprises.36733.1.2.1.3.3 | Average time it takes to logout a CTS session                      | Counter64     |
| enterprises.36733.1.2.1.3.4 | Average time it takes to destroy a CTS session                     | Counter64     |
| enterprises.36733.1.2.1.3.5 | Average time it takes to set a property on a CTS session           | Counter64     |

## 20.3 Is OpenAM Running?

You can check over HTTP whether OpenAM is up, using `isAlive.jsp`. Point your application to the file under the OpenAM 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 OpenAM is in operation.

## 20.4 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.

#### **20.4.1 Setting Debug Logging Levels**

To adjust the debug level while OpenAM is running, login to the OpenAM console as OpenAM administrator, and browse to Configuration > Servers and Sites > *Server Name* > General, and then scroll down to 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.

#### **20.4.2 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.

After changing this setting, restart OpenAM or the container in which it runs for the change to take effect.

#### **20.4.3 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. Login to OpenAM 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 console.

3. Select the service to debug and also the level required given the hints provided in the `Debug.jsp` page.

The change 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.

#### 20.4.4 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`

This property 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`

This property 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`

This property specifies an interval in minutes between debug log rotations. Set this to a value greater than zero to enable debug log rotation.

After you edit the `debugconfig.properties` file, you must restart OpenAM or the web container where it runs for the changes to take effect.

### 20.5 Session Management

OpenAM console lets the administrator view and manage current user sessions under the Sessions tab page.

The screenshot shows the OpenAM Session Management interface. At the top, there is a navigation bar with tabs: Common Tasks, Access Control, Federation, Configuration, and Sessions. The Sessions tab is selected. Below the navigation bar, there is a search bar labeled "Sessions" with a "Search" button. A link "Current Sessions" is visible. On the right, there is a "View:" dropdown set to "openam.example.com:8080". The main content area displays a table titled "Sessions (2 Item(s))". The table has columns: User Id, Time Remaining (minutes), Max Session Time (minutes), Time Idle(minutes), and Max Idle Time (minutes). There are two rows in the table:

| User Id | Time Remaining (minutes) | Max Session Time (minutes) | Time Idle(minutes) | Max Idle Time (minutes) |
|---------|--------------------------|----------------------------|--------------------|-------------------------|
| amadmin | 119                      | 120                        | 0                  | 30                      |
| demo    | 119                      | 120                        | 0                  | 30                      |

To end a user session manually, select the user's session, and then click the Invalidate Session button. As a result, the user has to authenticate again.

## Note

Deleting a user does not automatically remove any of the user's sessions. After deleting a user, check for any sessions for the user and remove them under the Console's Sessions tab.

---

## Chapter 21

# Tuning OpenAM

This chapter 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.

As a rule of thumb, an OpenAM server in production with a 3 GB heap can handle 100,000 sessions. Although you might be tempted to use a larger heap with a 64-bit JVM, smaller heaps are easier to manage. Thus, rather than scaling single servers up to increase the total number of simultaneous sessions, consider scaling out by adding more servers instead. The suggestions that follow pertain to production servers.

## 21.1 OpenAM Server Settings

OpenAM has a number of settings that can be tuned to increase performance.

### 21.1.1 General Settings

The following general points apply.

- Set debug level to `error`.
- Disable session failover debugging.
- Set container-level logging to a low level such as `error` or `severe`.

### 21.1.2 LDAP Settings

Tune your LDAP data stores, your LDAP authentication modules, and connection pools for CTS and configuration stores.

#### 21.1.2.1 Tuning LDAP Data Store Settings

To change LDAP data store settings, browse to Access Control > *Realm Name* > Data Stores > *Data Store Name* in the OpenAM console. Each data store has its own connection pool and therefore each data store needs its own tuning.

**Table 21.1. 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.<br><br>(sun-idrepo-ldapv3-config-connection_pool_min_size)  |
| 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.<br><br>(sun-idrepo-ldapv3-config-connection_pool_max_size) |

### 21.1.2.2 Tuning LDAP Authentication Module Settings

To change connection pool settings for the LDAP authentication module, browse to Configuration > Authentication > Core in the OpenAM console.

**Table 21.2. 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.<br><br>(iplanet-am-auth-ldap-connection-pool-default-size) |

### 21.1.2.3 Tuning LDAP CTS & 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 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 10, 1 connection is allocated for cleanup of expired CTS tokens, 4 connections are allocated for other CTS operations, and 5 connections are allocated for services accessing the configuration. If the Maximum Connection Pool size is 20, 1 connection is allocated for cleanup of expired CTS tokens, 8 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

---

<sup>1</sup> 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.

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 pool sizes under Configuration > Servers and Sites > *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 pool size under Configuration > Servers and Sites > *server name* > CTS > External Store Configuration.

In both cases, if you must change the default connection timeouts, set the advanced properties described below under Configuration > Servers and Sites > *server name* > Advanced.

**Table 21.3. CTS Store LDAP Connection Pool Settings**

| Property                | Default Value | Suggestions   |
|-------------------------|---------------|---|
| Maximum Connection Pool | 10            | Find this setting in OpenAM console under Configuration > Servers and Sites > <i>server name</i> > Directory Configuration.<br><br>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 OpenAM console under Configuration > Servers and Sites > <i>server name</i> > CTS > External Store Configuration.<br><br>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. |

| <b>Property</b>  | <b>Default Value</b> | <b>Suggestions</b>   |
|--|----------------------|--|
| CTS connection timeout<br>(advanced property)                      | 10<br>(seconds)      | <p>(org.forgerock.services.cts.store.max-connections)</p> <p>Most CTS requests to the directory server are handled quickly, so the default timeout is fine for most cases.</p>   |
| CTS reaper timeout<br>(advanced property)                          | None                 | <p>If you choose to vary this setting for performance testing, set the advanced property, <code>org.forgerock.services.datalayer.connection.timeout.cts.async</code>, under Configuration &gt; Servers and Sites &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> <p>The CTS token cleanup connection generally should not time out as it is used to request long-running queries that can return many results.</p>                           |
| Configuration management connection timeout<br>(advanced property) | 10<br>(seconds)      | <p>If you choose to vary this setting for performance testing, set the advanced property, <code>org.forgerock.services.datalayer.connection.timeout.cts.reaper</code>, to the number of seconds desired under Configuration &gt; Servers and Sites &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> <p>Most configuration management requests to the directory server are handled quickly, so the default timeout is fine for most cases.</p> |

### 21.1.3 Notification Settings

OpenAM has two thread pools used to send notifications to clients. The Service Management Service thread pool can be tuned in OpenAM console under Configuration > Servers and Sites > Default Server Settings > SDK.

**Table 21.4. SMS Notification Setting**

| Property               | Default Value | Suggestions   |
|------------------------|---------------|---|
| Notification Pool Size | 10            | 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.<br><br>(com.sun.identity.sm.notification.threadpool.size) |

The session service has its own thread pool to send notifications. This is configured under Configuration > Servers and Sites > Default Server Settings > Session.

**Table 21.5. 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.<br><br>(com.iplanet.am.notification.threadpool.size)   |
| 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.<br><br>(com.iplanet.am.notification.threadpool.threshold) |

### 21.1.4 Session Settings

The session service has additional properties to tune, which are configured under Configuration > Servers and Sites > Default Server Settings > Session.

**Table 21.6. Session Settings**

| Property             | Default Value | Suggestions   |
|----------------------|---------------|---|
| Maximum Sessions     | 5000          | In production this value can safely be set into the 100,000s. The maximum session limit is really controlled by the maximum size of the JVM heap which must be tuned appropriately to match the expected number of concurrent sessions.<br><br>(com.iplanet.am.session.maxSessions) |
| Sessions Purge Delay | 0             | This should be zero to ensure sessions are purged immediately.<br><br>(com.iplanet.am.session.purgedelay)   |

## 21.2 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 21.7. Heap Size Settings**

| JVM Parameters | Suggested Value  | Description                    |
|----------------|--|--------------------------------|
| -Xms & -Xmx    | At least 1024m (2048m with embedded OpenDJ), in production environments at least 2048m to 3072m. This setting depends on the available physical memory, and on whether a 32 or 64-bit JVM is used. | -                              |
| -server        | -  | Ensures the server JVM is used |

| JVM Parameters                         | Suggested Value                        | Description   |
|--|--|---|
| -XX:PermSize & -XX:MaxPermSize         | Set both to 256m                       | Controls the size of the permanent generation in the JVM  |
| -Dsun.net.client.defaultReadTimeout    | 60000                                  | Controls the read timeout in the Java HTTP client implementation  |
| -Dsun.net.client.defaultConnectTimeout | High setting:<br>30000<br>(30 seconds) | This applies only to the Sun/Oracle HotSpot JVM.<br><br>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 21.8. Garbage Collection Settings**

| JVM Parameters           | Suggested Value             | Description  |
|--------------------------|-----------------------------|--|
| -verbose:gc              | -                           | Verbose garbage collection reporting                                 |
| -Xloggc:                 | \$CATALINA_HOME/logs/gc.log | Location of the verbose garbage collection log file                  |
| -XX:+PrintClassHistogram | -                           | Prints a heap histogram when a SIGTERM signal is received by the JVM |
| -XX:+PrintGCDetails      | -                           | Prints detailed information about garbage collection                 |

| JVM Parameters                     | Suggested Value                     | Description   |
|------------------------------------|-------------------------------------|---|
| -XX:+PrintGCTimeStamps             | -                                   | Prints detailed garbage collection timings              |
| -XX:+HeapDumpOnOutOfMemoryError    | -                                   | Out of Memory errors generate a heap dump automatically |
| -XX:HeapDumpPath                   | \$CATALINA_HOME/logs/heapdump.hprof | Location of the heap dump                               |
| -XX:+UseConcMarkSweepGC            | -                                   | Use the concurrent mark sweep garbage collector         |
| -XX:+UseCMSCompactAtFullCollection | -                                   | Aggressive compaction at full collection                |
| -XX:+CMSClassUnloadingEnabled      | -                                   | Allow class unloading during CMS sweeps                 |

---

## 21.3 Caching in OpenAM

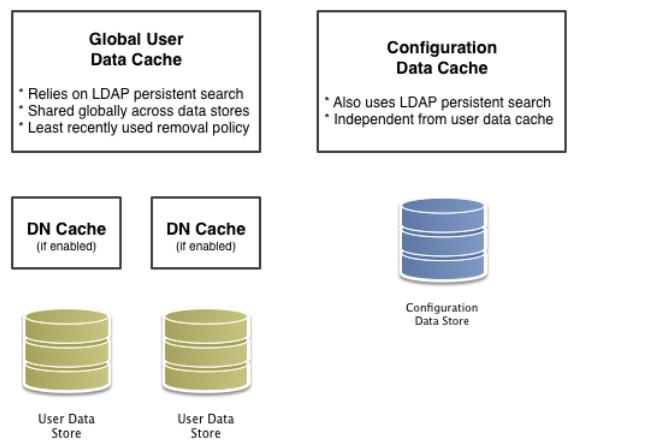
OpenAM caches data to avoid having to query user and configuration data stores each time it needs the information. By default, OpenAM makes use of LDAP persistent search to receive notification of changes to cached data. For this reason, caching works best when data are stored in a directory server that 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.



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 21.1, “LDAP Data Store Settings”](#).

### 21.3.1 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 21.1. 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 OpenAM console, browse to Configuration > Servers and Sites > *Server Name* > Advanced.
2. Set `com.iplanet.am.sdk.caching.enabled` to `false` to disable caching overall.

3. Set `com.sun.identity.sm.cache.enabled` 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 `com.iplanet.am.sdk.caching.enabled` to `false` in the previous step disables both user and configuration data caching.

4. Save your work.

### **Procedure 21.2. 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 OpenAM console, browse to Configuration > Servers and Sites > Default Server Settings > SDK.
2. Change the value of SDK Caching Max. Size, and then Save your work.

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.

### **21.3.2 Caching Properties For Java EE Policy Agents & 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 21.9, “OpenAM Cache Properties”](#). Clients however can receive updates by notification from OpenAM or, if notification fails, by polling OpenAM for changes.

### **Procedure 21.3. To Enable Notification & 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 OpenAM console, browse to Access Control > *Realm Name* > Agents > *Agent Type* > *Agent Name* to view and edit the policy agent profile.
2. 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`.

3. For any changes you make, Save your work.

You must restart the policy agent for the changes to take effect.

### 21.3.3 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 21.9. OpenAM Cache Properties**

| Property   | Description   | Default           | Applies To   |
|--|---|-------------------|--------------|
| <code>com.iplanet.am.sdk.cache.maxSize</code>      | Maximum number of user entries cached   | 10000             | Server & SDK |
| <code>com.iplanet.am.sdk.caching.enabled</code>    | Whether to enable caching for both configuration data and also for user data.<br><br>If <code>true</code> , this setting overrides <code>com.sun.identity.idm.cache.enabled</code> and <code>com.sun.identity.sm.cache.enabled</code> .                         | <code>true</code> | Server & SDK |
| <code>com.iplanet.am.sdk.remote.pollingTime</code> | If <code>false</code> , you can enable caching independently for configuration data and for user data using the aforementioned properties.<br><br>How often in minutes the SDK client such as a policy agent should poll OpenAM for modified user data entries. | 1<br>(minute)     | SDK          |
|  | 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.  |                   |              |

## Cache Settings

---

| <b>Property</b>   | <b>Description</b>  | <b>Default</b> | <b>Applies To</b> |
|---|---|----------------|-------------------|
| <code>com.sun.am.event.notification.expire.time</code>            | In other words, by default the SDK polls for entries changed in the last 3 minutes.<br><br>Set this to 0 or a negative integer to disable polling.  |                |                   |
| <code>com.sun.identity.idm.cache.enabled</code>                   | How long OpenAM stores a given change to a cached entry, so that clients polling for changes do not miss the change.<br><br>If <code>com.iplanet.am.sdk.caching.enabled</code> is true, this property is ignored. | 30 (minutes)   | Server only       |
| <code>com.sun.identity.idm.cache.entry.default.expire.time</code> | If <code>com.iplanet.am.sdk.caching.enabled</code> is true to enable caching of user data.  | false          | Server & SDK      |
| <code>com.sun.identity.idm.cache.entry.expire.enabled</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.remote.notification.enabled</code>     | Whether user data entries in the global user data cache should expire over time   | false          | Server & SDK      |
|   | Whether the SDK client such as a policy agent should register a notification listener for user data changes with the OpenAM server.   | true           | SDK               |
|   | 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.  |                |                   |

## Cache Settings

---

| Property                          | Description  | Default | Applies To   |
|-----------------------------------|--|---------|--------------|
| com.sun.identity.sm.cache.enabled | If com.iplanet.am.sdk.caching.enabled is true, this property is ignored.<br><br>Otherwise, set this to true to enable caching of configuration data. It is recommended that you always set this to true. | false   | Server & SDK |
| sun-idrepo-ldapv3-dncache-enabled | Set this to true to enable DN caching of user data.  | false   | Server & SDK |
| sun-idrepo-ldapv3-dncache-size    | Sets the cache size.   | 1500    | Server & SDK |

---

---

## Chapter 22

# 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 the procedure, [To Set Up Administration Tools](#).

- 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 [Backing Up and Restoring OpenAM Configurations](#) for instructions.

### Procedure 22.1. To Add the New Host Name As an Alias

1. Login to OpenAM console as administrator, `amadmin`.
2. Under Access Control > / (Top Level Realm), add the new host name to the Realm/DNS Aliases list, and then save your work.

---

## Procedure 22.2. To Export, Edit, & 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.iplanet.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://openam.example.com:8080/contextPath`.
- If you are changing the port number, then change the value of `com.iplanet.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 22.3. To Edit OpenAM Configuration Files For the New Host Name**

1. Stop OpenAM or the web container where it runs.
2. Edit the bootstrap file, such as /home/user/openam/bootstrap, changing the 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 22.4. To Remove the Old Host Name As an Alias**

1. Login to OpenAM console as administrator, amadmin.
2. Under Access Control > / (Top Level Realm), remove the old host name from the Realm/DNS Aliases list, and then save your work.



---

## Chapter 23

# Securing OpenAM

This chapter identifies best practices for securing your OpenAM deployment.

### 23.1 Avoiding Obvious Defaults

OpenAM includes default settings to make it easier for you to evaluate the software. Avoid these default settings in production deployments.

- When connecting to LDAP, bind with a specific administrative account rather than a root DN account if possible.
- Change the default iPlanetDirectoryPro cookie name both in OpenAM (`com.iplanet.am.cookie.name`) and in your policy agent profiles (`com.sun.identity.agents.config.cookie.name`).
- When installing OpenAM, do not use `/openam` or `/opensso` as the deployment URI.
- Set valid goto URL domains for OpenAM in the core authentication module configuration. The parameter is described in the section providing *Hints For the Core Authentication Module* (`iplanet-am-auth-valid-goto-domains`).
- Create an administrator in the top-level realm with a different ID than the default `amadmin`.
- Create specific administrator users to track better who makes configuration changes.

- Set the OpenAM advanced property `openam.auth.soap.rest.generic.authentication.exception` to true. This causes OpenAM to return the same exception both when the user does not exist, and also when the password is not valid.
- Remove the demo user account. For example, if you configure the embedded OpenDJ directory server as a configuration and CTS store, the default demo user account gets created during the installation process. You should remove the user using the OpenAM console under Access Control > / (Top Realm) > Subjects > User.
- Set the list of Valid goto URL Resources. By default, OpenAM redirects the user to the URL specified in the `goto` and `gotoOnFail` query string parameters supplied to the authentication interface in the login URL.

To increase security against possible phishing attacks through open redirect, you can specify a list of valid URL resources against which OpenAM validates these URLs. OpenAM only redirects a user if the `goto` and `gotoOnFail` URL matches any of the resources specified in this setting. If no setting is present, it is assumed that the `goto` or `gotoOnFail` URL is valid.

To set the Valid goto URL Resources, use the OpenAM console, and navigate to Access Control > *Realm Name* > Services. Click Add, select Validation Service, and then add one or more valid goto URLs.

When setting valid goto URLs, you can use the "\*" wildcard, where "\*" matches all characters except "?". For more specific patterns, use resource names with wildcards as described in the procedure, [Configuring Valid goto URL Resources](#).

## 23.2 Protecting Network Access

Anytime users interact with a web service, there are risks. With OpenAM, you can reduce those risks by deploying different parts of OpenAM in appropriate parts of an enterprise network.

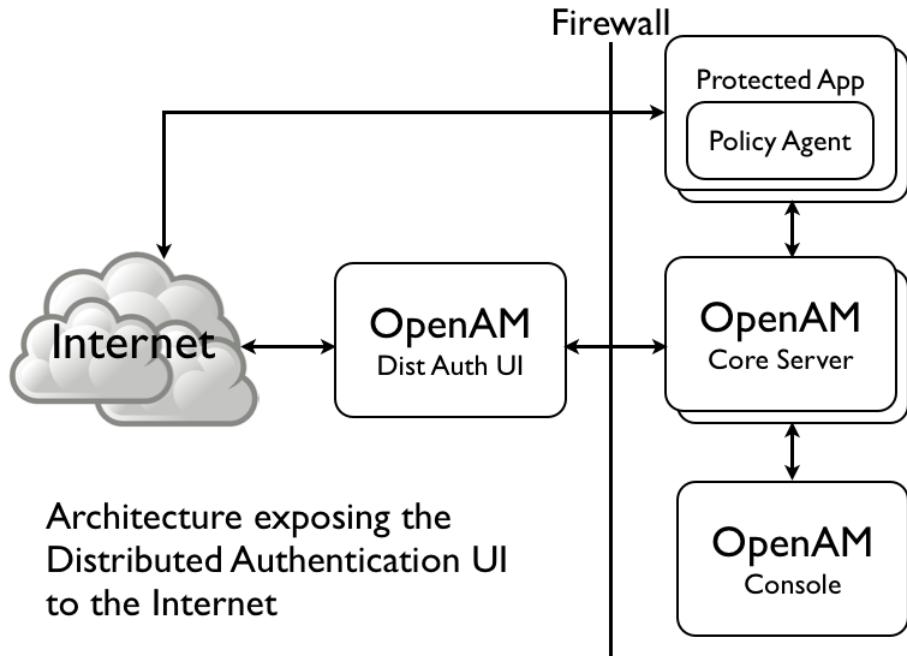
To minimize risks, deploy only the core OpenAM server on systems directly connected through a firewall. As a start, deploy only the core server (and the protected web application) on Internet-facing servers. For instructions, see the following section from the OpenAM Installation Guide, [Determine Which War File to Deploy](#).

You can further limit what is exposed through the firewall by using one of two strategies:

- Set up a distributed authentication user interface (UI) in a DMZ between firewalls.

The distributed authentication UI is essentially a small subset of the OpenAM server with just enough login logic to receive user authentication requests. Those requests are forwarded to the core OpenAM servers.

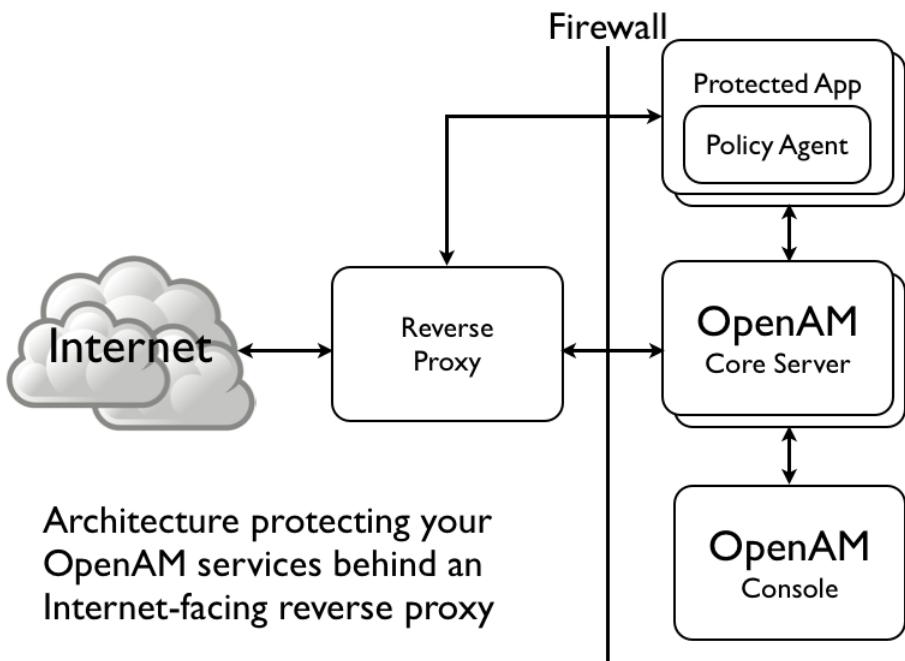
See [Installing OpenAM Distributed Authentication](#) for installation instructions. The following figure shows the recommended architecture.



- Alternatively, use a reverse proxy in front of OpenAM to allow access only to the necessary URLs. The distributed authentication user interface only exposes the authentication UI, while a reverse proxy can expose those endpoints needed for an application. For example, if you need to expose the OAuth2/ OpenID Connect endpoints and REST interface in addition to the authentication UI, then you should implement a reverse proxy. If you only need to expose a login page, then use the DAUI.

Also note that the DAUI does not support SAML 2.0, so in applications that require access to SAML 2.0 endpoints, you must implement a reverse proxy.

The following figure shows the recommended architecture with a reverse proxy.



For access to the console, deploy the full OpenAM application<sup>1</sup> on a separate system that is reachable only from internal systems. Do not include the full OpenAM server in the load-balanced pool of OpenAM servers serving applications.

- Leave `ssoadm.jsp` disabled in production. (Advanced property: `ssoadm.disabled=true`)
- If possible in your deployment, control access to OpenAM console by network address, such that administrators can only connect from well-known systems and networks.
- Restrict access to URIs that you do not use, and prevent internal endpoints such as `/sessionservice` from being reachable over the Internet.

For a full list of endpoints, see the *Reference* chapter on [Service Endpoints](#).

---

<sup>1</sup>Console only deployment is no longer supported.

## 23.3 Securing OpenAM Administration

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. For instructions, see [Configuring Realms](#). 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 realm/DNS aliases, and use them to control access to the realms.
- When customizing `config/auth/default*/Login.jsp`, make sure that you do not introduce any security vulnerabilities such as cross-site scripting due to unvalidated input.
- Create a policy agent profile for each policy agent. See [Configuring Policy Agent Profiles](#) for instructions.

## 23.4 Securing Communications

Keep communications secure by using encryption, properly configured cookies, and request and response signatures.

- Protect network traffic by using HTTPS and LDAPS where possible.
- When using HTTPS, use secure cookies, which are transmitted only over secured connections.

You can configure OpenAM server to use secure cookies by browsing in OpenAM console to Configuration > Servers and Sites > Default Server Settings > Security > Cookie, selecting Yes in the Secure Cookie field, and then clicking Save to keep your work.

HttpOnly cookies are meant to be transmitted only over HTTP and HTTPS, and not through non-HTTP methods such as JavaScript functions.

You can configure OpenAM server to use HttpOnly cookies by browsing in OpenAM console to Configuration > Servers and Sites > Default Server Settings > Advanced, adding the advanced property with name `com.sun.identity.cookie.httponly` and value `true`, and then clicking Save to keep your work.

- Where possible, use subdomain cookies, and control subdomains in a specific DNS master.

- Use cookie hijacking protection with restricted tokens, where each policy agent uses different SSO tokens for the same user. See [To Protect Against CDSSO Cookie Hijacking](#) for instructions.
- When using SAML 2.0:
  - Sign authentication requests, authentication responses, and single logout requests.
  - If the other entities in your circle of trust can handle encryption, then use encryption as well.
  - Use your own key, not the test key provided with OpenAM.

## 23.5 Administering the amadmin Account

You can make changes to the password and user name for the main OpenAM administrative account.

You can change the user name of the `amadmin` administrative account to something more obscure, such as `superroot`. However, the capabilities of that alternative administrative account would not be complete, due to some hard-coding of `amadmin` in the source files. When changing the password for the main OpenAM administrative account, you must make a corresponding change to the authentication datastore. That datastore could be OpenDJ. The steps you would take to change the OpenAM top-level administrative password and account name are shown in the following sections.

### Procedure 23.1. To Change the Password for the Top-Level Administrator (normally `amadmin`)

1. Login to the OpenAM console as the administrator, normally `amadmin`.
2. Under Access Control > / (Top Level Realm) > Subjects > User, select the name of the current top-level administrative user.
3. In the page that appears, navigate to the Password row and click Edit.
4. In the window that appears, enter the desired new password in the New Password and Re-Enter Password text boxes.
5. Click OK to implement the change. If you want to cancel, click Close or just close the window.
6. You'll also need to change the password for the administrator on the directory server. If you are using OpenDJ, refer to the [OpenDJ Administration](#)

*Guide* section on [Resetting Administrator Passwords](#). If you are using a different directory server, you will have to refer to the documentation for that server.

In the following steps, you will identify the new administrative user by assigning it to the `com.sun.identity.authentication.super.user` directive. You may also need to create an OpenAM account for the new administrative user. Don't forget to make sure that new administrative account is configured in the corresponding directory server such as OpenDJ.

### **Procedure 23.2. To Change the Account Name for the Top-Level Administrator (normally amadmin)**

1. Login to the OpenAM console as the administrator, normally `amadmin`.
2. Navigate to the page where you can set the properties for different classes. Select Configuration > Servers and Sites > *Server Name* > Advanced.
3. In the Advanced Properties window that appears, click Add.
4. You'll see blank entries in the end of the list of Property Names and Property Values. In the empty Property Name text box, enter `com.sun.identity.authentication.super.user`.
5. In the corresponding Property Values test box, enter appropriate values for the new administrative user in LDAP Data Interchange Format (LDIF). For example, the following entry would set up an administrative user named `superroot`, in the organizational unit named `people`, associated with the `example.com` domain: `uid=superroot,ou=people,dc=example,dc=com`.
6. Click Save to save the changes that you've made.
7. If the account doesn't already exist in OpenAM or on a connected directory server, you'll need to create it. To do so, select Access Control > / (Top Level Realm) > Subject > User > New. In the New User window that appears, create the new user. Make sure to enter an appropriate password and make that user Active. The ID for that new user is the user name.
8. As noted earlier, you'll also need to make sure that the corresponding account on the directory server has at least CN=Directory Manager privileges. If you're using OpenDJ, refer to the chapter on *Configuring Privileges & Access Control* in the [OpenDJ Administration Guide](#).

If you do change the account name of the top-level administrative account, you should be aware that the original `amadmin` account is "hard-coded" in the source code of several files. The code in these files may affect the functionality of a top-level administrative user with a name other than `amadmin`.

## Administering the amadmin Account

---

One of the improvements that we plan to make to OpenAM is to eliminate these instances of hard-coding. Until we make such improvements, the amadmin user would retain privileges related to the LoginState and some IDM-related classes.

---

## Chapter 24

# Troubleshooting

This chapter covers how to get debugging information and troubleshoot issues in OpenAM deployments.

## Solutions to Common Issues

This section offers solutions to common problems when working with OpenAM.

### 24.1. OpenAM Installation

**Q:** OpenAM configuration could not write to the configuration directory. Where must I change permissions, and what permissions are required?

**A:** If the user running the web container has a \$HOME directory, then the configuration directory is stored there, and you probably do not have this problem. If you do not know the user running the web container, use the **ps** command to check. In the following example, the user is **mark**, the web container **tomcat**.

```
$ ps -ef | grep tomcat
mark      1739      1  0 14:47...
```

For a container installed from native packages with a dedicated user, \$HOME may not be where you think it is. Look at the user's entry in /etc/passwd to locate the home directory. The user running the web container

---

where you install OpenAM must be able to read from and write in this directory.

If you cannot change the permissions to the user's home directory, you can, as a workaround, unpack `OpenAM-12.0.0.war`, set the `configuration.dir` property in the `WEB-INF/classes/bootstrap.properties` to a directory with appropriate permissions, and repack `openam.war` with the adjusted file before deploying that.

```
$ cd ~/Downloads/openam/OpenAM-12.0.0.war  
$ mkdir unpacked ; cd unpacked  
$ jar xf ../OpenAM-12.0.0.war  
$ vi WEB-INF/classes/bootstrap.properties  
$ grep ^config WEB-INF/classes/bootstrap.properties  
configuration.dir=/my/readwrite/config/dir  
$ jar cf ../openam.war *
```

**Q:** Deployment failed due to lack of memory. What do I do?

**A:** OpenAM requires at least a maximum heap size of 1024 MB, with a 256 MB maximum permanent generation heap size. For the Sun JVM, ensure the container starts with `-Xmx1024m -XX:MaxPermSize=256m` for these settings.

If you do not know the settings used when the web container was started, use the `ps` command to check. In the following example, the web container is `tomcat`.

```
$ ps -ef | grep tomcat | grep Xm  
... -Xmx1024m -XX:MaxPermSize=256m ...
```

Make sure you have at least 2 GB of RAM on the system where you run OpenAM to avoid running out of memory.

If you make it through deployment and seem to be running out of memory later, you can confirm memory errors in OpenAM by searching the `config-dir/openam/debug/*` files for `java.lang.OutOfMemoryError`.

**Q:** Deployment failed due to invalid hostname configuration. What do I do?

**A:** OpenAM requires that you use a fully qualified domain name (FQDN) that the host can resolve.

```
$ ping openam-ter.example.com  
PING openam-ter (192.168.56.2) 56(84) bytes of data.  
64 bytes from openam (192.168.56.2): icmp_seq=1 ttl=64 time=0.025 ms  
64 bytes from openam (192.168.56.2): icmp_seq=2 ttl=64 time=0.032 ms  
64 bytes from openam (192.168.56.2): icmp_seq=3 ttl=64 time=0.030 ms
```

---

For a test deployment (at home, on a laptop), you can use fake FQDNs in /etc/hosts (%SystemRoot%\system32\drivers\etc\hosts on Windows), depending on how your network is configured.

```
$ cat /etc/hosts | grep openam  
192.168.56.2 openam openam.example.com  
192.168.56.3 openam-bis openam-bis.example.com  
192.168.56.5 openam-ter openam-ter.example.com
```

- Q:** I configured OpenAM, and now am seeing the configuration screen again. Who deleted my configuration?
- A:** OpenAM uses a file in \$HOME/.openamcfg/ to bootstrap and find its configuration. The file is named after the path to OpenAM and contains the path to the configuration. The following example shows what the file looks like for OpenAM deployed in Apache Tomcat under /path/to/tomcat/webapps/openam, and running as user amuser with \$HOME /home/amuser.

```
$ cat ~/.openamcfg/AMConfig_path_to_tomcat_webapps_openam_  
/home/amuser/openam
```

If OpenAM cannot find its configuration, then it displays the configuration screen.

## 24.2. OpenAM Upgrades

- Q:** I have upgraded OpenAM, now my tools are not working properly. What happened?
- A:** Every OpenAM component must be upgraded, not just the main OpenAM .war file. If you did not upgrade the tools too, they may not work as intended.

## 24.3. OpenAM Administration

- Q:** I cannot use the browser-based equivalent of **ssoadm**, <http://openam.example.com:8080/openam/ssoadm.jsp>. Why not?
- A:** For security reasons, ssoadm.jsp is not activated by default. To activate it, browse to Configuration > Servers and Sites > Servers > *ServerName* > Advanced, and then add a property named ssoadm.disabled with value false.
- Q:** The **ssoadm** command is very, very slow on my virtual machine (VMWare, VirtualBox, etc.). How can I speed it up?

- 
- A:** Virtual machine random devices do not always produce enough random data. The **ssoadm** command can hang while reading random data from the virtual machine's random device, with the result that you can wait a minute or more for a single command to finish.

To work around this limitation on virtual machines, make sure you install something that generates enough random data, such as a [timer entropy daemon](#).

- Q:** I have OpenAM deployed on WebLogic 12.1.1 and am running Java 6. What can I do to fix the exceptions and strange results that I am seeing when I use the **ssoadm** command?
- A:** Edit the start up script for WebLogic as described in the *Installation Guide* section, [Preparing Oracle WebLogic](#), and then restart WebLogic.
- Q:** I added OpenDJ as a data store, and now I cannot add a user. OpenAM gives me the following error.

```
ERROR: LDAPv3Repo.create failed. errorCode=65 Entry
      uid=test,ou=people,dc=example,dc=com violates the Directory Server
      schema configuration because it includes attribute inetUserStatus which
      is not allowed by any of the objectclasses defined in that
      entry
```

- A:** When you set up a New Data Store to use OpenDJ as an identity repository under Access Control > *Realm Name* > Data Stores > New..., you need to check the Load schema when saved box if you want OpenAM to add the schema to OpenDJ. The box is not selected by default.

The full version of OpenAM includes directory server schema in the ~/Downloads/openam/ldif/ directory. To add the schema to OpenDJ afterwards, you can try the following command.

```
$ /path/to/opendj/bin/ldapmodify \
--port 1389 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--filename ~/Downloads/openam/ldif/fam_sds_schema.ldif
Processing MODIFY request for CN=schema
MODIFY operation successful for DN CN=schema
```

- Q:** I have OpenAM installed in WebSphere application server with IBM Java. I am doing REST-based user registration or forgotten password reset, or setting up the HOTP authentication module, sending mail to an SMTP server over SSL.

How come OpenAM cannot send mail over SSL?

- 
- A:** If you see in the OpenAM Authentication debug log that the SSL handshake is failing when connecting to the mail server, then it is likely that the SSL certificate presented by the mail server is not trusted.

This is a WebSphere/IBM Java issue, rather than an OpenAM issue.

To work around the problem, follow these steps to make sure that WebSphere trusts the mail server SSL certificate.

1. Log in as administrator to WebSphere console.
2. Browse to Security > SSL certificate and key management > Manage endpoint security configurations, and then click the link for the node where OpenAM runs.
3. In the menu on the right, click SSL configuration.
4. Click NodeDefaultSSLSettings.
5. In the menu on the right, click Key stores and certificates.
6. Click NodeDefaultTrustStore.
7. In the menu on the right, click Signer certificates.
8. Click Retrieve from port.
9. Set Host, Port, and Alias, and then click Retrieve signer information.

The Host is the host name of the SMTP server.

The Port is the port number of the SMTP server, such as 465.

The certificate Alias can be set to the user name used to authenticate to the mail server.

For example, if you are sending mail through Google mail as my.user, then set Host to `smtp.gmail.com`, set Port to 465, and set Alias to `my.user`.

10. After the information is retrieved, click Apply, save your work, and then restart WebSphere.

After WebSphere restarts, it should trust the mail server SSL certificate. OpenAM therefore should be able to connect to the mail server over SSL.

For more information, see the [WebSphere documentation](#).

- 
- Q:** My container log file is filling up with messages from OpenAM's OAuth authorization service and OpenID Connect provider.

---

What can I do to prevent all these messages from being logged?

- A:** This behavior is governed by the log settings for RESTlet, which is used by OpenAM for OAuth 2.0 and OpenID Connect 1.0.

Use log configuration settings to turn off logging from RESTlet.

For example, if your container is Apache Tomcat, follow these steps.

1. Stop Tomcat.

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

2. Edit the Tomcat settings script, catalina.sh or catalina.bat, to use the logging configuration file.

For example, in /path/to/tomcat/bin/catalina.sh, uncomment the following line, and then save your work:

```
LOGGING_CONFIG= \
    "-Djava.util.logging.config.file=${CATALINA_BASE}/conf/logging.properties"
```

3. In \$CATALINA\_BASE/conf/logging.properties, add the following line, and then save your work:

```
org.restlet.level=OFF
```

4. Start Tomcat.

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

- Q:** I have session failover configured for an OpenAM site. I see many connections in TIME\_WAIT state, and the connections seem to be used only for communication between OpenAM servers in that site. What should I set to have fewer connections in TIME\_WAIT?

- A:** When you have session failover configured for a site, OpenAM servers run health checks against other servers in the same site. By default, the health checks are run every second (1000 milliseconds) with a timeout of 1 second (1000 milliseconds).

If there is network latency between servers in a site, for example if you are running your servers in virtual machines, the default settings might not be

---

right for your deployment. In that case, consider changing the following advanced server properties.

- By lengthening `com.iplanet.am.session.failover.cluster.stateCheck.timeout` and `com.iplanet.am.session.failover.cluster.stateCheck.period` to something longer than the default, you can work around issues with network latency.
- By setting `com.sun.identity.urlchecker.dorequest` to true or false, you can change whether OpenAM performs an HTTP GET request or only checks the Socket connection of `com.sun.identity.urlchecker.targeturl` as a health check.

To set advanced properties, either use the OpenAM console page under Configuration > Servers and Sites > Default Server Settings > Advanced, or set the properties using the **ssoadm update-server-cfg** command as in the following example, which updates the default server configuration:

```
$ ./ssoadm \
update-server-cfg \
--servername default \
--adminid amadmin \
--password-file /tmp/pwd.txt \
--attributevalues com.iplanet.am.session.failover.cluster.stateCheck.timeout=2000
```

**Q:** I want to change the password for the UrlAccessAgent that was assigned during the OpenAM Installation process.

**A:** To change the UrlAccessAgent password, you can use the **ssoadm** that is installed with the OpenAM Administration tools, as described in the *Installing OpenAM Tools Chapter of the Installation Guide*.

You can then change the password for the UrlAccessAgent for the root realm (/) with the following commands:

```
$ cd /path/to/SSOAdminTools/bin
$ ./ssoadm \
set-identity-attrs \
--realm /
--idname amService-UrlAccessAgent \
--idtype user \
--adminid amadmin \
--password-file /tmp/passwd \
--attributevalues userpassword=changeit

Attribute values of identity, amService-UrlAccessAgent of type, user
in realm, / was modified.
```

The new password will take effect the next time you start OpenAM.



---

# OpenAM 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         | In general terms, a service exposing protected resources.   |
|                     | 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.         |
| Application type    | Application types act as templates for creating policy applications.  |

---

|                                       |  |
|---------------------------------------|--|
|                                       | <p>Application types define a preset list of actions and functional logic, such as policy lookup and resource comparator logic.</p>  |
|                                       | <p>Application types also define the internal normalization, indexing logic, and comparator logic for applications.</p>  |
| Attribute-based access control (ABAC) | Access control that is based on attributes of a user, such as how old a user is or whether she 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 2.0 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 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 both .NET and 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                         | Represents an entity that has been authenticated (such as a user, a device, or an application), and thus is distinguished from other entities   |
|                                   | When a <a href="#">Subject</a> successfully authenticates, OpenAM associates the Subject with the Principal.  |
| 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                             | OpenAM unit for organizing configuration and identity information   |
|                                   | 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. |

---

|   |   |
|---|---|
|   | Administrators can delegate realm administration. The administrator assigns administrative privileges to users, allowing them to perform administrative tasks within the realm.   |
| Referral                                  | Allows the policies for an application to be managed in the current realm, and also in sibling and child realms<br><br>During policy evaluation, OpenAM uses referrals to locate the application's policies in other realms in order to find all policies that apply for a given request.   |
| Resource                                  | Something a user can access over the network such as a web page<br><br>Defined as part of policies, these can include wildcards in order to match multiple actual resources.  |
| 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                                   | In OpenAM a user session is the interval that starts with the user authenticating through OpenAM and ends when the user logs out, or when her session is terminated. OpenAM manages user sessions across one or more applications by issuing a session token used to identify the session and by tracking the session state in order to handle session events like logout and timeout, to permit session constraints, and to notify applications involved in SSO when a session ends. |

---

|                        |   |
|------------------------|---|
| Session failover (SFO) | Capability to allow another OpenAM server to manage a session when the OpenAM server that initially authenticated the principal goes offline  |
| Session token          | Unique identifier issued by OpenAM after successful authentication, 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 minimize cross-talk in the site.</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  |
| Subject                | Entity that requests access to a resource   |
|                        | When a Subject successfully authenticates, OpenAM associates the Subject with the <a href="#">Principal</a> 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   |

---

# **Appendix A. Release Levels & Interface Stability**

This appendix includes ForgeRock definitions for product release levels and interface stability.

## **A.1      ForgeRock Product Release Levels**

ForgeRock defines Major, Minor, and Maintenance product release levels. The release level is reflected in the version number. The release level tells you what sort of compatibility changes to expect.

### **Major (version: x.0.0)**

Major releases bring big new features. Major releases can include changes even to Stable interfaces. Major releases can remove previously Deprecated functionality, and in rare cases remove Evolving functionality that has not been explicitly Deprecated. Major releases also include the changes present in previous Minor and Maintenance releases.

### **Minor (version: x.y.0)**

Minor releases might include new features, backwards-compatible changes to Stable interfaces in the same Major release, and incompatible changes to Evolving interfaces. Minor releases can remove previously Deprecated functionality. Minor releases also include the changes present in Maintenance releases.

Maintenance (version: x.y.z)

Maintenance releases can include bug fixes. Maintenance releases are intended to be fully compatible with previous versions from the same Minor release.

## A.2 ForgeRock Product Interface Stability

ForgeRock products support many protocols, APIs, GUIs, and command-line interfaces. Some of these interfaces are standard and very stable. Others offer new functionality that is continuing to evolve.

We realize that you invest in these interfaces, and therefore must know when and how ForgeRock expects them to change. For that reason, ForgeRock defines interface stability labels and uses these definitions in ForgeRock products.

### Stable

This documented interface is expected to undergo only backwards-compatible changes between major releases. Changes are announced at least one minor release before they take effect.

### Evolving

This documented interface is continuing to evolve and so is expected to change, potentially in backwards-incompatible ways even in a minor release. Changes are documented at the time of product release.

While new protocols and APIs are still in the process of standardization, they are Evolving. This applies for example to recent Internet-Draft implementations, and also to newly developed functionality.

### Deprecated

This interface is deprecated and likely to be removed in a future release. For previously stable interfaces, the change was likely announced in a previous release. Deprecated interfaces will be removed from ForgeRock products.

### Removed

This interface was deprecated in a previous release and has now been removed from the product.

### Internal/Undocumented

Internal and undocumented interfaces can change without notice. If you depend on one of these interfaces, contact ForgeRock support or email [info@forgerock.com](mailto:info@forgerock.com) to discuss your needs.

---

# Index

## A

Account lockout  
    configuring, 87  
Active Directory authentication, 18  
Adaptive risk authentication, 21  
Anonymous authentication, 28  
Application  
    Configuring, 98  
Authentication  
    about, 7  
    Active Directory, 21  
    Active Directory module, 18-21  
    Adaptive risk module, 21-28, 28  
    Anonymous module, 28-30, 30  
    Chains, 8  
    chains, 80  
Core  
    account lockout, 37  
    general, 39  
    global attributes, 34  
    persistent cookie (legacy), 36  
    post authentication processing, 41  
    realm attributes, 34  
    security, 40  
Core module, 33  
Data Store, 42  
Data store module, 42  
Device ID (Match) module, 43-48, 48  
Device ID (Save) module, 48  
Federation module, 49  
HMAC One-Time Password (HOTP)  
module, 49-52, 52  
HTTP Basic module, 52  
JDBC module, 52  
LDAP module, 53-57, 57  
Levels, 9  
Modules, 8  
modules  
    configuring, 17  
MSISDN module, 57-59, 59  
OAuth 2.0, 62

OAuth 2.0/OpenID Connect 1.0 module,  
62-67, 67  
Open Authentication (OATH) module, 59  
OpenID Connect 1.0, 68  
OpenID Connect id\_token Bearer module,  
67  
RADIUS module, 72  
Scripted Authentication module, 73  
Secure Attribute Exchange (SAE) module,  
73  
SecurID, 77  
social  
    configuring, 9  
Web Service Security (WSSAuth) module,  
79  
Windows Desktop SSO module, 77  
Windows NT module, 79  
X509 certificate-based module, 30-33, 33  
Authentication levels  
    and session upgrade, 86  
Authorization, 93  
    Configuring, 100, 337, 363

## B

Backup, 419, 421

## C

Caching, 451  
Certificates, 423  
Command line tools overview, 3  
Console overview, 1  
Core authentication  
    account lockout, 37  
    general, 39  
    global attributes, 34  
    persistent cookie (legacy), 36  
    post authentication processing, 41  
    realm attributes, 34  
    security, 40  
Cross-domain single sign on (CDSSO), 281

## D

Dashboard service, 409, 411, 412  
Data store authentication, 42  
Data stores

---

Active Directory, 125  
Active Directory Application Mode (ADAM), 133  
Database Repository (Early Access), 142  
Generic LDAPv3, 145  
OpenDJ, 153  
Oracle DSEE, 161  
Tivoli Directory Server, 171  
Debug logging  
  Level, 440  
  Rotation, 441  
  Service selection, 440  
  Single file, 440  
Delegating administration, 120  
Device ID (Match) authentication, 43  
Device ID (Save) authentication, 48

## E

Enabling ssoadm.jsp, 5  
Exhaustion actions, 88

## F

Federation, 289  
  Changing signing key, 427  
  Configuring, 291, 292, 293, 293, 294, 301, 305, 310, 312, 313, 323  
  Linking accounts, 324  
  OAuth 2.0, 343  
  SAML 1.x, 385  
  SAML 2.0 Single Logout (SLO), 317  
  SAML 2.0 Single Sign-On (SSO), 317  
Federation authentication, 49

## G

Google Apps, 312

## H

HMAC One-Time Password (HOTP) authentication, 49

## J

JDBC authentication, 52

## K

Kerberos, 77

## L

LDAP authentication, 54  
Logging  
  Audit, 253  
  Debug, 439

## M

Monitoring, 431  
  CTS, 435  
  Health check, 439  
  JMX, 433  
  Policy evaluation, 436  
  Sessions, 438  
  SNMP, 435  
MSISDN authentication, 57

## O

OAuth 2.0, 62, 337  
OAuth 2.0 clients  
  configuring, 248  
OAuth 2.0/OpenID Connection 1.0, 62  
Open Authentication (OATH) authentication, 59  
Open Identity Gateway, 181  
OpenAM  
  authenticating to, 83  
OpenID Connect 1.0, 68, 363  
  configuring, 248  
OpenID Connect id\_token Bearer module, 67

## P

Password reset, 265  
Performance, 443  
Persistent Cookie module, 70  
Policy, 93  
  Configuring, 100  
  Delegating management, 117  
  Import, Export, 112  
policy agent profiles  
  agent administrators  
    creating, 185  
    creating, 183  
  delegating creation of, 185  
  kinds of, 182  
  web policy agents

---

- creating, 186
- Policy agents
  - Configuring, 210, 247, 251
  - configuring, 243
  - Group inheritance, 183
  - OAuth 2.0 clients
    - configuring, 247
  - profiles, 181
  - Profiles, 182
  - security token service client, 243
  - web service client
    - configuring, 240
  - web service provider
    - configuring, 236
- Post authentication plugins, 82

## R

- RADIUS module, 72
- Realms, 119
  - Creating, 120
- Referral
  - Configuring, 108
  - Enabling, 108
- REST API, 413
- Restore, 422
- Restoring, 420

## S

- Salesforce CRM, 313
- SAML 1.x, 385
- Scripted authentication module, 73
- Secure Attribute Exchange (SAE) module, 73
- Securing OpenAM, 461
- Session quotas
  - configuring, 88
- Sessions, 441
- Silent installation, 3
- Single Sign-On (SSO), 275
- Social Authentication
  - configuring, 9
  - custom providers
    - configuring, 12
- implementations service
  - configuring, 15
- pre-populating providers, 10
  - configuring, 11

- SSL, 423

## T

- Troubleshooting, 469

## V

- Valid goto URL resources
  - configuring, 90

## W

- web policy agents
  - advanced properties
    - configuring, 205
  - application properties
    - configuring, 191
  - general properties, 188
  - global properties
    - configuring, 186
  - miscellaneous properties
    - configuring, 202
  - OpenAM services properties
    - configuring, 197
  - SSO properties
    - configuring, 196
- Web Service Security (WSSAuth) module, 79
- Windows Desktop SSO module, 77
- Windows NT module, 79

## X

- X509 Certificate-based authentication, 30

## Z

- Zero page login, 40

