



Setup Guide

/ ForgeRock Access Management 7.0.2

Latest update: 7.0.2

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

Copyright © 2019-2021 ForgeRock AS.

Abstract

Guide showing you how to set up ForgeRock® Access Management (AM). ForgeRock Access Management provides intelligent authentication, authorization, federation, and single sign-on functionality.



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

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

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

UNLESS OTHERWISE MUTUALLY AGREED BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, 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 NON-INFRINGEMENT 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 NON-INFRINGEMENT 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.

FontAwesome Copyright

Copyright (c) 2017 by Dave Gandy, <https://fontawesome.com/>.

This Font Software is licensed under the SIL Open Font License, Version 1.1. See <https://opensource.org/licenses/OFL-1.1>.

Table of Contents

Overview	iv
1. Administration Interfaces and Tools	1
Web-Based AM Console	1
Amster Command-Line Tool	5
Deprecated Command-Line Tools	5
2. Realms	8
Configuring Realms (Console)	9
Configuring Realms (REST)	12
3. Identity Stores	36
Testing External Identity Repository Access	101
Customizing Identity Stores	102
4. Policy and Application Stores	118
Preparing Policy and Application Stores	119
Setting Up Policy and Application Stores.	126
5. Load Balancers	130
Configuring Site Sticky Load Balancing	131
Handling HTTP Request Headers	132
6. Dashboards	134
Implementing the Dashboard Service	135
Glossary	140





Overview

This guide describes how to set up core AM functionality in AM both after a fresh install or when you are growing your environment.

This guide covers tasks you need to perform after installing AM, such as configuring additional identity and data stores and creating and configuring AM's administrative units, *realms*.

This guide is written for anyone installing and configuring Access Management during its lifecycle.

Quick Start

 Administrative Tools Learn about the AM console, Amster, and other tools.	 Realms Use realms to logically organize different groups of users. For example, create one realm for customers, and another for employees.
 Identity Stores Connect identity stores to AM and customize them if needed.	 Policy and Application Stores Learn about external policy and application data stores, and decide if your environment would benefit from having dedicated stores for those purposes.

About ForgeRock Identity Platform™ Software

ForgeRock Identity Platform™ serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see <https://www.forgerock.com>.

Chapter 1

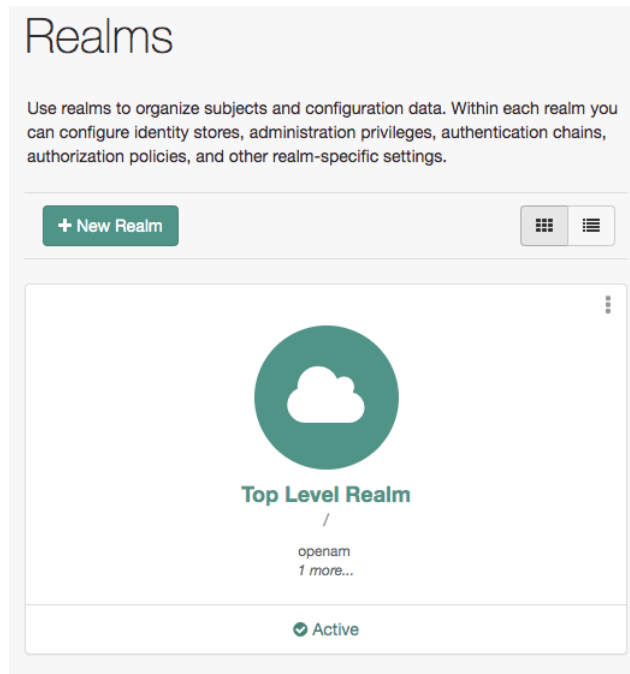
Administration Interfaces and Tools

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

Web-Based AM Console

After you install AM, log in to the web-based AM console as AM administrator, `amAdmin` with the password you set during installation. Navigate to a URL, such as `https://openam.example.com:8443/openam`. In this case, communications proceed over the HTTP protocol to a FQDN (`openam.example.com`), over a standard Java web container port number (8080), to a specific deployment URI (`/openam`).

The AM Console

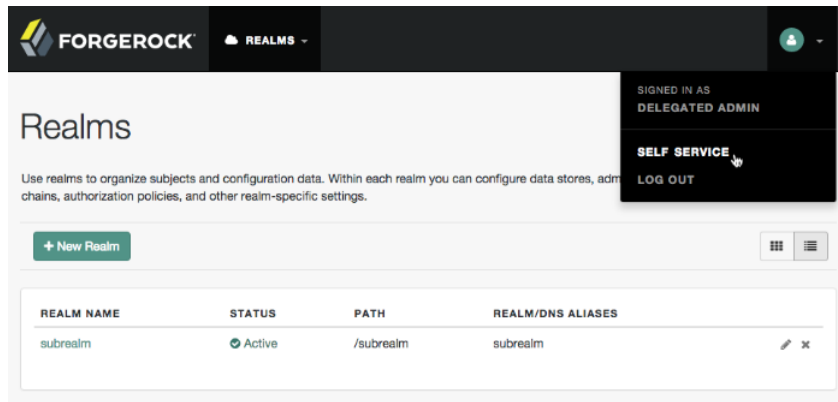


When you log in as the AM administrator, `amAdmin`, you have access to the complete AM console. In addition, AM has set a cookie in your browser that lasts until the session expires, you logout, or you close your browser.¹

The `amAdmin` account is a special user built-in to AM. The `amAdmin` account does not have a user profile and is not present in the configured identity store, so cannot use functionality that requires a user profile, such as Device Match or Push notifications. You should create users or groups, and delegate administrator privileges to them, as described in "Delegating Privileges" in the *Security Guide*.

If you configure AM to grant administrative capabilities to users that do have a user profile and appear in the configured identity store, then that user is able to access both the administration console in the realms they can administrate, and their self-service profile pages:

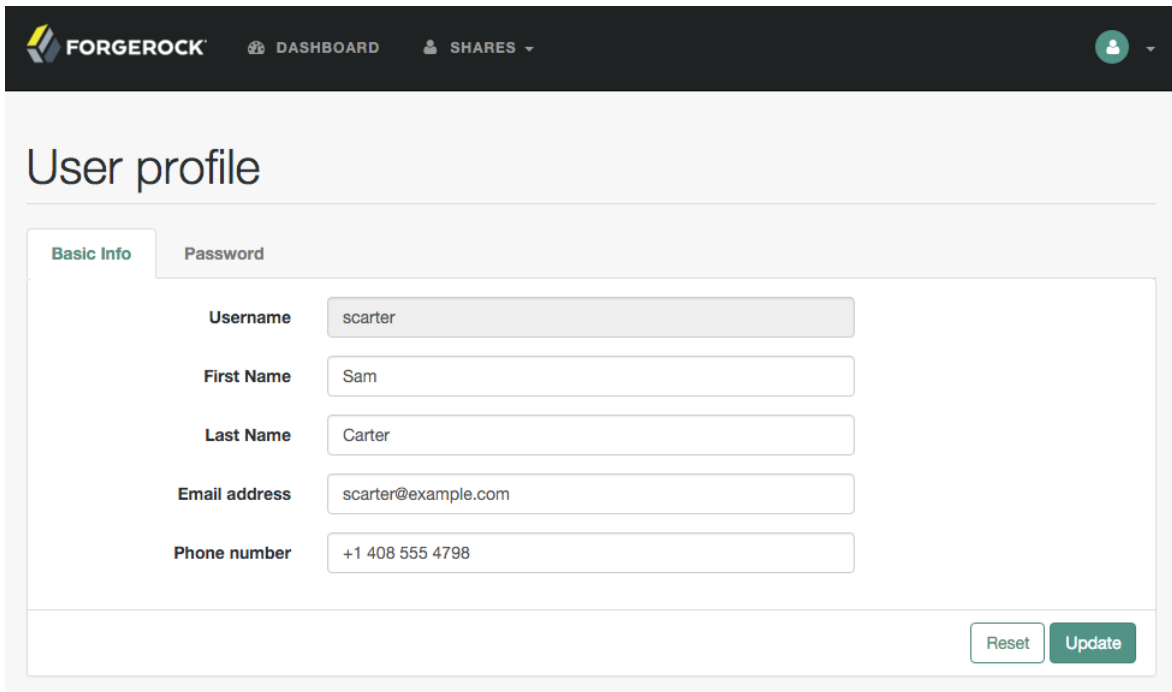
The AM Console for a Delegated Administrator



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

¹Persistent cookies can remain valid when you close your browser. This section reflects AM default behavior before you configure additional functionality.

The AM Console for Non-Administrative Users



The profile attribute whitelist controls the information returned to non-administrative users when accessing `json/user` endpoints. For example, the whitelist controls the attributes shown in the user profile page.

Common profile attributes are whitelisted by default, but you need to add any custom attribute you want your non-administrative users to see.

The whitelist can be set by realm, in the user self-service service, or globally. To modify it:

- **Globally:** Navigate to Configure > Global Services > User Self-Service > Profile Management, and edit the Self readable attributes field.
- **By realm:** Navigate to Realms > *Realm Name* > Services > User Self-Service > Profile Management, and edit the Self readable attributes field.

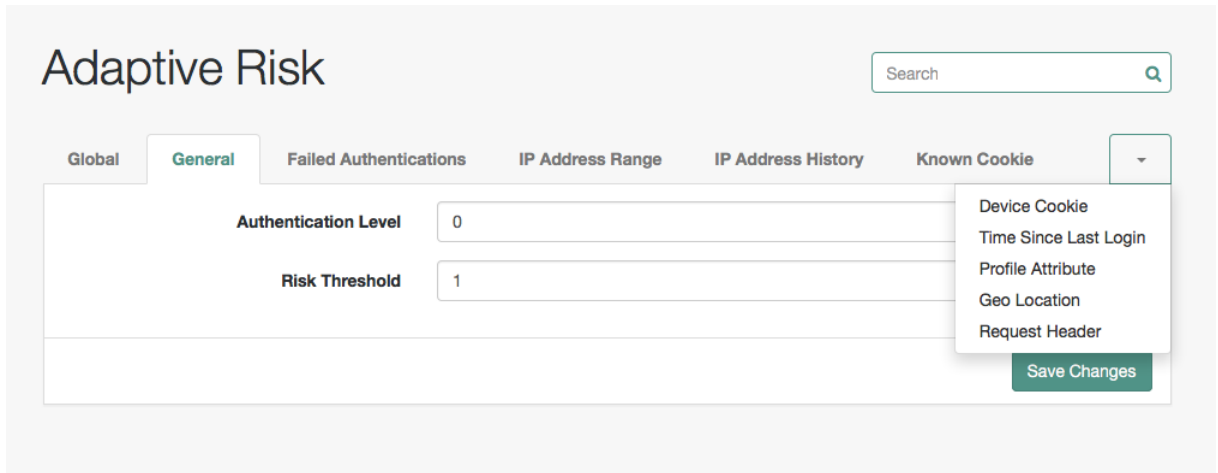
Note that you need to add the user self-service service to the realm if you have not done so already, but you do not need to configure anything other than the whitelist.

AM Console Responsiveness

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

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

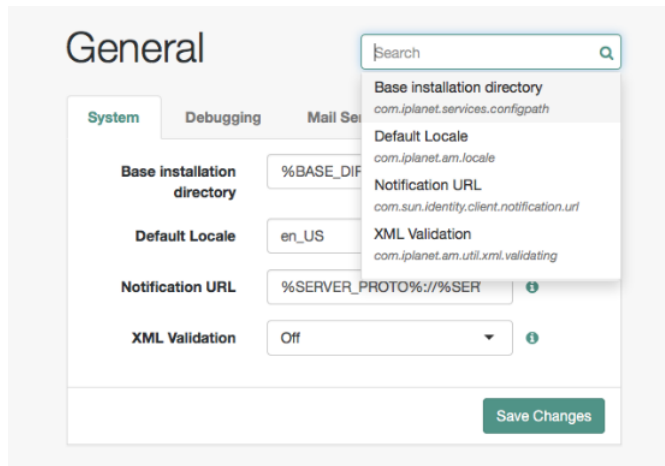
AM Console Responsiveness



The AM Console Search Feature

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

The AM Console Search Feature



Amster Command-Line Tool

The **amster** tool provides a lightweight command-line interface, ideal for use in DevOps processes, such as continuous integration and deployment. The **amster** tool manages an AM configuration over REST, so you can store AM server configuration as an artifact and import a stored configuration to set up an AM server.

For details, see the [amster documentation](#).

Deprecated Command-Line Tools

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

You can install the following command-line tools:

ampassword

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

Install this from the [AM-SSOAdminTools-5.1.3.11.zip](#).

amverifyarchive

This tool checks log archives for tampering.

Install this from [AM-SSOAdminTools-5.1.3.11.zip](#).

openam-configurator-tool-14.1.3.11.jar

This executable `.jar` file lets you perform a silent installation of an AM server with a configuration file. For example, the `java -jar configurator.jar -f config.file` command couples the `configurator.jar` archive with the `config.file`. The `sampleconfiguration` file provided with the tool is set up with the format for the `config.file`, and it must be adapted for your environment.

Install this from `AM-SSOConfiguratorTools-5.1.3.11.zip`.

ssoadm

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

Install this from `AM-SSOAdminTools-5.1.3.11.zip`.

To translate settings applied in the AM console to service attributes for use with **ssoadm**, log in to the AM console as `amAdmin` and access the services page, in a URL, such as `https://openam.example.com:8443/openam/services.jsp`.

The commands access the AM 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.ipplanet.am.naming.map.site.to.server=https://lb.example.com:443/openam=
http://server1.example.com:8080/openam"
```

Node 2:

```
-D"com.ipplanet.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 AM 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
```

Chapter 2

Realms

AM *realms* are administrative units that group configuration and identities together.

The AM installation process creates the Top Level Realm (/), which contains AM default configuration data. This realm cannot be deleted or renamed, since it is the root of the realm hierarchy in AM.

The Top Level Realm can contain sub-realms, which can also contain sub-realms.

Realms are associated with an identity store and, at least, an authorization chain or tree. When you create a new realm, AM copies part of the configuration of the parent realm to the new realm. For example, authentication trees are copied over, as well as the configuration of identity, policy, and application stores. When stored in the configuration store, policies are also copied over.

Once the realm is created, its configuration is separate from that of the parent realm. Configuration changes done to the parent realm or to the new realm will not be shared, with an exception; realms that share external stores will also share the configuration or data kept in the store.

For example, identities, groups, and privileges are linked to identity stores. Two realms that share a identity store will also share identity groups and the privileges granted to them. In the same way, two realms sharing the same policy store will share policy configuration, and two realms sharing the application store will share OAuth 2.0 client configuration.

Consider the following best practices when configuring realms:

Separate Administrative Users from End Users

Create realms for your organizations and separate administrative users from end users, keeping the administrative users constrained to the Top Level Realm. This way, in the event that one of your identity stores is compromised, malicious users will not have the credentials of any AM administrator, which could further compromise your platform.

For this configuration to work, you need an identity store in the Top Level Realm that contains your administrative users and different identity store(s) for other realms/users.

Create Realms to Isolate Identities

Keep users with different authentication and authorization requirements separate. For example, teachers in a University department would have access to more sensitive data than students do. Configuring a realm for teachers and another realm for students allows you to enforce stricter security policies for teachers.

Keep the Top-level Realm for Administration Purposes Only

The Top Level Realm should not contain federation configuration, agent profiles, OAuth 2.0/OpenID Connect/UMA providers, or STS services.

When working with multiple realms, you need direct the users to the correct realm for authentication. You can either:

- Use the `realm=realm-name` query string parameter.
- Create fully qualified domain name DNS aliases for the realms.

Caution

Creating DNS aliases for the realms does not make them more secure. An authenticated user that knows your AM configuration can navigate to other realms, such as the Top Level Realm, by adding the `?realm=/` parameter to their URL.

Configuring Realms



Configure Realms Using the AM Console

Learn about the AM console, Amster, and other tools.



Configure Realms Using REST

Use realms to logically organize different groups of users. For example, create one realm for customers, and another for employees.

Configuring Realms (Console)

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

Note

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

This section has the following procedures:

- "To Create a New Realm"
- "To Configure DNS Aliases for Accessing a Realm"

To Create a New Realm

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

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

Note

Realm names must not match any of the following:

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

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

4. The Active button is enabled by default.

Warning

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

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

Default: the top level realm (/).

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

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

Tip

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

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

8. To enable client-based sessions for the realm, toggle the Use Client-based Sessions switch. For more information on sessions, see *"Introducing Sessions"* in the *Sessions Guide*.
9. Click Create to save your configuration.

To Configure DNS Aliases for Accessing a Realm

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

For example, consider a deployment with the following characteristics:

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

Without applying DNS aliases to the relevant realm, when a user visits `http://realm1.example.com:8080/openam`, AM redirects that user to the top level realm, `https://openam.example.com:8443/openam`. If the authenticating user is present only in `realm1`, then authentication fails even with correct credentials.

If no DNS alias is configured for a realm, `realm1` users must visit URLs such as `https://openam.example.com:8443/openam/XUI/?realm=/realm1#login`. This format of URL reveals the top level realm, and exposes extra information about the service.

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

Note

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

1. Add the domains that AM services to the list of domains that created cookies will be applicable to, as follows:
 - a. Log in to the AM console as an AM administrator, for example, `amAdmin`.
 - b. Navigate to Configure > Global Services > Platform.
 - c. In Cookie Domains, enter the domains that AM will service.

For example, if you installed AM at `openam.example.net`, and intend to have realms associated with FQDNs `realm1.example.org` and `realm2.example.com`, then the Cookie Domains list would include `example.net`, `example.org`, and `example.com`.

2. Set the FQDN for each realm as follows:
 - a. Navigate to Realms > *Realm Name*, and then click Properties.
 - b. In DNS Aliases, enter one or more FQDN values for the realm.
 - c. Save your changes.
3. (Optional) Adding DNS aliases by using the AM console also adds FQDN mappings to the AM server.

To verify these have been created perform the following steps:

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

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

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

- c. Save your changes.

The new realm aliases take effect immediately, it is not necessary to restart AM. You can now use a URL such as `http://realm1.example.com:8080/openam` to access `realm1`, rather than `https://openam.example.com:8443/openam/XUI/?realm=/realm1#login`.

Configuring Realms (REST)

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

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

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

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

When making the REST API call, pass the session token in the HTTP header. For more information about the AM session token and its use in REST API calls, see "Using the Session Token After

Authentication" in the *Authentication and Single Sign-On Guide*. For general information about the REST API, see *Getting Started with REST*.

Tip

To make REST requests to a specific realm, see "*Specifying Realms in REST API Calls*" in the *Getting Started with REST*.

Identity Management

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

Important

AM is not primarily an identity store, nor is it provisioning software. For storing identity data, consider ForgeRock Directory Services. For provisioning, consider ForgeRock Identity Management. Both of these products provide REST APIs as well.

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

Examples in this section do not repeat the authentication shown in "*Authenticating (REST)*" in the *Authentication and Single Sign-On Guide*. For browser-based clients, you can rely on AM cookies rather than construct the header in your application. Managing agent profiles, groups, and users with these APIs requires authentication. The examples shown in this section were performed with the token ID gained after authenticating as an AM administrator, for example `amAdmin`.

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

The following sections cover this JSON-based API:

- "Creating Identities"
- "Reading Identities"
- "Updating Identities"
- "Deleting Identities"
- "Listing Identities"
- "Retrieving Identities Using the Session Cookie"
- "Changing Passwords"
- "Creating Groups"
- "Adding a User to a Group"

Creating Identities

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

The following example shows an administrator creating a new user. The only required fields are `username` and `userpassword`. If no other name is provided, `_id`, `cn`, `sn`, and `uid` are all set to the value of `username`. Values provided for `uid` are not used.

```
$ curl \
--request POST \
--header "Accept-API-Version: protocol=2.1,resource=3.0" \
--header "Content-Type: application/json" \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
--data \
'{
  "username": "bjensen",
  "userpassword": "secret12",
  "mail": "bjensen@example.com"
}' \
https://openam.example.com:8443/openam/json/realms/root/users/?_action=create
{
  "_id": "bjensen",
  "_rev": "-588258934",
  "username": "bjensen",
  "realm": "/",
  "uid": [
    "bjensen"
  ],
  "mail": [
    "bjensen@example.com"
  ],
  "universalid": [
    "id=bjensen,ou=user,dc=openam,dc=forgerock,dc=org"
  ],
  "objectClass": [
    "iplanet-am-managed-person",
    "inetuser",
    "sunFederationManagerDataStore",
    "sunFMSAML2NameIdentifier",
    "inetorgperson",
    "sunIdentityServerLibertyPPService",
    "devicePrintProfilesContainer",
    "iplanet-am-user-service",
    "iPlanetPreferences",
    "pushDeviceProfilesContainer",
    "forgerock-am-dashboard-service",
    "organizationalperson",
    "top",
    "kbaInfoContainer",
    "person",
    "sunAMAuthAccountLockout",
    "oathDeviceProfilesContainer",
    "iplanet-am-auth-configuration-service"
  ],
  "inetUserStatus": [
    "Active"
  ]
}
```

```

    ],
    "dn": [
      "uid=bjensen,ou=people,dc=openam,dc=forgerock,dc=org"
    ],
    "cn": [
      "bjensen"
    ],
    "sn": [
      "bjensen"
    ],
    "createTimestamp": [
      "20180426120642Z"
    ]
  }

```

When **LDAP User Search Attribute** and **Authentication Naming Attribute** are set to different attributes, AM treats **_id** and **username** as distinct values. In this case, **_id** is mapped to **LDAP User Search Attribute**, which is autogenerated if not specified in the payload, and **username** is mapped to **Authentication Naming Attribute**.

For example, given the same payload as above, if **LDAP User Search Attribute** is set to **cn**, the user data is set using the generated UUID as follows:

```

{
  "_id": "f3377274-99e4-44f3-8578-0a09914368fc",
  "_rev": "-1",
  "realm": "/",
  "username": "bjensen",
  "uid": [
    "bjensen"
  ],
  "mail": ["bjensen@example.com"],
  "universalid": [
    "id=f3377274-99e4-44f3-8578-0a09914368fc,ou=user,dc=openam,dc=forgerock,dc=org"
  ],
  "objectClass": [
    ...
  ],
  "inetUserStatus": [
    "Active"
  ],
  "dn": [
    "cn=f3377274-99e4-44f3-8578-0a09914368fc,ou=people,dc=openam,dc=forgerock,dc=org",
  ],
  "cn": [
    "f3377274-99e4-44f3-8578-0a09914368fc",
  ],
  "sn": [
    "bjensen"
  ],
  "createTimestamp": [
    "20220608100442Z"
  ]
}

```

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

```
$ curl \
--request PUT \
--header "Accept-API-Version: protocol=2.1,resource=3.0" \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
--header "Content-Type: application/json" \
--header "If-None-Match: *" \
--data \
'{
  "username": "janedoe",
  "userpassword": "secret12",
  "mail": "janedoe@example.com"
}' \
https://openam.example.com:8443/openam/json/realms/root/users/janedoe
{
  "_id": "janedoe",
  "_rev": "-1686380958",
  "username": "janedoe",
  "realm": "/",
  "uid": [
    "janedoe"
  ],
  "mail": [
    "janedoe@example.com"
  ],
  "universalid": [
    "id=janedoe,ou=user,dc=openam,dc=forgerock,dc=org"
  ],
  "objectClass": [
    "iplanet-am-managed-person",
    "inetuser",
    "sunFederationManagerDataStore",
    "sunFMSAML2NameIdentifier",
    "inetorgperson",
    "sunIdentityServerLibertyPPService",
    "devicePrintProfilesContainer",
    "iplanet-am-user-service",
    "iPlanetPreferences",
    "pushDeviceProfilesContainer",
    "forgerock-am-dashboard-service",
    "organizationalperson",
    "top",
    "kbaInfoContainer",
    "person",
    "sunAMAuthAccountLockout",
    "oathDeviceProfilesContainer",
    "iplanet-am-auth-configuration-service"
  ],
  "dn": [
    "uid=janedoe,ou=people,dc=openam,dc=forgerock,dc=org"
  ],
  "inetUserStatus": [
    "Active"
  ],
  "cn": [
    "janedoe"
  ],
  "sn": [
    "janedoe"
  ]
}
```

```

    ],
    "createTimestamp": [
      "20180426121152Z"
    ]
  }

```

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

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

```

$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
--data '{
  "username": "newGroup"
}' \
https://openam.example.com:8443/openam/json/realms/root/groups?_action=create
{
  "username": "newGroup",
  "realm": "/",
  "uniqueMember": [
    "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
  ],
  "cn": [
    "newGroup"
  ],
  "dn": [
    "cn=newGroup,ou=groups,dc=openam,dc=forgerock,dc=org"
  ],
  "objectclass": [
    "groupofuniquenames",
    "top"
  ],
  "universalid": [
    "id=newGroup,ou=group,dc=openam,dc=forgerock,dc=org"
  ]
}

```

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

Reading Identities

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

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

Note

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

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

```
$ curl \
--header "iPlanetDirectoryPro: AQIC5w...2NzEz*" \
https://openam.example.com:8443/openam/json/realms/root/users/demo
{
```

```

    "_id": "demo",
    "_rev": "-320505756",
    "username": "demo",
    "realm": "/",
    "uid": [
        "demo"
    ],
    "universalid": [
        "id=demo,ou=user,dc=openam,dc=forgerock,dc=org"
    ],
    "objectClass": [
        "iplanet-am-managed-person",
        "inetuser",
        "sunFederationManagerDataStore",
        "sunFMSAML2NameIdentifier",
        "devicePrintProfilesContainer",
        "inetorgperson",
        "sunIdentityServerLibertyPPService",
        "iPlanetPreferences",
        "pushDeviceProfilesContainer",
        "iplanet-am-user-service",
        "forgerock-am-dashboard-service",
        "organizationalperson",
        "top",
        "kbaInfoContainer",
        "sunAMAuthAccountLockout",
        "person",
        "oathDeviceProfilesContainer",
        "iplanet-am-auth-configuration-service"
    ],
    "dn": [
        "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
    ],
    "inetUserStatus": [
        "Active"
    ],
    "sn": [
        "demo"
    ],
    "cn": [
        "demo"
    ],
    "createTimestamp": [
        "20170105101638Z"
    ],
    "modifyTimestamp": [
        "20170110102632Z"
    ]
}

```

Use the `_fields` query string parameter to restrict the list of attributes returned. This parameter takes a comma-separated list of JSON object fields to include in the result. Note that the `_fields` argument is case-sensitive. In the following example, the returned value matches the specified argument, `uid`, whereas `UID` would not:

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

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

Updating Identities

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

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

```
$ curl \
--request PUT \
--header "iPlanetDirectoryPro: AQIC5...Y3MTAx*" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: protocol=2.1,resource=3.0" \
--header "If-Match: *" \
--data '{ "mail": "demo@example.com" }' \
https://openam.example.com:8443/openam/json/realms/root/users/demo
{
  "username": "demo",
  "realm": "/",
  "uid": [
    "demo"
  ],
  "mail": [
    "demo@example.com"
  ],
  "universalid": [
    "id=demo,ou=user,dc=openam,dc=forgerock,dc=org"
  ],
  "objectClass": [
    "iPlanet-am-managed-person",
    "inetuser",
    "sunFederationManagerDataStore",
    "sunFMSAML2NameIdentifier",
    "devicePrintProfilesContainer",
    "inetorgperson",
    "sunIdentityServerLibertyPPService",
    "iPlanetPreferences",
    "pushDeviceProfilesContainer",
    "iPlanet-am-user-service",
    "forgerock-am-dashboard-service",
    "organizationalperson",
    "top",
  ]
}
```



```

    "kbaInfoContainer",
    "sunAMAuthAccountLockout",
    "person",
    "oathDeviceProfilesContainer",
    "iplanet-am-auth-configuration-service"
  ],
  "dn": [
    "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
  ],
  "inetUserStatus": [
    "Active"
  ],
  "sn": [
    "demo"
  ],
  "cn": [
    "demo"
  ],
  "createTimestamp": [
    "20170105101638Z"
  ],
  "modifyTimestamp": [
    "20170110105038Z"
  ],
  "roles": [
    "ui-self-service-user"
  ]
}

```

As shown in the example, AM returns the JSON representation of the profile on success. On failure, AM returns a JSON representation of the error including the [HTTP status code](#).

You can use HTTP PUT to update other objects as well, such as groups:

Notice in the following example, which updates `newGroup`, the object class value is not included in the JSON sent to AM:

```

$ curl \
--request PUT \
--header "iPlanetDirectoryPro: AQIC5...Y3MTAx*" \
--header "Content-Type: application/json" \
--data '{
  "uniquemember": [
    "uid=newUser,ou=people,dc=openam,dc=forgerock,dc=org",
    "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
  ]
}' \
https://openam.example.com:8443/openam/json/realms/root/groups/newGroup
{
  "name": "newGroup",
  "realm": "/",
  "uniqueMember": [
    "uid=newUser,ou=people,dc=openam,dc=forgerock,dc=org",
    "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
  ],
  "cn": [
    "newGroup"
  ]
}

```

```
{
  "dn": [
    "cn=newGroup,ou=groups,dc=openam,dc=forgerock,dc=org"
  ],
  "objectclass": [
    "groupofuniquenames",
    "top"
  ],
  "universalid": [
    "id=newGroup,ou=group,dc=openam,dc=forgerock,dc=org"
  ]
}
```

Deleting Identities

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

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

```
$ curl \
--request DELETE \
--header "Accept-API-Version: protocol=2.1,resource=3.0" \
--header "iPlanetDirectoryPro: AQIC5w...2NzEz*" \
https://openam.example.com:8443/openam/json/realms/root/users/bjensen
{
  "_id": "bjensen",
  "_rev": "-1870449267",
  "success": "true"
}
```

On success, AM returns a JSON object indicating success. On failure, AM returns a JSON representation of the error including the HTTP status code.

You can use this same logic for other resources such as performing an HTTP DELETE on a group:

```
$ curl \
--request DELETE \
--header "iPlanetDirectoryPro: AQIC5w...2NzEz*" \
--header "Accept-API-Version: resource=1.0" \
https://openam.example.com:8443/openam/json/realms/root/groups/newGroup
{
  "success": "true"
}
```

Important

Deleting a user does not automatically remove any of the user's sessions. If you are using CTS-based sessions, you can remove a user's sessions by checking for any sessions for the user and then removing them using the

console's Sessions page. If you are using client-based sessions, you cannot remove users' sessions; you must wait for the sessions to expire.

Listing Identities

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

```
$ curl \
--header "iPlanetDirectoryPro: AQIC5w...2NzEz*" \
"https://openam.example.com:8443/openam/json/realms/root/users?_queryId="
{
  "result":[
    {
      "username":"amAdmin",
      "realm":"dc=openam,dc=forgerock,dc=org",
      "sunIdentityMSISDNNumber":[

      ],
      "mail":[

      ],
      "sn":[
        "amAdmin"
      ],
      "givenName":[
        "amAdmin"
      ],
      "universalid":[
        "id=amAdmin,ou=user,dc=openam,dc=forgerock,dc=org"
      ],
      "cn":[
        "amAdmin"
      ],
      "iplanet-am-user-success-url":[

      ],
      "telephoneNumber":[

      ],
      "roles":[
        "ui-global-admin",
        "ui-realm-admin"
      ],
      "iplanet-am-user-failure-url":[

      ],
      "inetuserstatus":[
        "Active"
      ],
      "postalAddress":[

      ],
      "dn":[
        "uid=amAdmin,ou=people,dc=openam,dc=forgerock,dc=org"
      ],
      "employeeNumber":[

      ]
    }
  ]
}
```

```

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

```

```
{
  "questionId": "1",
  "answer": {
    "$crypto": {
      "value": {
        "algorithm": "SHA-256",
        "data": "cfYYzi9U...rVfFl0Tdw0iX"
      },
      "type": "salted-hash"
    }
  }
},
"dn": [
  "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
],
"universalid": [
  "id=demo,ou=user,dc=openam,dc=forgerock,dc=org"
],
"modifyTimestamp": [
  "20160113010610Z"
]
}
],
"resultCount": 2,
"pagedResultsCookie": null,
"totalPagedResultsPolicy": "NONE",
"totalPagedResults": -1,
"remainingPagedResults": -1
}
```

The `users` endpoint also supports the `_queryFilter` parameter to alter the returned results. For more information, see "Query" in the *Getting Started with REST*.

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

```
$ curl \
--header "iPlanetDirectoryPro: AQIC5w...2NzEz*" \
--header "Accept-API-Version: resource=1.0" \
"https://openam.example.com:8443/openam/json/realms/root/groups?_queryId="
{
  "result" : [ "newGroup", "anotherGroup" ],
  "resultCount" : 2,
  "pagedResultsCookie" : null,
  "remainingPagedResults" : -1
}
```

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

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

Retrieving Identities Using the Session Cookie

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

```
$ curl \
--verbose \
--request POST \
--header "Content-Type: application/json" \
--header "Accept-API-Version: protocol=2.1,resource=3.0" \
--header "iPlanetDirectoryPro: AQIC5wM2LY4Sfcz...c50Dk4MjYzMzA2MQ..*" \
https://openam.example.com:8443/openam/json/realms/root/users?_action=idFromSession
{
  "id": "demo",
  "realm": "/",
  "dn": "id=demo,ou=user,dc=openam,dc=forgerock,dc=org",
  "successURL": "/openam/console",
  "fullLoginURL": "/openam/UI/Login?realm=%2F"
}
```

Changing Passwords

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

Note

Changing the top-level administrator's password requires a more complex procedure. See "Changing the amAdmin Password (Console)" in the *Security Guide* for more information.

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

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

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

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "Accept-API-Version: protocol=2.1,resource=3.0" \
--header "iPlanetDirectoryPro: AQIC5w...NTcy*" \
--data '{
  "currentpassword": "Ch4ng31t",
  "userpassword": "password"
}' \
https://openam.example.com:8443/openam/json/realms/root/users/demo?_action=changePassword
{}
```

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

On failure, AM returns a JSON representation of the error including the HTTP status code. See also "HTTP Status Codes" in the *Getting Started with REST* for more information.

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

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

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

```
$ curl \
--request PUT \
--header "iPlanetDirectoryPro: AQIC5w...NTcy*" \
--header "Accept-API-Version: protocol=2.1,resource=3.0" \
--header "Content-Type: application/json" \
--data '{
  "userpassword":"cangetin"
}' \
https://openam.example.com:8443/openam/json/realms/root/users/demo
{
  "_id":"demo",
  "_rev":"-1942782480",
  "username":"demo",
  "realm":"/",
  "uid":[
    "demo"
  ],
  "mail":[
    "demo@example.com"
  ],
  "universalid":[
    "id=demo,ou=user,dc=openam,dc=forgerock,dc=org"
  ],
  "objectClass":[
    "iplanet-am-managed-person",
    "inetuser",
    "sunFederationManagerDataStore",
    "sunFMSAML2NameIdentifier",
    "devicePrintProfilesContainer",
    "inetorgperson",
    "sunIdentityServerLibertyPPService",
    "iPlanetPreferences",
    "pushDeviceProfilesContainer",
    "iplanet-am-user-service",
    "forgerock-am-dashboard-service",
    "organizationalperson",
    "top",
    "kbaInfoContainer",
    "sunAMAuthAccountLockout",
    "person",
    "oathDeviceProfilesContainer",
    "iplanet-am-auth-configuration-service"
  ],
  "dn":[
    "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
  ],
  "inetUserStatus":[
    "Active"
  ],
  "sn":[
    "demo"
  ],
  "cn":[
    "demo"
  ]
}
```

```

    },
    "modifyTimestamp": [
      "20170110105705Z"
    ],
    "createTimestamp": [
      "20170105101638Z"
    ]
  }
}

```

As shown in the example, AM returns the JSON representation of the profile on success. On failure, AM returns a JSON representation of the error including the HTTP status code. See also "HTTP Status Codes" in the *Getting Started with REST* for more information.

Creating Groups

AM lets administrators create a group with an HTTP POST of the JSON representation of the group to the `/json/subrealm/groups?_action=create` endpoint.

The following example shows how to create a group called `newGroup` in the top level realm using the REST API after authenticating to AM:

```

$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
--data '{
  "username": "newGroup"
}' \
https://openam.example.com:8443/openam/json/realms/root/groups?_action=create
{
  "username": "newGroup",
  "realm": "/",
  "uniqueMember": [
    "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
  ],
  "cn": [
    "newGroup"
  ],
  "dn": [
    "cn=newGroup,ou=groups,dc=openam,dc=forgerock,dc=org"
  ],
  "objectclass": [
    "groupofuniquenames",
    "top"
  ],
  "universalid": [
    "id=newGroup,ou=group,dc=openam,dc=forgerock,dc=org"
  ]
}

```


Adding a User to a Group

AM lets administrators add a user to an existing group with an HTTP PUT to the JSON representation of the group to the `/json/subrealm/groups/groupName` endpoint.

The following example shows how to add users to an existing group by using the REST API. The example assumes that the DS backend is in use. Make sure to use the `uniquemember` attribute to specify the user using the DS server:

```
$ curl \
--request PUT \
--header "iPlanetDirectoryPro: AQIC5...Y3MTAx*" \
--header "Content-Type: application/json" \
--data '{
  "uniquemember": [
    "uid=newUser,ou=people,dc=openam,dc=forgerock,dc=org",
    "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
  ]
}' \
https://openam.example.com:8443/openam/json/realms/root/groups/newGroup
{
  "name": "newGroup",
  "realm": "/",
  "uniqueMember": [
    "uid=newUser,ou=people,dc=openam,dc=forgerock,dc=org",
    "uid=demo,ou=people,dc=openam,dc=forgerock,dc=org"
  ],
  "cn": [
    "newGroup"
  ],
  "dn": [
    "cn=newGroup,ou=groups,dc=openam,dc=forgerock,dc=org"
  ],
  "objectclass": [
    "groupofuniquenames",
    "top"
  ],
  "universalid": [
    "id=newGroup,ou=group,dc=openam,dc=forgerock,dc=org"
  ]
}
```

Note

For Active Directory implementations, use the `member` attribute when adding a user to a group using the REST API:

```
$ curl \
--request PUT \
--header "iPlanetDirectoryPro: AQIC5...Y3MTAx*" \
--header "Content-Type: application/json" \
--data '{
  "member": [
    "cn=newUser,cn=users,dc=forgerock,dc=org",
    "cn=demo,cn=users,dc=forgerock,dc=org"
  ]
}' \
https://openam.example.com:8443/openam/json/realms/root/groups/newGroup
{
  "username": "newGroup",
  "realm": "/",
  "sAMAccountName": ["$FL2000-EP4RN8LPBKUS"],
  "universalid": ["id=newGroup,ou=group,dc=forgerock,dc=org"],
  "sAMAccountType": ["268435456"],
  "member": ["cn=newUser,cn=users,dc=forgerock,dc=org", "cn=demo,cn=users,dc=forgerock,dc=org"],
  "name": ["newGroup"],
  "objectClass": [
    "top",
    "group"
  ],
  "distinguishedName": ["CN=newGroup,CN=Users,DC=forgerock,DC=org"],
  "dn": ["CN=newGroup,CN=Users,DC=forgerock,DC=org"],
  "cn": ["newGroup"],
  "objectCategory": ["CN=Group,CN=Schema,CN=Configuration,DC=forgerock,DC=org"]
}
```

Realm Management

This section shows how to create, read, update, and delete realms using the [/json/global-config/realms](#) endpoint.

Tip

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

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

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

- "Required Properties for Managing Realms"
- "Creating Realms"
- "Listing Realms"
- "Reading Realms"
- "Updating Realms"

- "Deleting Realms"

Required Properties for Managing Realms

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

Realm Properties for JSON-based API

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

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

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

Creating Realms

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

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

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
--header "Accept-API-Version: resource=1.0" \
--data '{
  "name": "mySubRealm",
  "active": true,
  "parentPath": "/",
  "aliases": [ "payroll.example.com" ]
}' \
https://openam.example.com:8443/openam/json/global-config/realms
{
  "_id": "L2Fub3RoZXJSZWFSbQ",
  "_rev": "-1054013208",
  "parentPath": "/",
  "active": true,
  "name": "mySubRealm",
  "aliases": [ "payroll.example.com" ]
}
```

AM returns a 201 HTTP status code and a JSON representation of the realm on success. The value returned in the `_id` field is used in subsequent REST calls relating to the realm. On failure, AM returns a JSON representation of the error including the HTTP status code. For more information, see "HTTP Status Codes" in the *Getting Started with REST*.

Listing Realms

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

```
$ curl \
--header "iPlanetDirectoryPro: AQIC5..." \
--header "Accept-API-Version: resource=1.0, protocol=2.1" \
https://openam.example.com:8443/openam/json/global-config/realms?_queryFilter=true
{
  "result":[
    {
      "_id": "Lw",
      "_rev": "252584985",
      "parentPath": null,
      "active": true,
      "name": "/",
      "aliases": [
        "openam.example.com",
        "openam"
      ]
    },
    {
      "_id": "L215U3ViUmVhbG0",
      "_rev": "949061198",
      "parentPath": "/",
      "active": true,
      "name": "mySubRealm",
      "aliases": [

```

```
        "payroll.example.com"
    ]
  }
},
"resultCount":2,
"pagedResultsCookie":null,
"totalPagedResultsPolicy":"NONE",
"totalPagedResults":-1,
"remainingPagedResults":-1
}
```

For more information about using the `_queryString` parameter in REST calls, see "Query" in the *Getting Started with REST*.

Reading Realms

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

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

```
$ curl \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
--header "Accept-API-Version: resource=1.0" \
https://openam.example.com:8443/openam/json/global-config/realms/L215U3ViUmVhbG0
{
  "_id": "L215U3ViUmVhbG0",
  "_rev": "949061698",
  "parentPath": "/",
  "active": true,
  "name": "mySubRealm",
  "aliases": [
    "payroll.example.com"
  ]
}
```

AM returns a 200 HTTP status code and a JSON representation of the realm on success. On failure, AM returns a JSON representation of the error including the HTTP status code. For more information, see "HTTP Status Codes" in the *Getting Started with REST*.

Updating Realms

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

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

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

```
$ curl \
--request PUT \
--header "iplanetDirectoryPro: AQIC5...Y3MTax*" \
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0, protocol=1.0" \
--data '{
  "parentPath": "/",
  "active": false,
  "name": "mySubRealm",
  "aliases": [
    "payroll.example.com"
  ]
}' \
https://openam.example.com:8443/openam/json/global-config/realms/L215U3ViUmVhbG0
{
  "_id": "L215U3ViUmVhbG0",
  "_rev": "94906213",
  "parentPath": "/",
  "active": false,
  "name": "mySubRealm",
  "aliases": [
    "payroll.example.com"
  ]
}
```

AM returns a 200 HTTP status code and a JSON representation of the realm on success. On failure, AM returns a JSON representation of the error including the HTTP status code. For more information, see "HTTP Status Codes" in the *Getting Started with REST*.

Deleting Realms

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

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

Caution

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

```
$ curl \
--request DELETE \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
--header "Accept-API-Version: resource=1.0" \
https://openam.example.com:8443/openam/json/global-config/realms/L215U3ViUmVhbG0
{
  "_id": "L215U3ViUmVhbG0",
  "_rev": "949061708",
  "parentPath": "/",
  "active": false,
  "name": "mySubRealm",
  "aliases": [
    "payroll.example.com"
  ]
}
```

AM returns a 200 HTTP status code and a JSON representation of the deleted realm on success. On failure, AM returns a JSON representation of the error including the HTTP status code. For more information, "[HTTP Status Codes](#)" in the *Getting Started with REST*.

Chapter 3

Identity Stores

An identity store, or an identity repository, is a persistent repository of user data. For example, DS or Microsoft Active Directory. You can configure identity stores either when installing AM, or by adding them to an existing AM instance.

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

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

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

Note

You should not configure more than one writeable identity repository in a single realm. AM will try to perform write operations on each identity repository configured in a realm, and there is no way to configure which repository is written to.

To manage identities and reconcile differences between multiple identity repositories, use ForgeRock Identity Management.

Tasks to Connect Identity Stores

Task	Resources
Prepare an Identity Store You must prepare the identity store before AM can use it.	"Preparing Identity Repositories" in the <i>Installation Guide</i>
Configure an Identity Store Configure the store in a realm so that users can be authenticated. By default, AM re-uses your configuration store as the identity store of the Top Level realm.	"To Configure an Identity Store"
Customize an Identity Store Create custom attributes for your users or custom identity plugins to change how AM maps users and groups to a realm.	"Customizing Identity Stores".

To Configure an Identity Store

1. Share the identity store certificate with the AM container to prepare for TLS/LDAPS. Identity stores should communicate over secure connections for security reasons.

DS 7 or later is configured to require secure connections by default; therefore, share its certificate with the AM container before continuing.

+ *Sharing the DS Certificate with AM*

1. Export the DS server certificate:

```
$ /path/to/openssl/bin/dskeymgr export-ca-cert \
--deploymentKey $DEPLOYMENT_KEY \
--deploymentKeyPassword password \
--alias ds-ca-cert \
--outputFile ds-ca-cert.pem
```

Note that `$DEPLOYMENT_KEY` is a Unix variable that contains the DS deployment key, so that it is not logged in the user's command history.

The default DS server certificate only has the hostname you supplied at setup time, and `localhost`, as the value of the `SubjectAlternativeName` attribute; however, certificate hostname validation is strict.

Ensure that the certificate matches the hostname (or the FQDN) of the DS server before continuing.

2. Import the DS certificate into the AM truststore:

```
$ keytool \
-importcert \
-alias ds-ca-cert \
-file ds-ca-cert.pem \
-keystore /path/to/openam/security/keystores/truststore
```

For more information on configuring AM's truststore, see "Preparing a Truststore" in the *Installation Guide*.

2. In the AM console, browse to Realms > *Realm Name* > Identity Stores.
3. Click Add Identity Store, enter an ID, and select the type of identity store.
4. Click Create.
5. In the tabbed view, provide information on how to connect to your identity store.

See the following for hints depending on the type of identity store:

+ *Active Directory Configuration Properties*

Use these attributes when configuring Active Directory data stores:

amster service name: `IdRepository`

ssoadm service name: `sunIdentityRepositoryService`

ID

The ID of the data store configuration

LDAP Server

`host:port` to contact the directory server, with optional `|server_ID|site_ID` for deployments with multiple servers and sites.

AM uses the optional settings to determine which directory server to contact first. AM 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 AM server.
2. The first directory server in the list whose `site_ID` matches the current AM server.
3. The first directory server in the remaining list.

If the directory server is not available, AM proceeds to the next directory server in the list.

XUI

Default: `host:port` of the initial directory server configured for this AM server.

ssoadm

ssoadm attribute: `sun-idrepo-ldapv3-config-ldap-server`

Default: `=[0]=host:port` of the initial directory server configured for this AM server.

You must add `=n=` before the `host:port`, where *n* is an array index, starting with 0, of servers listed. See the example below.

Example:

```
sun-idrepo-ldapv3-config-ldap-server=[0]=localhost:51636|01  
sun-idrepo-ldapv3-config-ldap-server=[1]=openam.example.com:52389|02  
sun-idrepo-ldapv3-config-ldap-server=[2]=zzz.example.com:1636|01|02  
sun-idrepo-ldapv3-config-ldap-server=[3]=xxx.example.com:1636|01|02
```

LDAP Bind DN

Bind DN of the service account AM uses to connect to the directory server. Some AM 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.

Ensure that the identity store is setup with the specified DN before making any changes to this property in AM.

ssoadm attribute: `sun-idrepo-ldapv3-config-organization_name`

Default: `base-dn`

LDAP Connection Mode

Whether to use LDAP, LDAPS or StartTLS to connect to the directory server. When LDAPS or StartTLS are enabled, AM 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 AM runs, or because you imported the certificates into the trust store.

ssoadm attribute: `sun-idrepo-ldapv3-config-connection-mode`

Possible values: `LDAP`, `LDAPS`, and `StartTLS`

LDAP Connection Pool Maximum Size

Maximum number of connections to the directory server. Make sure the directory service can cope with the maximum number of client connections across all servers.

ssoadm attribute: `sun-idrepo-ldapv3-config-connection_pool_max_size`

Default: 10

LDAP Connection Heartbeat Interval

How often to send a heartbeat request to the directory server to ensure that the connection does not remain idle. Some network administrators configure firewalls and load balancers to drop connections that are idle for too long. You can turn this off by setting the value to 0 or to a negative number. To set the units for the interval use LDAP Connection Heartbeat Time Unit.

ssoadm attribute: `openam-idrepo-ldapv3-heartbeat-interval`

Default: 10

LDAP Connection Heartbeat Time Unit

Time unit for the LDAP Connection Heartbeat Interval setting.

ssoadm attribute: `openam-idrepo-ldapv3-heartbeat-timeunit`

Default: `SECONDS`

Maximum Results Returned from Search

A cap for the number of search results to return, for example, when viewing profiles under Identities. Rather than raise this number, consider narrowing your search to match fewer directory entries.

ssoadm attribute: `sun-idrepo-ldapv3-config-max-result`

Default: 1000

Search Timeout

Maximum time to wait for search results in seconds. Does not apply to persistent searches.

ssoadm attribute: `sun-idrepo-ldapv3-config-time-limit`

Default: 10

LDAPv3 Plugin Search Scope

LDAP searches can apply to a single entry (SCOPE_BASE), entries directly below the search DN (SCOPE_ONE), or all entries below the search DN (SEARCH_SUB)

ssoadm attribute: `sun-idrepo-ldapv3-config-search-scope`

Default: `SCOPE_SUB`

LDAPv3 Repository Plugin Class Name

AM identity repository implementation.

ssoadm attribute: `sunIdRepoClass`

Default: `org.forgerock.openam.idrepo.ldap.DJLDAPv3Repo`

Attribute Name Mapping

Map of AM profile attribute names to directory server attribute names.

ssoadm attribute: `sunIdRepoAttributeMapping`

Default: `userPassword=unicodePwd`

LDAPv3 Plugin Supported Types and Operations

Specifies the identity types supported by the data store, such as `user`, `group`, or `realm`, and which operations can be performed on them.

The following table illustrates the identity types supported by this data store, and the operations that can be performed on them:

Supported Identity Types and Operations

	read	create	edit	delete	service
group	✓	✓	✓	✓	-
realm	✓	✓	✓	✓	✓
user	✓	✓	✓	✓	✓
	Read the identity type	Create new identities of the given identity type	Edit entities of the given identity type	Delete entities of the given identity type	Read and write service settings associated with the given identity type.

You can remove permissions based on your data store needs. For example, if the data store should not be written to, you can set the operations to `read` only for the identity types.

The `service` operation is only relevant to the `realm` and the `user` identity types. For example, the Session Service configuration can be stored by realm, and a user can have specific session timeout settings.

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`

Warning

Do not modify the value of the search attribute in user profiles. Modifying this attribute value can result in incorrectly cached identity data. For example, if you configure the search attribute to `mail`, it could prevent users from being able to update their email addresses in their user profiles.

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.

If specified, AM will limit searches for user profiles to the provided base DN. Otherwise, AM searches the entire directory.

ssoadm attribute: `sun-idrepo-ldapv3-config-people-container-value`

Default: `users`

LDAP User Object Class

User profiles have these LDAP object classes.

AM handles only those attributes listed in this setting. AM discards any such unlisted attributes from requests and the request proceeds without the attribute.

For example, with default settings, if you request that AM execute a search that asks for the `mailAlternateAddress` attribute, AM does the search, but does not request `mailAlternateAddress`. In the same way, AM does perform an update operation with a request to set the value of an unlisted attribute like `mailAlternateAddress`, but it drops the unlisted attribute from the update request.

ssoadm attribute: `sun-idrepo-ldapv3-config-user-objectclass`

Default: `organizationalPerson, person, top, User,`

LDAP User Attributes

User profiles have these LDAP attributes.

ssoadm attribute: `sun-idrepo-ldapv3-config-user-attributes`

Default: `assignedDashboard, cn, createTimestamp, devicePrintProfiles, displayName, distinguishedName, dn, employeeNumber, givenName, iplanet-am-auth-configuration, iplanet-am-session-destroy-sessions, iplanet-am-session-get-valid-sessions, iplanet-am-session-max-caching-time, iplanet-am-session-max-idle-time, iplanet-am-session-max-session-time, iplanet-am-session-quota-limit, iplanet-am-session-service-status, iplanet-am-user-account-life, iplanet-am-user-admin-start-dn, iplanet-am-user-alias-list, iplanet-am-user-auth-config, iplanet-am-user-auth-modules, iplanet-am-user-failure-url, iplanet-am-user-federation-info, iplanet-am-user-federation-info-key, iplanet-am-user-login-status, iplanet-am-user-password-reset-force-reset, iplanet-am-user-password-reset-options, iplanet-am-user-password-reset-question-answer, iplanet-am-user-success-url, kbaActiveIndex, kbaInfo, mail, modifyTimestamp, name, oath2faEnabled, oathDeviceProfiles, objectGUID, objectclass, postalAddress, preferredLocale, preferredlanguage, preferredtimezone, pushDeviceProfiles, sAMAccountName, sn, sun-fm-saml2-nameid-info, sun-fm-saml2-nameid-infokey, sunAMAuthInvalidAttemptsData, sunIdentityMSISDNNumber, sunIdentityServerDiscoEntries, sunIdentityServerPPAddressCard, sunIdentityServerPPCommonNameAltCN, sunIdentityServerPPCommonNameCN, sunIdentityServerPPCommonNameFN, sunIdentityServerPPCommonNameMN, sunIdentityServerPPCommonNamePT, sunIdentityServerPPCommonNameSN, sunIdentityServerPPDemographicsAge, sunIdentityServerPPDemographicsBirthDay, sunIdentityServerPPDemographicsDisplayLanguage, sunIdentityServerPPDemographicsLanguage, sunIdentityServerPPDemographicsTimeZone, sunIdentityServerPPEmergencyContact, sunIdentityServerPPEmploymentIdentityAlt0, sunIdentityServerPPEmploymentIdentityJobTitle, sunIdentityServerPPEmploymentIdentityOrg, sunIdentityServerPPEncryPTKey, sunIdentityServerPPFacadeGreetSound, sunIdentityServerPPFacadeMugShot, sunIdentityServerPPFacadeNamePronounced, sunIdentityServerPPFacadeWebSite, sunIdentityServerPPFacadeGreetmesound, sunIdentityServerPPInformalName, sunIdentityServerPPLegalIdentityAltIdType, sunIdentityServerPPLegalIdentityAltIdValue, sunIdentityServerPPLegalIdentityDOB, sunIdentityServerPPLegalIdentityGender, sunIdentityServerPPLegalIdentityLegalName, sunIdentityServerPPLegalIdentityMaritalStatus,`

`sunIdentityServerPPLegalIdentityVATIdType`, `sunIdentityServerPPLegalIdentityVATIdValue`,
`sunIdentityServerPPMsgContact`, `sunIdentityServerPPSignKey`, `telephoneNumber`, `unicodePwd`,
`userAccountControl`, `userPrincipalname`, `userpassword`

Create User Attribute Mapping

When creating a user profile, apply this map of AM profile attribute names to directory server attribute names.

The LDAP user profile entries require the Common Name (`cn`) and Surname (`sn`) attributes, so that LDAP constraint violations do not occur when performing an add operation.

The `cn` attribute gets its value from the `uid` attribute, which comes from the User Name field on the AM console's login page. The `sn` attribute gets the value of the `givenName` attribute. Attributes not mapped to another attribute and attributes mapped to themselves (for example, `cn=cn`) take the value of the username unless the attribute values are provided when creating the profile.

ssoadm attribute: `sun-idrepo-ldapv3-config-createuser-attr-mapping`

Default: `cn`, `sn`

Attribute Name of User Status

Attribute to check/set user status.

ssoadm attribute: `sun-idrepo-ldapv3-config-isactive`

Default: `userAccountControl`

User Status Active Value

Active users have the user status attribute set to this value.

ssoadm attribute: `sun-idrepo-ldapv3-config-active`

Default: 544

User Status Inactive Value

Inactive users have the user status attribute set to this value.

ssoadm attribute: `sun-idrepo-ldapv3-config-inactive`

Default: 546

Authentication Naming Attribute

RDN attribute for building the bind DN when given a username and password to authenticate a user against the directory server.

ssoadm attribute: `sun-idrepo-ldapv3-config-auth-naming-attr`

Default: `cn`

LDAP Groups Search Attribute

When searching for a group by name, match values against this attribute.

ssoadm attribute: `sun-idrepo-ldapv3-config-groups-search-attribute`

Default: `cn`

LDAP Groups Search Filter

When searching for groups, apply this LDAP search filter as well.

ssoadm attribute: `sun-idrepo-ldapv3-config-groups-search-filter`

Default: `(objectclass=group)`

LDAP Groups Container Naming Attribute

RDN attribute of the LDAP base DN which contains group profiles.

ssoadm attribute: `sun-idrepo-ldapv3-config-group-container-name`

Default: `cn`

LDAP Groups Container Value

RDN attribute value of the LDAP base DN which contains group profiles.

If specified, AM will limit searches for group profiles to the provided base DN. Otherwise, AM searches the entire directory.

ssoadm attribute: `sun-idrepo-ldapv3-config-group-container-value`

Default: `users`

LDAP Groups Object Class

Group profiles have these LDAP object classes.

ssoadm attribute: `sun-idrepo-ldapv3-config-group-objectclass`

Default: `Group, top`

LDAP Groups Attributes

Group profiles have these LDAP attributes.

ssoadm attribute: `sun-idrepo-ldapv3-config-group-attributes`

Default: `cn, distinguishedName, dn, member, name, objectCategory, objectclass, sAMAccountName, sAMAccountType`

Attribute Name for Group Membership

LDAP attribute in the member's LDAP entry whose values are the groups to which a member belongs.

ssoadm attribute: `sun-idrepo-ldapv3-config-memberof`

Attribute Name of Unique Member

Attribute in the group's LDAP entry whose values are the members of the group.

ssoadm attribute: `sun-idrepo-ldapv3-config-uniquemember`

Default: `member`

Persistent Search Base DN

Base DN for LDAP-persistent searches used to receive notification of changes in directory server data.

ssoadm attribute: `sun-idrepo-ldapv3-config-psearchbase`

Default: `base-dn`

Persistent Search Scope

LDAP searches can apply to a single entry (SCOPE_BASE), entries directly below the search DN (SCOPE_ONE), or all entries below the search DN (SEARCH_SUB).

Specify either `SCOPE_BASE` or `SCOPE_ONE`. Do not specify `SCOPE_SUB`, as it can have a severe impact on Active Directory performance.

ssoadm attribute: `sun-idrepo-ldapv3-config-psearch-scope`

Default: `SCOPE_SUB`

The Delay Time Between Retries

How long to wait after receiving an error result that indicates AM 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 AM uses persistent searches to obtain notification of such updates.

ssoadm attribute: `sun-idrepo-ldapv3-dncache-enabled`

Default: false

DN Cache Size

Maximum number of DN's cached when caching is enabled.

ssoadm attribute: `sun-idrepo-ldapv3-dncache-size`

Default: 1500 items

Load Schema

Import appropriate LDAP schema to the directory server before saving the configuration. The LDAP Bind DN service account must have access to perform this operation.

This option is not available for use with the **ssoadm** command.

Default: Disabled.

+ Active Directory Application Mode Configuration Properties

Use these attributes when configuring Active Directory Application Mode (ADAM) Identity Stores:

amster service name: `IdRepository`

ssoadm service name: `sunIdentityRepositoryService`

ID

The ID of the data store configuration.

LDAP Server

`host:port` to contact the directory server, with optional `|server_ID|site_ID` for deployments with multiple servers and sites.

You must add `=n` before the `host:port`, where *n* is an array index, starting with 0, of servers listed. See the example below.

AM uses the optional settings to determine which directory server to contact first. AM 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 AM server.
2. The first directory server in the list whose `site_ID` matches the current AM server.
3. The first directory server in the remaining list.

If the directory server is not available, AM proceeds to the next directory server in the list.

XUI

Default: `host:port` of the initial directory server configured for this AM server.

ssoadm

ssoadm attribute: `sun-idrepo-ldapv3-config-ldap-server`

Default: `=0`=`host:port` of the initial directory server configured for this AM server.

You must add `=n` before the `host:port`, where *n* is an array index, starting with 0, of servers listed. See the example below.

Example:

```
sun-idrepo-ldapv3-config-ldap-server=0=localhost:51636|01
sun-idrepo-ldapv3-config-ldap-server=1=openam.example.com:52389|02
sun-idrepo-ldapv3-config-ldap-server=2=zzz.example.com:1636|01|02
sun-idrepo-ldapv3-config-ldap-server=3=xxx.example.com:1636|01|02
```

LDAP Bind DN

Bind DN of the service account AM uses to connect to the directory server. Some AM 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.

Ensure that the identity store is setup with the specified DN before making any changes to this property in AM.

ssoadm attribute: `sun-idrepo-ldapv3-config-organization_name`

Default: `base-dn`

LDAP Connection Mode

Whether to use LDAP, LDAPS or StartTLS to connect to the directory server. When LDAPS or StartTLS are enabled, AM 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 AM runs, or because you imported the certificates into the trust store.

ssoadm attribute: `sun-idrepo-ldapv3-config-connection-mode`

Possible values: `LDAP`, `LDAPS`, and `StartTLS`

LDAP Connection Pool Maximum Size

Maximum number of connections to the directory server. Make sure the directory service can cope with the maximum number of client connections across all servers.

ssoadm attribute: `sun-idrepo-ldapv3-config-connection_pool_max_size`

Default: 10

LDAP Connection Heartbeat Interval

How often to send a heartbeat request to the directory server to ensure that the connection does not remain idle. Some network administrators configure firewalls and load balancers to drop connections that are idle for too long. You can turn this off by setting the value to 0 or to a negative number. To set the units for the interval, use LDAP Connection Heartbeat Time Unit.

ssoadm attribute: `openam-idrepo-ldapv3-heartbeat-interval`

Default: 10

LDAP Connection Heartbeat Time Unit

Time unit for the LDAP Connection Heartbeat Interval setting

ssoadm attribute: `openam-idrepo-ldapv3-heartbeat-timeunit`

Default: `second`

Maximum Results Returned from Search

A cap for the number of search results to return, for example, when viewing profiles under Identities. Rather than raise this number, consider narrowing your search to match fewer directory entries.

ssoadm attribute: `sun-idrepo-ldapv3-config-max-result`

Default: 1000

Search Timeout

Maximum time to wait for search results in seconds. Does not apply to persistent searches.

ssoadm attribute: `sun-idrepo-ldapv3-config-time-limit`

Default: 10

LDAPv3 Plugin Search Scope

LDAP searches can apply to a single entry (SCOPE_BASE), entries directly below the search DN (SCOPE_ONE), or all entries below the search DN (SEARCH_SUB).

ssoadm attribute: `sun-idrepo-ldapv3-config-search-scope`

Default: `SCOPE_SUB`

LDAPv3 Repository Plugin Class Name

AM identity repository implementation.

ssoadm attribute: `sunIdRepoClass`

Default: `org.forgerock.openam.idrepo.ldap.DJLDAPv3Repo`

Attribute Name Mapping

Map of AM profile attribute names to directory server attribute names.

ssoadm attribute: `sunIdRepoAttributeMapping`

Default: `userPassword=unicodePwd`

LDAPv3 Plugin Supported Types and Operations

Specifies the identity types supported by the data store, such as `user`, `group`, or `realm`, and which operations can be performed on them.

The following table illustrates the identity types supported by this data store, and the operations that can be performed on them:

Supported Identity Types and Operations

	read	create	edit	delete	service
group	✓	✓	✓	✓	-
realm	✓	✓	✓	✓	✓
user	✓	✓	✓	✓	✓
	Read the identity type	Create new identities of the given identity type	Edit entities of the given identity type	Delete entities of the given identity type	Read and write service settings associated with the given identity type.

You can remove permissions based on your data store needs. For example, if the data store should not be written to, you can set the operations to **read** only for the identity types.

The **service** operation is only relevant to the **realm** and the **user** identity types. For example, the Session Service configuration can be stored by realm, and a user can have specific session timeout settings.

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`

Warning

Do not modify the value of the search attribute in user profiles. Modifying this attribute value can result in incorrectly cached identity data. For example, if you configure the search attribute to `mail`, it could prevent users from being able to update their email addresses in their user profiles.

LDAP Users Search Filter

When searching for users, apply this LDAP search filter as well.

ssoadm attribute: `sun-idrepo-ldapv3-config-users-search-filter`

Default: `(objectclass=person)`

LDAP People Container Naming Attribute

RDN attribute of the LDAP base DN which contains user profiles.

ssoadm attribute: `sun-idrepo-ldapv3-config-people-container-name`

LDAP People Container Value

RDN attribute value of the LDAP base DN which contains user profiles.

If specified, AM will limit searches for user profiles to the provided base DN. Otherwise, AM searches the entire directory.

ssoadm attribute: `sun-idrepo-ldapv3-config-people-container-value`

LDAP User Object Class

User profiles have these LDAP object classes.

AM handles only those attributes listed in this setting. AM discards any unlisted attributes from requests and the request proceeds without the attribute.

For example, with default settings, if you request that AM execute a search that asks for the `mailAlternateAddress` attribute, AM does the search, but does not request `mailAlternateAddress`. In the same way, AM does perform an update operation with a request to set the value of an unlisted attribute like `mailAlternateAddress`, but it drops the unlisted attribute from the update request.

ssoadm attribute: `sun-idrepo-ldapv3-config-user-objectclass`

Default: `devicePrintProfilesContainer, forgerock-am-dashboard-service, iPlanetPreferences, iplanet-am-auth-configuration-service, iplanet-am-managed-person, iplanet-am-user-service, kbaInfoContainer, oathDeviceProfilesContainer, organizationalPerson, person, pushDeviceProfilesContainer, sunAMAuthAccountLockout, sunFMSAML2NameIdentifier, sunFederationManagerDataStore, sunIdentityServerLibertyPPService, top, User`

LDAP User Attributes

User profiles have these LDAP attributes.

AM handles only those attributes listed in this setting. AM discards any unlisted attributes from requests and the request proceeds without the attribute.

ssoadm attribute: `sun-idrepo-ldapv3-config-user-attributes`

Default: `assignedDashboard, cn, createTimeStamp, devicePrintProfiles, displayName, distinguishedName, dn, employeeNumber, givenName, iplanet-am-auth-configuration, iplanet-am-session-destroy-sessions, iplanet-am-session-get-valid-sessions, iplanet-am-session-max-caching-time, iplanet-am-session-max-idle-time, iplanet-am-session-max-session-time, iplanet-am-session-quota-limit, iplanet-am-session-service-status, iplanet-am-user-account-life, iplanet-am-user-admin-start-dn, iplanet-am-user-alias-list, iplanet-am-user-auth-config, iplanet-am-user-auth-modules, iplanet-am-user-failure-url, iplanet-am-user-federation-info, iplanet-am-user-federation-info-key, iplanet-am-user-login-status, iplanet-am-user-password-reset-force-reset, iplanet-am-user-password-reset-options, iplanet-am-user-password-reset-question-answer, iplanet-am-user-success-url, kbaActiveIndex, kbaInfo, mail, modifyTimeStamp, msDS-UserAccountDisabled, name, oath2faEnabled, oathDeviceProfiles, objectGUID, objectclass, postalAddress, preferredLocale, preferredlanguage, preferredtimezone, pushDeviceProfiles, sn, sun-fm-saml2-nameid-info, sun-fm-saml2-nameid-infokey, sunAMAuthInvalidAttemptsData, sunIdentityMSISDNNumber, sunIdentityServerDiscoEntries, sunIdentityServerPPAddressCard, sunIdentityServerPPCommonNameAltCN, sunIdentityServerPPCommonNameCN, sunIdentityServerPPCommonNameFN, sunIdentityServerPPCommonNameMN, sunIdentityServerPPCommonNamePT, sunIdentityServerPPCommonNameSN, sunIdentityServerPPDemographicsAge, sunIdentityServerPPDemographicsBirthDay, sunIdentityServerPPDemographicsDisplayLanguage, sunIdentityServerPPDemographicsLanguage, sunIdentityServerPPDemographicsTimeZone, sunIdentityServerPPEmergencyContact, sunIdentityServerPPEmploymentIdentityAlt0, sunIdentityServerPPEmploymentIdentityJobTitle, sunIdentityServerPPEmploymentIdentityOrg, sunIdentityServerPPEncryPTKey, sunIdentityServerPPFacadeGreetSound, sunIdentityServerPPFacadeMugShot, sunIdentityServerPPFacadeNamePronounced, sunIdentityServerPPFacadeWebSite, sunIdentityServerPPFacadeGreetmesound, sunIdentityServerPPInformalName, sunIdentityServerPPLegalIdentityAltIdType, sunIdentityServerPPLegalIdentityAltIdValue, sunIdentityServerPPLegalIdentityDOB, sunIdentityServerPPLegalIdentityGender, sunIdentityServerPPLegalIdentityLegalName, sunIdentityServerPPLegalIdentityMaritalStatus, sunIdentityServerPPLegalIdentityVATIdType, sunIdentityServerPPLegalIdentityVATIdValue, sunIdentityServerPPMsgContact, sunIdentityServerPPSignKey, telephoneNumber, unicodePwd, userPrincipalname, userpassword`

Create User Attribute Mapping

When creating a user profile, apply this map of AM profile attribute names to directory server attribute names.

Attributes not mapped to another attribute (for example, `cn`) and attributes mapped to themselves, (for example, `cn=cn`) take the value of the username unless the attribute values are provided when creating the profile. The object classes for user profile LDAP entries generally require Common Name (`cn`) and Surname (`sn`) attributes, so this prevents an LDAP constraint violation when performing the add operation.

ssoadm attribute: `sun-idrepo-ldapv3-config-createuser-attr-mapping`

Default: `cn, sn`

Attribute Name of User Status

Attribute to check/set user status.

ssoadm attribute: `sun-idrepo-ldapv3-config-isactive`

Default: `msDS-UserAccountDisabled`

User Status Active Value

Active users have the user status attribute set to this value.

ssoadm attribute: `sun-idrepo-ldapv3-config-active`

Default: `FALSE`

User Status Inactive Value

Inactive users have the user status attribute set to this value.

ssoadm attribute: `sun-idrepo-ldapv3-config-inactive`

Default: `TRUE`

Authentication Naming Attribute

RDN attribute for building the bind DN when given a username and password to authenticate a user against the directory server.

ssoadm attribute: `sun-idrepo-ldapv3-config-auth-naming-attr`

Default: `cn`

LDAP Groups Search Attribute

When searching for a group by name, match values against this attribute.

ssoadm attribute: `sun-idrepo-ldapv3-config-groups-search-attribute`

Default: `cn`

LDAP Groups Search Filter

When searching for groups, apply this LDAP search filter as well.

ssoadm attribute: `sun-idrepo-ldapv3-config-groups-search-filter`

Default: `(objectclass=group)`

LDAP Groups Container Naming Attribute

RDN attribute of the LDAP base DN which contains group profiles.

ssoadm attribute: `sun-idrepo-ldapv3-config-group-container-name`

Default: `cn`

LDAP Groups Container Value

RDN attribute value of the LDAP base DN which contains group profiles.

If specified, AM will limit searches for group profiles to the provided base DN. Otherwise, AM searches the entire directory.

ssoadm attribute: `sun-idrepo-ldapv3-config-group-container-value`

LDAP Groups Object Class

Group profiles have these LDAP object classes.

ssoadm attribute: `sun-idrepo-ldapv3-config-group-objectclass`

Default: `Group, top`

LDAP Groups Attributes

Group profiles have these LDAP attributes.

ssoadm attribute: `sun-idrepo-ldapv3-config-group-attributes`

Default: `cn, distinguishedName, dn, member, name, objectCategory, objectclass, sAMAccountName, sAMAccountType`

Attribute Name for Group Membership

LDAP attribute in the member's LDAP entry whose values are the groups to which a member belongs.

ssoadm attribute: `sun-idrepo-ldapv3-config-memberof`

Attribute Name of Unique Member

Attribute in the group's LDAP entry whose values are the members of the group.

ssoadm attribute: `sun-idrepo-ldapv3-config-uniquemember`

Default: `member`

Persistent Search Base DN

Base DN for LDAP-persistent searches used to receive notification of changes in directory server data.

ssoadm attribute: `sun-idrepo-ldapv3-config-psearchbase`

Default: `base-dn`

Persistent Search Scope

LDAP searches can apply to a single entry (SCOPE_BASE), entries directly below the search DN (SCOPE_ONE), or all entries below the search DN (SEARCH_SUB).

Specify either `SCOPE_BASE` or `SCOPE_ONE`. Do not specify `SCOPE_SUB`, as it can have a severe impact on Active Directory performance.

ssoadm attribute: `sun-idrepo-ldapv3-config-psearch-scope`

Default: `SCOPE_SUB`

The Delay Time Between Retries

How long to wait after receiving an error result that indicates AM should try the LDAP operation again.

ssoadm attribute: `com.ipplanet.am.ldap.connection.delay.between.retries`

Default: 1000 milliseconds

DN Cache Enabled

Whether to enable the DN cache, which is used to cache DN lookups that can happen in bursts during authentication. As the cache can become stale when a user is moved or renamed, enable DN caching when the directory service allows move/rename operations (Mod DN), and when AM uses persistent searches to obtain notification of such updates.

ssoadm attribute: `sun-idrepo-ldapv3-dncache-enabled`

Default: false

DN Cache Size

Maximum number of DN's cached when caching is enabled.

ssoadm attribute: `sun-idrepo-ldapv3-dncache-size`

Default: 1500 items

Load Schema

Import appropriate LDAP schema to the directory server before saving the configuration. The LDAP Bind DN service account must have access to perform this operation.

This option is not available for use with the **ssoadm** command.

Default: Disabled.

+ Sun/Oracle DSEE Configuration Properties

Use these attributes when configuring data stores for Oracle DSEE or Sun DSEE using AM schema:

amster service name: `IdRepository`

ssoadm service name: `sunIdentityRepositoryService`

ID

The ID of the data store configuration.

LDAP Server

`host:port` to contact the directory server, with optional `|server_ID|site_ID` for deployments with multiple servers and sites.

You must add `=[n]=` before the `host:port`, where `n` is an array index, starting with 0, of servers listed. See the example below.

AM uses the optional settings to determine which directory server to contact first. AM 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 AM server.
2. The first directory server in the list whose `site_ID` matches the current AM server.
3. The first directory server in the remaining list.

If the directory server is not available, AM proceeds to the next directory server in the list.

XUI

Default: `host:port` of the initial directory server configured for this AM server.

ssoadm

ssoadm attribute: `sun-idrepo-ldapv3-config-ldap-server`

Default: `=[0]=host:port` of the initial directory server configured for this AM server.

You must add `=[n]=` before the `host:port`, where `n` is an array index, starting with 0, of servers listed. See the example below.

Example:

```
sun-idrepo-ldapv3-config-ldap-server=[0]=localhost:51636|01
sun-idrepo-ldapv3-config-ldap-server=[1]=openam.example.com:52389|02
sun-idrepo-ldapv3-config-ldap-server=[2]=zzz.example.com:1636|01|02
sun-idrepo-ldapv3-config-ldap-server=[3]=xxx.example.com:1636|01|02
```

LDAP Bind DN

Bind DN of the service account AM uses to connect to the directory server. Some AM 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.

Ensure that the identity store is setup with the specified DN before making any changes to this property in AM.

ssoadm attribute: `sun-idrepo-ldapv3-config-organization_name`

Default: `base-dn`

LDAP Connection Mode

Whether to use LDAP, LDAPS or StartTLS to connect to the directory server. When LDAPS or StartTLS are enabled, AM 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 AM runs, or because you imported the certificates into the trust store.

ssoadm attribute: `sun-idrepo-ldapv3-config-connection-mode`

Possible values: `LDAP`, `LDAPS`, and `StartTLS`

LDAP Connection Pool Maximum Size

Maximum number of connections to the directory server. Make sure the directory service can cope with the maximum number of client connections across all servers.

ssoadm attribute: `sun-idrepo-ldapv3-config-connection_pool_max_size`

Default: 10

LDAP Connection Heartbeat Interval

How often to send a heartbeat request to the directory server to ensure that the connection does not remain idle. Some network administrators configure firewalls and load balancers to drop connections that are idle for too long. You can turn this off by setting the value to 0 or to a negative number. To set the units for the interval, use LDAP Connection Heartbeat Time Unit.

ssoadm attribute: `openam-idrepo-ldapv3-heartbeat-interval`

Default: 10

LDAP Connection Heartbeat Time Unit

Time unit for the LDAP Connection Heartbeat Interval setting.

ssoadm attribute: `openam-idrepo-ldapv3-heartbeat-timeunit`

Default: `second`

Maximum Results Returned from Search

A cap for the number of search results to return, for example, when viewing profiles under Identities. Rather than raise this number, consider narrowing your search to match fewer directory entries.

ssoadm attribute: `sun-idrepo-ldapv3-config-max-result`

Default: 1000

Search Timeout

Maximum time to wait for search results in seconds. Does not apply to persistent searches.

ssoadm attribute: `sun-idrepo-ldapv3-config-time-limit`

Default: 10

LDAPv3 Plugin Search Scope

LDAP searches can apply to a single entry (SCOPE_BASE), entries directly below the search DN (SCOPE_ONE), or all entries below the search DN (SEARCH_SUB).

ssoadm attribute: `sun-idrepo-ldapv3-config-search-scope`

Default: `SCOPE_SUB`

LDAPv3 Repository Plugin Class Name

AM identity repository implementation.

ssoadm attribute: `sunIdRepoClass`

Default: `org.forgerock.openam.idrepo.ldap.DJLDAPv3Repo`

Attribute Name Mapping

Map of AM profile attribute names to directory server attribute names.

ssoadm attribute: `sunIdRepoAttributeMapping`

LDAPv3 Plugin Supported Types and Operations

Specifies the identity types supported by the data store, such as `user`, `group`, or `realm`, and which operations can be performed on them.

The following table illustrates the identity types supported by this data store, and the operations that can be performed on them:

Supported Identity Types and Operations

	read	create	edit	delete	service
group	✓	✓	✓	✓	-
realm	✓	✓	✓	✓	✓
user	✓	✓	✓	✓	✓

	read	create	edit	delete	service
role	✓	✓	✓	✓	-
filteredrole	✓	✓	✓	✓	-
	Read the identity type	Create new identities of the given identity type	Edit entities of the given identity type	Delete entities of the given identity type	Read and write service settings associated with the given identity type.

You can remove permissions based on your data store needs. For example, if the data store should not be written to, you can set the operations to **read** only for the identity types.

The **service** operation is only relevant to the **realm** and the **user** identity types. For example, the Session Service configuration can be stored by realm, and a user can have specific session timeout settings.

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**

Warning

Do not modify the value of the search attribute in user profiles. Modifying this attribute value can result in incorrectly cached identity data.

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.

If specified, AM will limit searches for user profiles to the provided base DN. Otherwise, AM searches the entire directory.

ssoadm attribute: `sun-idrepo-ldapv3-config-people-container-value`

Default: `people`

LDAP User Object Class

User profiles have these LDAP object classes.

AM handles only those attributes listed in this setting. AM discards any unlisted attributes from requests and the request proceeds without the attribute.

For example, with default settings, if you request that AM execute a search that asks for the `mailAlternateAddress` attribute, AM does the search, but does not request `mailAlternateAddress`. In the same way, AM does perform an update operation with a request to set the value of an unlisted attribute like `mailAlternateAddress`, but it drops the unlisted attribute from the update request.

ssoadm attribute: `sun-idrepo-ldapv3-config-user-objectclass`

Default: `devicePrintProfilesContainer, forgerock-am-dashboard-service, iPlanetPreferences, inetadmin, inetorgperson, inetuser, iplanet-am-auth-configuration-service, iplanet-am-managed-person, iplanet-am-user-service, kbaInfoContainer, oathDeviceProfilesContainer, organizationalperson, person, pushDeviceProfilesContainer, sunAMAuthAccountLockout, sunFMSAML2NameIdentifier, sunFederationManagerDataStore, sunIdentityServerLibertyPPService, top`

LDAP User Attributes

User profiles have these LDAP attributes.

AM handles only those attributes listed in this setting. AM discards any unlisted attributes from requests and the request proceeds without the attribute.

ssoadm attribute: `sun-idrepo-ldapv3-config-user-attributes`

Default: assignedDashboard, authorityRevocationList, caCertificate, cn, createTimestamp, devicePrintProfiles, distinguishedName, adminRole, dn, employeeNumber, givenName, inetUserHttpURL, inetUserStatus, iplanet-am-auth-configuration, iplanet-am-session-destroy-sessions, iplanet-am-session-get-valid-sessions, iplanet-am-session-max-caching-time, iplanet-am-session-max-idle-time, iplanet-am-session-max-session-time, iplanet-am-session-quota-limit, iplanet-am-session-service-status, iplanet-am-static-group-dn, iplanet-am-user-account-life, iplanet-am-user-admin-start-dn, iplanet-am-user-alias-list, iplanet-am-user-auth-config, iplanet-am-user-auth-modules, iplanet-am-user-failure-url, iplanet-am-user-federation-info, iplanet-am-user-federation-info-key, iplanet-am-user-login-status, iplanet-am-user-password-reset-force-reset, iplanet-am-user-password-reset-options, iplanet-am-user-password-reset-question-answer, iplanet-am-user-success-url, kbaActiveIndex, kbaInfo, mail, manager, memberOf, modifyTimestamp, oath2faEnabled, oathDeviceProfiles, objectClass, postalAddress, preferredLocale, preferredlanguage, preferredtimezone, pushDeviceProfiles, sn, sun-fm-saml2-nameid-info, sun-fm-saml2-nameid-infokey, sunAMAuthInvalidAttemptsData, sunIdentityMSISDNNumber, sunIdentityServerDiscoEntries, sunIdentityServerPPAddressCard, sunIdentityServerPPCommonNameAltCN, sunIdentityServerPPCommonNameCN, sunIdentityServerPPCommonNameFN, sunIdentityServerPPCommonNameMN, sunIdentityServerPPCommonNamePT, sunIdentityServerPPCommonNamesN, sunIdentityServerPPDemographicsAge, sunIdentityServerPPDemographicsBirthDay, sunIdentityServerPPDemographicsDisplayLanguage, sunIdentityServerPPDemographicsLanguage, sunIdentityServerPPDemographicsTimeZone, sunIdentityServerPPEmergencyContact, sunIdentityServerPPEmploymentIdentityAlt0, sunIdentityServerPPEmploymentIdentityJobTitle, sunIdentityServerPPEmploymentIdentityOrg, sunIdentityServerPPEncryPTKey, sunIdentityServerPPFacadeGreetSound, sunIdentityServerPPFacadeMugShot, sunIdentityServerPPFacadeNamePronounced, sunIdentityServerPPFacadeWebSite, sunIdentityServerPPFacadeGreetmesound, sunIdentityServerPPInformalName, sunIdentityServerPPLegalIdentityAltIdType, sunIdentityServerPPLegalIdentityAltIdValue, sunIdentityServerPPLegalIdentityDOB, sunIdentityServerPPLegalIdentityGender, sunIdentityServerPPLegalIdentityLegalName, sunIdentityServerPPLegalIdentityMaritalStatus, sunIdentityServerPPLegalIdentityVATIdType, sunIdentityServerPPLegalIdentityVATIdValue, sunIdentityServerPPMsgContact, sunIdentityServerPPSignKey, telephoneNumber, uid, userCertificate, userPassword

Create User Attribute Mapping

When creating a user profile, apply this map of AM 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.

If specified, AM will limit searches for group profiles to the provided base DN. Otherwise, AM searches the entire directory.

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, ipplanet-am-managed-group, ipplanet-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, ipplanet-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, nscomplexroledescription, nsfilteredroledescription, nsroledescription`

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 AM should try the LDAP operation again.

ssoadm attribute: `com.ipplanet.am.ldap.connection.delay.between.retries`

Default: 1000 milliseconds

DN Cache Enabled

Whether to enable the DN cache, which is used to cache DN lookups that can happen in bursts during authentication. As the cache can become stale when a user is moved or renamed, enable DN caching when the directory service allows move/rename operations (Mod DN), and when AM uses persistent searches to obtain notification of such updates.

ssoadm attribute: `sun-idrepo-ldapv3-dncache-enabled`

Default: true

DN Cache Size

Maximum number of DN's cached when caching is enabled.

ssoadm attribute: `sun-idrepo-ldapv3-dncache-size`

Default: 1500 items

Load Schema

Import appropriate LDAP schema to the directory server before saving the configuration. The LDAP Bind DN service account must have access to perform this operation.

This option is not available for use with the **ssoadm** command.

Default: Disabled.

+ Generic LDAPv3 Configuration Properties

Use these attributes when configuring Generic LDAPv3 compliant data stores:

amster service name: `IdRepository`

ssoadm service name: `sunIdentityRepositoryService`

ID

The ID of the data store configuration.

LDAP Server

`host:port` to contact the directory server, with optional `|server_ID|site_ID` for deployments with multiple servers and sites.

You must add `=[n]=` before the `host:port`, where `n` is an array index, starting with 0, of servers listed. See the example below.

AM uses the optional settings to determine which directory server to contact first. AM 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 AM server.
2. The first directory server in the list whose `site_ID` matches the current AM server.
3. The first directory server in the remaining list.

If the directory server is not available, AM proceeds to the next directory server in the list.

XUI

Default: `host:port` of the initial directory server configured for this AM server.

ssoadm

ssoadm attribute: `sun-idrepo-ldapv3-config-ldap-server`

Default: `=[0]=host:port` of the initial directory server configured for this AM server.

You must add `=[n]=` before the `host:port`, where `n` is an array index, starting with 0, of servers listed. See the example below.

Example:

```
sun-idrepo-ldapv3-config-ldap-server=[0]=localhost:51636|01  
sun-idrepo-ldapv3-config-ldap-server=[1]=openam.example.com:52389|02  
sun-idrepo-ldapv3-config-ldap-server=[2]=zzz.example.com:1636|01|02  
sun-idrepo-ldapv3-config-ldap-server=[3]=xxx.example.com:1636|01|02
```

LDAP Bind DN

Bind DN of the service account AM uses to connect to the directory server. Some AM 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.

Ensure that the identity store is setup with the specified DN before making any changes to this property in AM.

ssoadm attribute: `sun-idrepo-ldapv3-config-organization_name`

Default: `base-dn`

LDAP Connection Mode

Whether to use LDAP, LDAPS or StartTLS to connect to the directory server. When LDAPS or StartTLS are enabled, AM 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 AM runs, or because you imported the certificates into the trust store.

ssoadm attribute: `sun-idrepo-ldapv3-config-connection-mode`

Possible values: `LDAP`, `LDAPS`, and `StartTLS`

LDAP Connection Pool Maximum Size

Maximum number of connections to the directory server. Make sure the directory service can cope with the maximum number of client connections across all servers.

ssoadm attribute: `sun-idrepo-ldapv3-config-connection_pool_max_size`

Default: 10

LDAP Connection Heartbeat Interval

How often to send a heartbeat request to the directory server to ensure that the connection does not remain idle. Some network administrators configure firewalls and load balancers to drop connections that are idle for too long. You can turn this off by setting the value to 0 or to a negative number. To set the units for the interval, use LDAP Connection Heartbeat Time Unit.

ssoadm attribute: `openam-idrepo-ldapv3-heartbeat-interval`

Default: 10

LDAP Connection Heartbeat Time Unit

Time unit for the LDAP Connection Heartbeat Interval setting.

ssoadm attribute: `openam-idrepo-ldapv3-heartbeat-timeunit`

Default: `second`

Maximum Results Returned from Search

A cap for the number of search results to return, for example, when viewing profiles under Identities. Rather than raise this number, consider narrowing your search to match fewer directory entries.

ssoadm attribute: `sun-idrepo-ldapv3-config-max-result`

Default: 1000

Search Timeout

Maximum time to wait for search results in seconds. Does not apply to persistent searches.

ssoadm attribute: `sun-idrepo-ldapv3-config-time-limit`

Default: 10

LDAPv3 Plugin Search Scope

LDAP searches can apply to a single entry (SCOPE_BASE), entries directly below the search DN (SCOPE_ONE), or all entries below the search DN (SEARCH_SUB).

ssoadm attribute: `sun-idrepo-ldapv3-config-search-scope`

Default: `SCOPE_SUB`

LDAPv3 Repository Plugin Class Name

AM identity repository implementation.

ssoadm attribute: `sunIdRepoClass`

Default: `org.forgerock.openam.idrepo.ldap.DJLDAPv3Repo`

Attribute Name Mapping

Map of AM profile attribute names to directory server attribute names.

ssoadm attribute: `sunIdRepoAttributeMapping`

LDAPv3 Plugin Supported Types and Operations

Specifies the identity types supported by the data store, such as `user`, `group`, or `realm`, and which operations can be performed on them.

The following table illustrates the identity types supported by this data store, and the operations that can be performed on them:

Supported Identity Types and Operations

	read	create	edit	delete	service
group	✓	✓	✓	✓	-
realm	✓	✓	✓	✓	✓
user	✓	✓	✓	✓	✓
	Read the identity type	Create new identities of the given identity type	Edit entities of the given identity type	Delete entities of the given identity type	Read and write service settings associated with the given identity type.

You can remove permissions based on your data store needs. For example, if the data store should not be written to, you can set the operations to `read` only for the identity types.

The `service` operation is only relevant to the `realm` and the `user` identity types. For example, the Session Service configuration can be stored by realm, and a user can have specific session timeout settings.

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: `uid`

Warning

Do not modify the value of the search attribute in user profiles. Modifying this attribute value can result in incorrectly cached identity data. For example, if you configure the search attribute to `mail`, it could prevent users from being able to update their email addresses in their user profiles.

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.

If specified, AM will limit searches for user profiles to the provided base DN. Otherwise, AM searches the entire directory.

ssoadm attribute: `sun-idrepo-ldapv3-config-people-container-value`

LDAP User Object Class

User profiles have these LDAP object classes.

AM handles only those attributes listed in this setting. AM discards any unlisted attributes from requests and the request proceeds without the attribute.

For example, with default settings, if you request that AM execute a search that asks for the `mailAlternateAddress` attribute, AM does the search, but does not request `mailAlternateAddress`. In the same way, AM 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.

AM handles only those attributes listed in this setting. AM discards any unlisted attributes from requests and the request proceeds without the attribute.

ssoadm attribute: `sun-idrepo-ldapv3-config-user-attributes`

Default: `uid, caCertificate, authorityRevocationList, inetUserStatus, mail, sn, manager, userPassword, adminRole, objectClass, givenName, memberOf, cn, telephoneNumber, preferredLanguage, userCertificate, postalAddress, dn, employeeNumber, distinguishedName`

Create User Attribute Mapping

When creating a user profile, apply this map of AM 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.

If specified, AM will limit searches for group profiles to the provided base DN. Otherwise, AM searches the entire directory.

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 AM should try the LDAP operation again.

ssoadm attribute: `com.ipplanet.am.ldap.connection.delay.between.retries`

Default: 1000 milliseconds

DN Cache Enabled

Whether to enable the DN cache, which is used to cache DN lookups that can happen in bursts during authentication. As the cache can become stale when a user is moved or renamed, enable DN caching when the directory service allows move/rename operations (Mod DN), and when AM uses persistent searches to obtain notification of such updates.

ssoadm attribute: `sun-idrepo-ldapv3-dncache-enabled`

Default: false

DN Cache Size

Maximum number of DN's cached when caching is enabled.

ssoadm attribute: `sun-idrepo-ldapv3-dncache-size`

Default: 1500 items

Load Schema

Import appropriate LDAP schema to the directory server before saving the configuration. The LDAP Bind DN service account must have access to perform this operation.

This option is not available for use with the **ssoadm** command.

Default: Disabled.

+ Directory Services Configuration Properties

Use these attributes when configuring DS data stores:

amster service name: `IdRepository`

ssoadm service name: `sunIdentityRepositoryService`

The following property appears on every tab:

Load Schema

Import appropriate LDAP schema to the directory server when saving the configuration. The LDAP Bind DN service account must have access to perform this operation.

For more information, see "Preparing Identity Repositories" in the *Installation Guide*.

Server Settings

The following properties appear on the Server Settings tab:

LDAP Server

`host:port` to contact the directory server, with optional `|server_ID|site_ID` for deployments with multiple servers and sites.

AM uses the optional settings to determine which directory server to contact first. AM 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 AM server.
2. The first directory server in the list whose `site_ID` matches the current AM server.
3. The first directory server in the remaining list.

If the directory server is not available, AM proceeds to the next directory server in the list.

XUI

Default: `host:port` of the initial directory server configured for this AM server.

ssoadm

ssoadm attribute: `sun-idrepo-ldapv3-config-ldap-server`

Default: `=[0]=host:port` of the initial directory server configured for this AM server.

You must add `=[n]=` before the `host:port`, where `n` is an array index, starting with 0, of servers listed. See the example below.

Example:

```
sun-idrepo-ldapv3-config-ldap-server=[0]=localhost:51636|01
sun-idrepo-ldapv3-config-ldap-server=[1]=openam.example.com:52389|02
sun-idrepo-ldapv3-config-ldap-server=[2]=zzz.example.com:1636|01|02
sun-idrepo-ldapv3-config-ldap-server=[3]=xxx.example.com:1636|01|02
```

LDAP Bind DN

Bind DN of the service account AM uses to connect to the directory server. Some AM 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`

Proxied Authorization using Bind DN

When the `force-change-on-reset` password policy is configured on the DS user data store, users resetting their passwords using AM's forgotten password feature may be required to reset their passwords twice (prompted by both AM's User Self-Service and DS's password policy).

When the Proxied Authorization using Bind DN property is enabled, AM leverages DS's `proxied authorization` to reset user passwords acting as themselves rather than as the service account configured in the LDAP Bind DN property. This way, DS does not require users to reset their passwords again.

Before enabling this setting, ensure that the service account configured in the LDAP Bind DN property has the `proxied-auth` privilege granted. If the service account does not have

the required privilege, users would not be able to reset their passwords and AM and DS will log an error message.

For examples of setting the privileges required for the password reset feature, see "Installing and Configuring Directory Services for Identity Data" in the *Installation Guide*.

Enable this property only if:

- The `force-change-on-reset` password policy is configured in the DS user data store.
- The forgotten password user self-service feature is configured in AM.
- Users are being forced to reset their passwords twice.

ssoadm attribute: `openam-idrepo-ldapv3-proxied-auth-enabled`

Default: `Disabled`

Fallback using Bind DN if Proxied Authorization denied

Enable this setting to fallback and retry using non-proxied authorization (without the Directory Services `proxied-auth` privilege) when proxied authorization is denied.

Enabling this property causes AM to attempt to make LDAP changes as the LDAP Bind DN service account if proxied auth was unsuccessful; for example, if the user account attempting the changes originally is locked or the password has expired.

This setting is effective only when Proxied Authorization using Bind DN property is also enabled.

ssoadm attribute: `openam-idrepo-ldapv3-proxied-auth-denied-fallback`

Default: `Disabled`

LDAP Organization DN

The base DN under which to find user and group profiles.

Ensure that the identity store is setup with the specified DN before making any changes to this property in AM.

ssoadm attribute: `sun-idrepo-ldapv3-config-organization_name`

Default: `base-dn`

LDAP Connection Mode

Whether to use LDAP, LDAPS or StartTLS to connect to the directory server. When LDAPS or StartTLS are enabled, AM 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 AM runs, or because you imported the certificates into the trust store.

ssoadm attribute: `sun-idrepo-ldapv3-config-connection-mode`

Possible values: `LDAP`, `LDAPS`, and `StartTLS`

LDAP Connection Pool Minimum Size

Minimum number of connections to the directory server.

ssoadm attribute: `sun-idrepo-ldapv3-config-connection_pool_min_size`

Default: `1`

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 return, for example, when viewing profiles under Identities. Rather than raise this number, consider narrowing your search to match fewer directory entries.

ssoadm attribute: `sun-idrepo-ldapv3-config-max-result`

Default: `1000`

Search Timeout

Maximum time to wait for search results in seconds. Does not apply to persistent searches.

ssoadm attribute: `sun-idrepo-ldapv3-config-time-limit`

Default: `10`

LDAPv3 Plugin Search Scope

LDAP searches can apply to a single entry (SCOPE_BASE), entries directly below the search DN (SCOPE_ONE), or all entries below the search DN (SEARCH_SUB).

ssoadm attribute: `sun-idrepo-ldapv3-config-search-scope`

Default: `SCOPE_SUB`

Behera Support Enabled

Enable this property to use Behera draft control in outgoing requests for operations that may modify password values.

Behera draft control allows AM to display password policy related error messages when password policies are not met.

ssoadm attribute: `openam-idrepo-ldapv3-behera-support-enabled`

Default: `Enabled`

Affinity Enabled

Enables affinity-based load balanced access to the identity stores. Specify each of the directory server instances that form the affinity deployment in the LDAP Server field.

The directory server instance used for each operation is based on the DN of the identity involved.

Important

When enabled, you **must** use an identical LDAP Server value in every AM instance in the deployment.

ssoadm attribute: `openam-idrepo-ldapv3-affinity-enabled`

Default: **Disabled**

Plug-in Configuration

The following properties appear on the Plug-in Configuration tab:

LDAPv3 Repository Plugin Class Name

AM identity repository implementation.

ssoadm attribute: **sunIdRepoClass**

Default: **org.forgerock.openam.idrepo.ldap.DJLDAPv3Repo**

Attribute Name Mapping

Map of AM profile attribute names to directory server attribute names.

ssoadm attribute: **sunIdRepoAttributeMapping**

LDAPv3 Plugin Supported Types and Operations

Specifies the identity types supported by the data store, such as **user**, **group**, or **realm**, and which operations can be performed on them.

The following table illustrates the identity types supported by this data store, and the operations that can be performed on them:

Supported Identity Types and Operations

	read	create	edit	delete	service
group	✓	✓	✓	✓	-
realm	✓	✓	✓	✓	✓
user	✓	✓	✓	✓	✓
	Read the identity type	Create new identities of the given identity type	Edit entities of the given identity type	Delete entities of the given identity type	Read and write service settings associated with the given identity type.

You can remove permissions based on your data store needs. For example, if the data store should not be written to, you can set the operations to **read** only for the identity types.

The **service** operation is only relevant to the **realm** and the **user** identity types. For example, the Session Service configuration can be stored by realm, and a user can have specific session timeout settings.

ssoadm attribute: `sunIdRepoSupportedOperations`

Default:

```
group=read,create,edit,delete
realm=read,create,edit,delete,service
user=read,create,edit,delete,service
```

User Configuration

The following properties appear on the User Configuration tab:

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`

Warning

Do not modify the value of the search attribute in user profiles. Modifying this attribute value can result in incorrectly cached identity data. For example, if you configure the search attribute to `mail`, it could prevent users from being able to update their email addresses in their user profiles.

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 User Object Class

User profiles have these LDAP object classes.

AM handles only those attributes listed in this setting. AM discards any unlisted attributes from requests and the request proceeds without the attribute.

For example, with default settings, if you request that AM execute a search that asks for the `mailAlternateAddress` attribute, AM does the search, but does not request `mailAlternateAddress`. In the same way, AM does perform an update operation with a request

to set the value of an unlisted attribute like `mailAlternateAddress`, but it drops the unlisted attribute from the update request.

ssoadm attribute: `sun-idrepo-ldapv3-config-user-objectclass`

Default: `devicePrintProfilesContainer, forgerock-am-dashboard-service, iPlanetPreferences, inetorgperson, inetuser, iplanet-am-auth-configuration-service, iplanet-am-managed-person, iplanet-am-user-service, kbaInfoContainer, oathDeviceProfilesContainer, organizationalperson, person, pushDeviceProfilesContainer, sunAMAuthAccountLockout, sunFMSAML2NameIdentifier, sunFederationManagerDataStore, sunIdentityServerLibertyPPService, top`

LDAP User Attributes

User profiles have these LDAP attributes.

AM handles only those attributes listed in this setting. AM discards any unlisted attributes from requests and the request proceeds without the attribute.

ssoadm attribute: `sun-idrepo-ldapv3-config-user-attributes`

Default: `adminRole, assignedDashboard, authorityRevocationList, caCertificate, cn, createTimeStamp, devicePrintProfiles, distinguishedName, dn, employeeNumber, givenName, inetUserHttpURL, inetUserStatus, iplanet-am-auth-configuration, iplanet-am-session-destroy-sessions, iplanet-am-session-get-valid-sessions, iplanet-am-session-max-caching-time, iplanet-am-session-max-idle-time, iplanet-am-session-max-session-time, iplanet-am-session-quota-limit, iplanet-am-session-service-status, iplanet-am-user-account-life, iplanet-am-user-admin-start-dn, iplanet-am-user-alias-list, iplanet-am-user-auth-config, iplanet-am-user-auth-modules, iplanet-am-user-failure-url, iplanet-am-user-federation-info, iplanet-am-user-federation-info-key, iplanet-am-user-login-status, iplanet-am-user-password-reset-force-reset, iplanet-am-user-password-reset-options, iplanet-am-user-password-reset-question-answer, iplanet-am-user-success-url, kbaActiveIndex, kbaInfo, mail, manager, memberOf, modifyTimestamp, oath2faEnabled, oathDeviceProfiles, objectClass, postalAddress, preferredLocale, preferredlanguage, preferredtimezone, pushDeviceProfiles, sn, sun-fm-saml2-nameid-info, sun-fm-saml2-nameid-infokey, sunAMAuthInvalidAttemptsData, sunIdentityMSISDNNumber, sunIdentityServerDiscoEntries, sunIdentityServerPPAddressCard, sunIdentityServerPPCommonNameAltCN, sunIdentityServerPPCommonNameCN, sunIdentityServerPPCommonNameFN, sunIdentityServerPPCommonNameMN, sunIdentityServerPPCommonNamePT, sunIdentityServerPPCommonNameSN, sunIdentityServerPPDemographicsAge, sunIdentityServerPPDemographicsBirthDay, sunIdentityServerPPDemographicsDisplayLanguage, sunIdentityServerPPDemographicsLanguage, sunIdentityServerPPDemographicsTimeZone, sunIdentityServerPPEmergencyContact, sunIdentityServerPPEmploymentIdentityAlt0, sunIdentityServerPPEmploymentIdentityJobTitle, sunIdentityServerPPEmploymentIdentityOrg, sunIdentityServerPPEncryPTKey, sunIdentityServerPPFacadeGreetSound, sunIdentityServerPPFacadeMugShot, sunIdentityServerPPFacadeNamePronounced, sunIdentityServerPPFacadeWebSite, sunIdentityServerPPFacadeGreetmesound, sunIdentityServerPPInformalName, sunIdentityServerPPLegalIdentityAltIdType, sunIdentityServerPPLegalIdentityAltIdValue, sunIdentityServerPPLegalIdentityDOB, sunIdentityServerPPLegalIdentityGender, sunIdentityServerPPLegalIdentityLegalName, sunIdentityServerPPLegalIdentityMaritalStatus,`

`sunIdentityServerPPLegalIdentityVATIdType`, `sunIdentityServerPPLegalIdentityVATIdValue`,
`sunIdentityServerPPMsgContact`, `sunIdentityServerPPSignKey`, `telephoneNumber`, `uid`, `userCertificate`,
`userPassword`

Create User Attribute Mapping

When creating a user profile, apply this map of AM 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`

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.

If specified, AM will limit searches for user profiles to the provided base DN. Otherwise, AM searches the entire directory.

ssoadm attribute: `sun-idrepo-ldapv3-config-people-container-value`

Default: `people`

Knowledge Based Authentication Attribute Name

Profile attribute in which knowledge-based authentication information is stored.

ssoadm attribute: `sun-idrepo-ldapv3-config-auth-kba-attr`

Default: `kbaInfo`

Knowledge Based Authentication Active Index

Profile attribute in the which knowledge-based authentication index is stored.

ssoadm attribute: `sun-idrepo-ldapv3-config-auth-kba-index-attr`

Default: `kbaActiveIndex`

Knowledge Based Authentication Attempts Attribute Name

Profile attribute in which the number of failed attempts by a user when completing knowledge-based authentication information is stored.

ssoadm attribute: `sun-idrepo-ldapv3-config-auth-kba-attempts-attr`

Default: `kbaInfoAttempts`

Authentication Configuration

The following properties appear on the Authentication Configuration tab:

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`

Group Configuration

The following properties appear on the Group Configuration tab:

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.

If specified, AM will limit searches for group profiles to the provided base DN. Otherwise, AM searches the entire directory.

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`

Attribute Name of Group Member URL

Attribute in the group's LDAP entry whose values are LDAP URLs which define dynamic members of the group.

ssoadm attribute: `sun-idrepo-ldapv3-config-memberurl`

Default: `memberUrl`

Persistent Search Controls

The following properties appear on the Persistent Search Controls tab:

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`

Error Handling Configuration

The following properties appear on the Error Handling Configuration tab:

The Delay Time Between Retries

How long to wait after receiving an error result that indicates AM should try the LDAP operation again.

The DS data store uses this setting only for persistent searches.

ssoadm attribute: `com.iplanet.am.ldap.connection.delay.between.retries`

Default: `1000 milliseconds`

Cache Control

The following properties appear on the Cache Control tab:

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 AM uses persistent searches to obtain notification of such updates.

ssoadm attribute: `sun-idrepo-ldapv3-dncache-enabled`

Default: `true`

DN Cache Size

Maximum number of DN's cached when caching is enabled.

ssoadm attribute: `sun-idrepo-ldapv3-dncache-size`

Default: `1500 items`

+ Tivoli Directory Server Configuration Properties

Use these attributes when configuring Tivoli Directory Server data stores:

amster service name: `IdRepository`

ssoadm service name: `sunIdentityRepositoryService`

ID

The ID of the data store configuration.

LDAP Server

`host:port` to contact the directory server, with optional `|server_ID|site_ID` for deployments with multiple servers and sites.

AM uses the optional settings to determine which directory server to contact first. AM 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 AM server.
2. The first directory server in the list whose `site_ID` matches the current AM server.
3. The first directory server in the remaining list.

If the directory server is not available, AM proceeds to the next directory server in the list.

XUI

Default: `host:port` of the initial directory server configured for this AM server.

ssoadm

ssoadm attribute: `sun-idrepo-ldapv3-config-ldap-server`

Default: `=[0]=host:port` of the initial directory server configured for this AM server.

You must add `=[n]=` before the `host:port`, where `n` is an array index, starting with 0, of servers listed. See the example below.

Example:

```
sun-idrepo-ldapv3-config-ldap-server=[0]=localhost:51636|01  
sun-idrepo-ldapv3-config-ldap-server=[1]=openam.example.com:52389|02  
sun-idrepo-ldapv3-config-ldap-server=[2]=zzz.example.com:1636|01|02  
sun-idrepo-ldapv3-config-ldap-server=[3]=xxx.example.com:1636|01|02
```

LDAP Bind DN

Bind DN of the service account AM uses to connect to the directory server. Some AM 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.

Ensure that the identity store is setup with the specified DN before making any changes to this property in AM.

ssoadm attribute: `sun-idrepo-ldapv3-config-organization_name`

Default: `base-dn`

LDAP Connection Mode

Whether to use LDAP, LDAPS or StartTLS to connect to the directory server. When LDAPS or StartTLS are enabled, AM 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 AM runs, or because you imported the certificates into the trust store.

ssoadm attribute: `sun-idrepo-ldapv3-config-connection-mode`

Possible values: `LDAP`, `LDAPS`, and `StartTLS`

LDAP Connection Pool Maximum Size

Maximum number of connections to the directory server. Make sure the directory service can cope with the maximum number of client connections across all servers.

ssoadm attribute: `sun-idrepo-ldapv3-config-connection_pool_max_size`

Default: 10

LDAP Connection Heartbeat Interval

How often to send a heartbeat request to the directory server to ensure that the connection does not remain idle. Some network administrators configure firewalls and load balancers to drop connections that are idle for too long. You can turn this off by setting the value to 0 or to a negative number. To set the units for the interval, use LDAP Connection Heartbeat Time Unit.

ssoadm attribute: `openam-idrepo-ldapv3-heartbeat-interval`

Default: 10

LDAP Connection Heartbeat Time Unit

Time unit for the LDAP Connection Heartbeat Interval setting.

ssoadm attribute: `openam-idrepo-ldapv3-heartbeat-timeunit`

Default: `second`

Maximum Results Returned from Search

A cap for the number of search results to return, for example, when viewing profiles under Identities. Rather than raise this number, consider narrowing your search to match fewer directory entries.

ssoadm attribute: `sun-idrepo-ldapv3-config-max-result`

Default: 1000

Search Timeout

Maximum time to wait for search results in seconds. Does not apply to persistent searches.

ssoadm attribute: `sun-idrepo-ldapv3-config-time-limit`

Default: 10

LDAPv3 Plugin Search Scope

LDAP searches can apply to a single entry (SCOPE_BASE), entries directly below the search DN (SCOPE_ONE), or all entries below the search DN (SEARCH_SUB).

ssoadm attribute: `sun-idrepo-ldapv3-config-search-scope`

Default: `SCOPE_SUB`

LDAPv3 Repository Plugin Class Name

AM identity repository implementation.

ssoadm attribute: `sunIdRepoClass`

Default: `org.forgerock.openam.idrepo.ldap.DJLDAPv3Repo`

Attribute Name Mapping

Map of AM profile attribute names to directory server attribute names.

ssoadm attribute: `sunIdRepoAttributeMapping`

LDAPv3 Plugin Supported Types and Operations

Specifies the identity types supported by the data store, such as `user`, `group`, or `realm`, and which operations can be performed on them.

The following table illustrates the identity types supported by this data store, and the operations that can be performed on them:

Supported Identity Types and Operations

	read	create	edit	delete	service
group	✓	✓	✓	✓	-
realm	✓	✓	✓	✓	✓
user	✓	✓	✓	✓	✓
	Read the identity type	Create new identities of the given identity type	Edit entities of the given identity type	Delete entities of the given identity type	Read and write service settings associated with the given identity type.

You can remove permissions based on your data store needs. For example, if the data store should not be written to, you can set the operations to `read` only for the identity types.

The `service` operation is only relevant to the `realm` and the `user` identity types. For example, the Session Service configuration can be stored by realm, and a user can have specific session timeout settings.

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`

Warning

Do not modify the value of the search attribute in user profiles. Modifying this attribute value can result in incorrectly cached identity data. For example, if you configure the search attribute to `mail`, it could prevent users from being able to update their email addresses in their user profiles.

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.

If specified, AM will limit searches for user profiles to the provided base DN. Otherwise, AM searches the entire directory.

ssoadm attribute: `sun-idrepo-ldapv3-config-people-container-value`

LDAP User Object Class

User profiles have these LDAP object classes.

AM handles only those attributes listed in this setting. AM discards any unlisted attributes from requests and the request proceeds without the attribute.

For example, with default settings if you request that AM execute a search that asks for the `mailAlternateAddress` attribute, AM does the search, but does not request `mailAlternateAddress`. In the same way, AM 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.

AM handles only those attributes listed in this setting. AM discards any unlisted attributes from requests and the request proceeds without the attribute.

ssoadm attribute: `sun-idrepo-ldapv3-config-user-attributes`

Default: `adminRole, assignedDashboard, authorityRevocationList, caCertificate, cn, devicePrintProfiles, distinguishedName, dn, employeeNumber, givenName, inetUserHttpURL, inetUserStatus, iplanet-am-auth-configuration, iplanet-am-session-add-session-listener-on-all-sessions, iplanet-am-session-destroy-sessions, iplanet-am-session-get-valid-sessions, iplanet-am-session-max-caching-time, iplanet-am-session-max-idle-time, iplanet-am-session-max-session-time, iplanet-am-session-quota-limit, iplanet-am-session-service-status, iplanet-am-user-account-life, iplanet-am-user-admin-start-dn, iplanet-am-user-alias-list, iplanet-am-user-auth-config, iplanet-am-user-auth-modules, iplanet-am-user-failure-url, iplanet-am-user-federation-info-key, iplanet-am-user-federation-info, iplanet-am-user-login-status, iplanet-am-user-password-reset-force-reset, iplanet-am-user-password-reset-options, iplanet-am-user-password-reset-question-answer, iplanet-am-user-success-url, mail, manager, memberOf, objectClass, postalAddress, preferredlanguage, preferredLocale, preferredtimezone, sn, sun-fm-saml2-nameid-info, sun-fm-saml2-nameid-infokey, sunAMAuthInvalidAttemptsData, sunIdentityMSISDNNumber, sunIdentityServerDiscoEntries, sunIdentityServerPPAddressCard, sunIdentityServerPPCommonNameAltCN, sunIdentityServerPPCommonNameCN, sunIdentityServerPPCommonNameFN, sunIdentityServerPPCommonNameMN, sunIdentityServerPPCommonNamePT, sunIdentityServerPPCommonNameSN, sunIdentityServerPPDemographicsAge, sunIdentityServerPPDemographicsBirthDay, sunIdentityServerPPDemographicsDisplayLanguage, sunIdentityServerPPDemographicsLanguage, sunIdentityServerPPDemographicsTimeZone, sunIdentityServerPPEmergencyContact, sunIdentityServerPPEmploymentIdentityAlt0, sunIdentityServerPPEmploymentIdentityJobTitle, sunIdentityServerPPEmploymentIdentityOrg, sunIdentityServerPPEncryPTKey, sunIdentityServerPPFacadegreetmesound, sunIdentityServerPPFacadeGreetSound, sunIdentityServerPPFacadeMugShot, sunIdentityServerPPFacadeNamePronounced, sunIdentityServerPPFacadeWebSite, sunIdentityServerPPInformalName,`

```
sunIdentityServerPPLegalIdentityAltIdType, sunIdentityServerPPLegalIdentityAltIdValue,
sunIdentityServerPPLegalIdentityDOB, sunIdentityServerPPLegalIdentityGender,
sunIdentityServerPPLegalIdentityLegalName, sunIdentityServerPPLegalIdentityMaritalStatus,
sunIdentityServerPPLegalIdentityVATIdType, sunIdentityServerPPLegalIdentityVATIdValue,
sunIdentityServerPPMsgContact, sunIdentityServerPPSignKey, telephoneNumber, uid, userCertificate,
userPassword
```

Create User Attribute Mapping

When creating a user profile, apply this map of AM 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.

If specified, AM will limit searches for group profiles to the provided base DN. Otherwise, AM searches the entire directory.

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 AM should try the LDAP operation again.

ssoadm attribute: `com.ipplanet.am.ldap.connection.delay.between.retries`

Default: 1000 milliseconds

DN Cache Enabled

Whether to enable the DN cache, which is used to cache DN lookups that can happen in bursts during authentication. As the cache can become stale when a user is moved or renamed, enable DN caching when the directory service allows move/rename operations (Mod DN), and when AM uses persistent searches to obtain notification of such updates.

ssoadm attribute: `sun-idrepo-ldapv3-dncache-enabled`

Default: true

DN Cache Size

Maximum number of DN's cached when caching is enabled.

ssoadm attribute: `sun-idrepo-ldapv3-dncache-size`

Default: 1500 items

Load Schema

Import appropriate LDAP schema to the directory server before saving the configuration. The LDAP Bind DN service account must have access to perform this operation.

This option is not available for use with the **ssoadm** command.

Default: Disabled.

6. (Optional) If you have not applied the schema configuration to your identity data, but the AM service account used to bind to the directory service has permission to alter schema, enable the Load Schema option.
7. Click Save Changes.
8. If you no longer need the connection to the inherited, embedded identity store *in this realm*, then you can delete its entry in the Identity Stores list.

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

9. To test the connection, see "Testing External Identity Repository Access".

Testing External Identity Repository Access

You should verify that you have configured the repository and administrator privileges correctly. You can test configuration as follows:

- Attempt to create an AM user by navigating to Realms > *Realm Name* > Identities in the AM console. Run this test only if you have given the AM bind account write privileges to your identity repository.

For example, create a **demo** user. When creating a **demo** user's account, set the fields as follows:

Demo User Account Settings

Field	Value
ID	demo
First Name	Leave this field blank.
Last Name	demo
Full Name	demo
Password	Ch4ng31t
User Status	Active

- Attempt to access an AM user from Realms > *Realm Name* > Identities in the AM console.

If you receive an LDAP error code 65 while attempting to create a user, it indicates that you did not correctly prepare the identity repository. Error code 65 is an LDAP object class violation and often indicates a problem with the directory schema or permissions.

A common reason for this error while attempting to create a user is that the bind account might not have adequate rights to add data to the directory. Review the DS **access** log and locate the entries for the *add* operation to determine if it is an access rights issue.

For information on setting up Directory Services as an identity store, see "Installing and Configuring Directory Services for Identity Data" in the *Installation Guide*.

Customizing Identity Stores

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

- "Adding User Profile Attributes".
- "Customizing Identity Data Storage".

Adding User Profile Attributes

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

Adding a custom attribute involves both updating the identity repository schema to hold the new attribute and updating the UI. Furthermore, to give users write permissions to the custom attribute, you must also update the AM configuration store.

This section uses DS 5 or later in the examples and includes the following procedures:

- To Update the Identity Repository for the New Attribute
- To Allow Users To Update the New Attribute Using an LDAP Browser
- To Allow Users to Update the New Attribute Using the Command Line
- To Add Custom Attributes to the User Interface

To Update the Identity Repository for the New Attribute

Perform the following steps to update the identity repository schema for the custom attribute, and then update AM to use the custom attribute and object class.

If you intend to use an existing attribute that is already allowed on user profile entries, you can skip this procedure.

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

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

In the example, the attribute type is called `customAttribute` and the object class is called `customObjectclass`.

2. Add the schema definitions to the directory.

```
$ /path/to/openssl/bin/ldapmodify \
--hostname 'id.example.com' \
--port 1636 \
--useSsl \
--usePkcs12TrustStore /path/to/openssl/config/keystore \
--trustStorePasswordFile /path/to/openssl/config/keystore.pin \
--bindDN uid=admin \
--bindPassword str0ngAdm1nPa55word \
/path/to/custom-attr.ldif
Processing MODIFY request for cn=schema
MODIFY operation successful for DN cn=schema
```

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

The profile attribute whitelist controls the information returned to non-administrative users when accessing `json/user` endpoints. For example, the whitelist controls the attributes shown in the user profile page.

Common profile attributes are whitelisted by default, but you need to add any custom attribute you want your non-administrative users to see.

The whitelist can be set by realm, in the user self-service service, or globally. To modify it:

- **Globally:** Navigate to Configure > Global Services > User Self-Service > Profile Management, and edit the Self readable attributes field.
- **By realm:** Navigate to Realms > *Realm Name* > Services > User Self-Service > Profile Management, and edit the Self readable attributes field.

Note that you need to add the user self-service service to the realm if you have not done so already, but you do not need to configure anything other than the whitelist.

To Allow Users To Update the New Attribute Using an LDAP Browser

Perform the following steps to update AM configuration store to give users write permission to the custom attribute.

This procedure assumes you use an LDAP browser, for example [Apache Directory Studio](#). Alternatively, you can follow ["To Allow Users to Update the New Attribute Using the Command Line"](#) if you use the command line.

1. Connect to the AM configuration store. You can see the configuration store details by navigating to Deployment > Servers > Directory Configuration > Server.
2. Search for `ou=SelfWriteAttributes`. You should find DN's similar to the following:
 - `dn:ou=SelfWriteAttributes,ou=Policies,ou=default,ou=OrganizationConfig,ou=1.0,ou=iPlanetAMPolicyService,ou=services,o=sunamhiddenrealmdelegationsservicepermissions,ou=services,dc=openam,dc=forgerock,dc=org`
 - `dn:ou=SelfWriteAttributes,ou=default,ou=default,ou=OrganizationConfig,ou=1.0,ou=sunEntitlementIndexes,ou=services,o=sunamhiddenrealmdelegationsservicepermissions,ou=services,dc=openam,dc=forgerock,dc=org`
3. In the entry under `iPlanetAMPolicyService`, edit the `sunKeyValue` attribute to add the custom attribute to the list of self-writable attributes. For example, `<Value>customAttribute</Value>`.
4. In the entry under `sunEntitlementIndexes`, edit the `sunKeyValue` attribute to add the custom attribute to the list of self-writable attributes. The following is an example of the custom attribute as the first element of the list: `\\"attributes\\": [\\n \\\"customAttribute\\\",\\n ...`
5. Restart AM or the web container where it runs.

To Allow Users to Update the New Attribute Using the Command Line

Perform the following steps to update the AM configuration store to give users write permission to the custom attribute.

This procedure assumes you use the command line. Alternatively, you can follow ["To Allow Users To Update the New Attribute Using an LDAP Browser"](#) if you use an LDAP browser.

1. Search for the value of `sunKeyValue` in `ou=SelfWriteAttributes` by running the following command:

```
$ /path/to/openssl/bin/ldapssearch --hostname openam.example.com --port 1636 \
--bindDn uid=admin --bindPassword forgerock \
--baseDn "dc=openam,dc=forgerock,dc=org" "(ou=SelfWriteAttributes)" sunKeyValue

dn:
ou=SelfWriteAttributes,ou=Policies,ou=default,ou=OrganizationConfig,ou=1.0,ou=iPlanetAMPolicyService,ou=services
sunKeyValue:: eGlscG9saWN5PTw.....

dn:
ou=SelfWriteAttributes,ou=default,ou=default,ou=OrganizationConfig,ou=1.0,ou=sunEntitlementIndexes,ou=services
sunKeyValue: serializable={"eCondition":{"className":"com.sun....
```

Note that the command returns two DN's, and the value of `sunKeyValue` in the first one is base64-encoded.

2. Decode the base64 string of the `iPlanetAMPolicyService` DN. For example:

```
$ ./base64 decode --encodedData eG1scG9saWN5PTw22.....
xmlpolicy=<?xml version="1.0" encoding="UTF-8"?>
<Policy name="SelfWriteAttributes" createdby="cn=dsameuser,ou=DSAME
Users,dc=openam,dc=forgerock,dc=org" lastmodifiedby="cn=dsameuser,ou=DSAME
Users,dc=openam,dc=forgerock,dc=org" creationdate="1528296269883" lastmodifieddate="1528296269883"
referralPolicy="false" active="true" >
<Rule name="user-read-rule">
  <ServiceName name="sunAMDelegationService" />
  <ResourceName name="sms://*dc=openam,dc=forgerock,dc=org/sunIdentityRepositoryService/1.0/
application/*" />
  <AttributeValuePair>
    <Attribute name="MODIFY" />
    <Value>allow</Value>
  </AttributeValuePair>
</Rule>
<Subjects name="Subjects" description="">
  <Subject name="delegation-subject" type="AuthenticatedUsers" includeType="inclusive">
  </Subject>
</Subjects>
<Conditions name="AttrCondition" description="">
  <Condition name="condition" type="UserSelfCheckCondition">
    <AttributeValuePair><Attribute name="attributes"/><Value>givenname</Value><Value>sn</
Value><Value>cn</Value><Value>userpassword</Value><Value>mail</Value><Value>telephonenumber</
Value><Value>postaladdress</Value><Value>preferredlocale</Value><Value>iplanet-am-user-
password-reset-options</Value><Value>iplanet-am-user-password-reset-question-answer</
Value><Value>description</Value><Value>oath2faEnabled</Value><Value>sunIdentityServerDeviceKeyValue</
Value><Value>sunIdentityServerDeviceStatus</Value>
    </AttributeValuePair>
  </Condition>
</Conditions>
</Policy>
```

3. Create a file with the decoded string. Then, add the custom attribute to the `<AttributeValuePair>` list. For example:

```
$ vi to-encode.xml
....
<Attribute name="attributes"/><Value>customAttribute</Value><Value>givenname</Value>...</
AttributeValuePair>
....
```

4. Base64-encode the content of the file. For example:

```
$ ./base64 encode -f to-encode.xml
EG1scG9saWN5PTw22.....
```

5. Create an LDIF file, for example, `change.ldif`, containing the following:

- The LDIF properties and rules required to modify the value of the `sunKeyValue` attribute for both DNs.
- The newly base64-encoded string as the value of the `sunKeyValue` attribute of the `iPlanetAMPolicyService` DN. The string already contains the custom attribute.
- The value of the `sunKeyValue` attribute of the `sunEntitlementIndexes` DN. You must add the custom attribute inside the `attributes` list.

The following excerpt is an example of LDIF file:

```
dn:
  ou=SelfWriteAttributes,ou=Policies,ou=default,ou=OrganizationConfig,ou=1.0,ou=iPlanetAMPolicyService,ou=services
changetype: modify
replace: sunKeyValue
sunKeyValue: EG1scG9saWN5PTw22.....

dn:
  ou=SelfWriteAttributes,ou=default,ou=default,ou=OrganizationConfig,ou=1.0,ou=sunEntitlementIndexes,ou=services
changetype: modify
replace: sunKeyValue
sunKeyValue: serializable={"eCondition":{"className": ... \ "properties\ ": {"attributes\ ": [{"n
  \ "customAttribute\ ",\n \ "givenname\ ",\n \ "sn\ ",\n ... \ "values\ ": [{"n}]}}}
```

6. Apply the changes in the LDIF file to the LDAP configuration store, as follows:

```
$ /path/to/openssl/bin/openssl \
--hostname 'id.example.com' \
--port 1636 \
--useSsl \
--usePkcs12TrustStore /path/to/openssl/config/keystore \
--trustStorePasswordFile /path/to/openssl/config/keystore.pin \
--bindDn uid=admin \
--bindPassword strongAdminPa55word \
--filename change.ldif
# MODIFY operation successful for DN
ou=SelfWriteAttributes,ou=Policies,ou=default,ou=OrganizationConfig,ou=1.0,ou=iPlanetAMPolicyService,ou=services
# MODIFY operation successful for DN
ou=SelfWriteAttributes,ou=default,ou=default,ou=OrganizationConfig,ou=1.0,ou=sunEntitlementIndexes,ou=services
```

7. Restart AM or the web container where it runs.

To Add Custom Attributes to the User Interface

To allow the UI to show the new attribute in the user profile, you need to download the UI source, edit it, and then rebuild the UI.

Perform the following steps to configure the UI to show the new attribute:

1. Download the UI source as explained in "[Downloading the UI](#)" in the *UI Customization Guide*.
2. Modify the UI as follows:
 - a. Edit the `openam-ui-user/src/resources/locales/en/translation.json` file and add a new line with the description for the custom attribute. This description will show in the UI user's profile page. For example:

```
...
"profile": "Profile",
"username" : "Username",
"emailAddress" : "Email address",
"givenName" : "First Name",
"customAttribute" : "My Custom Attribute",
"sn" : "Last Name",
"changePassword" : "Change password",
...
```

Note that the example adds the custom attribute under the `common.user` JSON path.

Tip

If you have translated the UI pages, remember to edit all the `translation.json` files in your installation.

- b. Edit the `openam-ui-user/src/resources/templates/user/UserProfileTemplate.html` file and add a new line for the custom attribute. Consider the following points:
 - `property` must contain the name of the custom attribute created in the LDAP. For example, `customAttribute`.
 - `label` must contain the path to the label created in the `translation.json` file. In this case, `common.user.customAttribute`.

For example:

```
{{#user}}
{{> form/_basicInput property="username" label="common.user.username" readonly=true}}
{{> form/_basicInput property="givenName" label="common.user.givenName"}}
{{> form/_basicInput property="sn" label="common.user.sn" required="true"}}
{{> form/_basicInput type="email" property="mail" label="common.user.emailAddress"
extraAttributes='data-validator="validEmailAddressFormat" data-validator-event="keyup"' }}
{{> form/_basicInput type="tel" property="telephoneNumber" label="common.user.phoneNumber"
extraAttributes='data-validator="validPhoneFormat" data-validator-event="keyup"' }}
{{> form/_basicInput property="customAttribute" label="common.user.customAttribute"}}
{/user}}
```

- c. Edit the `openam-ui-user/src/js/org/forgerock/openam/ui/user/UserModel.js` file and add the custom attribute on the `ServiceInvoker.restCall` function.

Consider the following constraints when modifying this file:

- The file does not support tab indentation. You must use space indentation.
- The file does not support lines longer than 120 characters. If the line you are modifying exceeds this limit, break it in multiple lines.

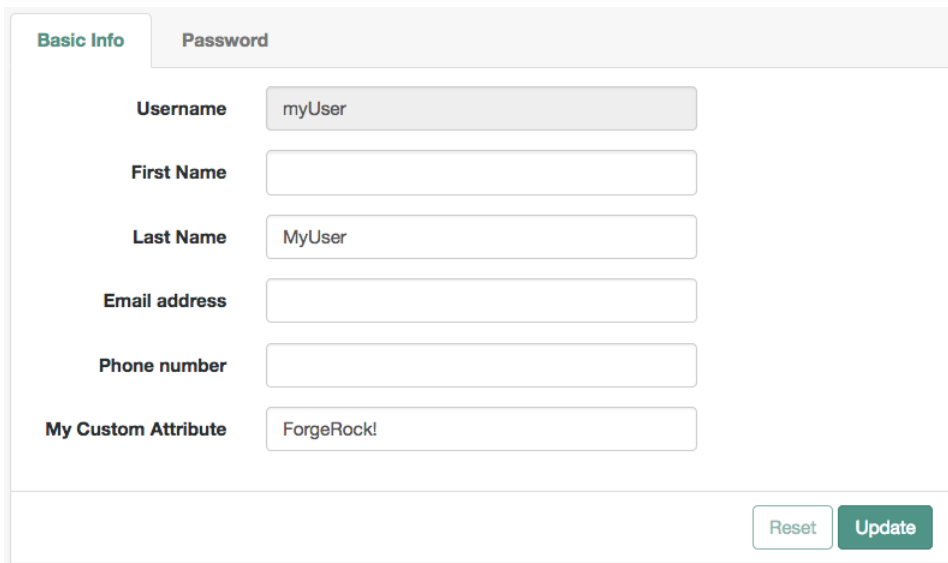
For example:

```
return ServiceInvoker.restCall(_.extend(
{
  type: "PUT",
  data: JSON.stringify(
    _.chain(this.toJSON())
      .pick(["givenName", "sn", "mail", "postalAddress", "telephoneNumber", "customAttribute"])
      .mapValues((val) => {
        ...
      })
  )
})
```

3. Rebuild the UI by running the **yarn build** command.
4. Test the UI pages by following the steps detailed in "To Test UI Pages in a Development Server" in the *UI Customization Guide*.

The UI user profile page now shows the custom attribute, and users are able to read and write its values:

User Custom Attribute



5. Once you are satisfied with the changes, deploy the output in the **build** directory to the `/path/to/tomcat/webapps/openam/XUI/` directory of your AM instances.

There is no need to restart the AM instance. Subsequent visits to the UI pages will use the rebuilt files.

Customizing Identity Data Storage

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

About the Identity Repository Plugin

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

IdRepo Inheritance

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

IdRepo Lifecycle

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

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

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

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

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

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

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

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

```
public void shutdown()
```

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

IdRepo Plugin Capabilities

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

```
public Set getSupportedTypes()
```

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

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

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

```
public boolean supportsAuthentication()
```

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

Identity Repository Plugin Implementation

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

Create

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

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

Read

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

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

public boolean isActive(
    SSOToken token,
    IdType type,
    String name
```

```

) throws IdRepoException, SS0Exception

public Map getAttributes(
    SS0Token token,
    IdType type,
    String name
) throws IdRepoException, SS0Exception

public Map getAttributes(
    SS0Token token,
    IdType type,
    String name,
    Set attrNames
) throws IdRepoException, SS0Exception

public Map getBinaryAttributes(
    SS0Token token,
    IdType type,
    String name,
    Set attrNames
) throws IdRepoException, SS0Exception

public RepoSearchResults search(
    SS0Token token,
    IdType type,
    String pattern,
    Map avPairs,
    boolean recursive,
    int maxResults,
    int maxTime,
    Set returnAttrs
) throws IdRepoException, SS0Exception

public RepoSearchResults search(
    SS0Token token,
    IdType type,
    String pattern,
    int maxTime,
    int maxResults,
    Set returnAttrs,
    boolean returnAllAttrs,
    int filterOp,
    Map avPairs,
    boolean recursive
) throws IdRepoException, SS0Exception

```

Edit

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

```

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

```

```
public void setBinaryAttributes(
    SSOToken token,
    IdType type,
    String name,
    Map attributes,
    boolean isAdd
) throws IdRepoException, SSOException

public void removeAttributes(
    SSOToken token,
    IdType type,
    String name,
    Set attrNames
) throws IdRepoException, SSOException

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

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

Authenticate

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

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

Delete

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

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

Service

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

Identity Repository Plugin Deployment

When you build your IdRepo plugin, include `openam-core-7.0.2.jar` in the classpath. This file is found under `WEB-INF/lib/` where AM is deployed.

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

To Register Your Plugin With AM (Plugin Tools API)

The steps in this procedure make use of a number of AM API interfaces and annotations. Click the following links to view the *AM Public API JavaDoc*:

- [PluginTools](#)
- [AmPlugin](#)
- [IdRepoConfig](#)

To register your custom ID repo in AM by using the built-in `PluginTools` interface, perform the following steps:

1. Use the `@IdRepoConfig` annotation on your configuration interface, as shown below:

```
package com.example.custom;

import java.util.Optional;

import org.forgerock.openam.annotations.sm.Attribute;
import org.forgerock.openam.annotations.sm.IdRepoConfig;

/**
 * Custom IdRepo config.
 */
@IdRepoConfig(name = "MyIdRepo")
public interface CustomIdRepoConfig {

    /**
     * The IdRepo implementation fully qualified class name.
     *
     * @return The implementation class name.
     */
    @Attribute(order = 10, requiredValue = true)
    default String idRepoClass() {
        return CustomIdRepo.class.getCanonicalName();
    }

    /**
     * Sets the connection pool minimum size.
     *
     * @return The connection pool minimum size.
     */
    @Attribute(order = 20)
    default Optional<Integer> connectionPoolMinSize() {
        return Optional.of(1);
    }

    /**
     * Sets the connection pool max size.
     */
}
```

```
* @return The connection pool max size.
*/
@Attribute(order = 30)
default Optional<Integer> connectionPoolMaxSize() {
    return Optional.of(10);
}
}
```

2. Create a `.properties` file based on the name you provided in the configuration interface; for example, `MyIdRepo.properties`.

The contents of the file might resemble the following:

```
CustomIdRepoConfig=Custom IdRepo
idRepoClass=LDAPv3 Repository Plug-in Class Name
connectionPoolMinSize=LDAP Connection Pool Minimum Size
connectionPoolMaxSize=LDAP Connection Pool Maximum Size
```

3. Create a class that implements `AmPlugin`, and uses the `pluginTools` interface to handle the following events:

- `onInstall()`

Call the `pluginTools.installIdRepo` function with your configuration class as the parameter.

- `onStartup()`

Call the `pluginTools.startService` function with your configuration class as the parameter.

- `upgrade()`

Call the `pluginTools.upgradeIdRepo` function with your configuration class as the parameter.

Your plugin class may resemble the following:

```
package com.example.custom;

import javax.inject.Inject;

import org.forgerock.openam.plugins.AmPlugin;
import org.forgerock.openam.plugins.PluginException;
import org.forgerock.openam.plugins.PluginTools;
import org.forgerock.openam.plugins.StartupType;

/**
 * A custom identity repository plugin. This uses the plugin framework to install the custom identity
 * repository.
 */
public class CustomIdRepoPlugin implements AmPlugin {

    private static final String CURRENT_VERSION = "1.0.0";
    private PluginTools pluginTools;

    /**
     * The constructor.
     */
}
```

```

    * @param pluginTools The PluginTools instance.
    */
    @Inject
    public CustomIdRepoPlugin(PluginTools pluginTools) {
        this.pluginTools = pluginTools;
    }

    @Override
    public String getPluginVersion() {
        return CustomIdRepoPlugin.CURRENT_VERSION;
    }

    @Override
    public void onInstall() throws PluginException {
        pluginTools.installIdRepo(CustomIdRepoConfig.class);
    }

    @Override
    public void onStartup(StartupType startupType) throws PluginException {
        pluginTools.startService(CustomIdRepoConfig.class);
    }

    @Override
    public void upgrade(String fromVersion) throws PluginException {
        pluginTools.upgradeIdRepo(CustomIdRepoConfig.class);
    }
}

```

To Register Your Plugin With AM (ssoadm)

To create a custom ID repo plugin for your identity stores and register it with the **ssoadm** command, perform the following steps:

1. Create a **SubSchema** for your plugin, and register it to AM:
 - a. Create the **SubSchema** document, for example **customIdRepo.xml**, using the following structure:

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

Ensure that you include the `AttributeSchema` required to configure your IdRepo plugin.

Notice the `i18nKey` attributes on `SubSchema` elements. The `i18nKey` attribute values correspond to properties in the `amIdRepoService.properties` file under `WEB-INF/classes/` where AM is deployed. The AM console displays the label for the configuration user interface that it retrieves from the value of the `i18nKey` property in the `amIdRepoService.properties` file.

- b. To make changes to the properties, first extract `amIdRepoService.properties` and, if necessary, the localized versions of this file from `openam-core-7.0.2.jar` to `WEB-INF/classes/`, where AM is deployed.

For example, if AM is deployed under `/path/to/tomcat/webapps/openam`, then you could run the following commands:

```
$ cd /path/to/tomcat/webapps/openam/WEB-INF/classes/
$ jar -xvf ../lib/openam-core-7.0.2.jar amIdRepoService.properties
inflated: amIdRepoService.properties
```

2. Create a properties file for your plugin so it is displayed in the AM console:
 - a. To create a custom identity store named `CustomRepo`, where the attributes will be displayed under the *Plug-in Configuration* tab, create a file named `CustomRepo.section.properties`, with the following contents:


```
#####
# Plug-in Configuration
#####
# LDAPv3 Repository Plug-in Class Name
pluginconfig=sunIdRepoClass
# Attribute Name Mapping
pluginconfig=sunIdRepoAttributeMapping
```

Tip

Ensure you include any properties you added to the subschema that need to be displayed in the UI. AM includes a number of `section.properties` in the `/path/to/tomcat/WEB-INF` folder that you can use for templates.

- b. To get your configuration properties in the UI, place the `CustomRepo.section.properties` into `WEB-INF/classes/`, where AM is deployed.
- c. Register your plugin using the `ssoadm` command after you have copied the files into place.

```
$ ssoadm \
add-sub-schema \
--adminid uid=amAdmin,ou=People,dc=openam,dc=forgerock,dc=org \
--password-file /tmp/pwd.txt \
--servicename sunIdentityRepositoryService \
--schematype Organization \
--filename customIdRepo.xml
```

3. Restart your AM instance.
4. Configure a new ID repo in AM using your plugin:
 - a. As an administrative user, in the AM console, navigate to Realms > *Realm Name* > Identity Stores.
 - b. In the Identity Stores page, select Add Identity Store, enter an ID, and select the type of identity store corresponding to your custom IdRepo plugin. You can now add values to any custom properties you configured to be visible in the UI.
 - c. Navigate to Realms > *Realm Name* > Identities, and create a new identity.
 - d. If your plugin supports authentication, then users are now able to authenticate using the `DataStore` module for the realm, by using a URL similar to the following:

```
https://openam.example.com:8443/openam/XUI/?realm=/myrealm&module=DataStore#login
```

Chapter 4

Policy and Application Stores

In addition to the identity store, AM allows you to configure data stores for different types of data.

Depending on the characteristics of the different data types, it can be beneficial to store them separately from other data types, for example to allow specific tuning of the indexes in the directory server.

AM supports configuration of data stores for the following data types:

- UMA data.

Provides storage for UMA-related data, such as resources, labels, and pending requests.

- Core Token Service (CTS) data.

Provides highly available storage for sessions and tokens used by AM.

- Policy data.

Provides storage for policy-related data, such as policies, policy sets, and resource types. Policy stores also store [delegated realm administration privileges](#).

Warning

If you change the policy data store, any existing policy sets and resource types will no longer be available to the realm where you made the change. Either recreate these items manually, or use [Amster](#) to export them from the old data store, then import them back after changing to the new data store.

- Application data.

Provides storage for application-related data, such as web and Java agent configuration, federation entities and configuration, and OAuth 2.0 clients definitions.

Tasks to Configure Policy and/or Application Stores

Task	Resources
Prepare the Store(s) You must install new DS servers for the store(s). Moreover, if you are configuring an AM instance that already has policy or	"Preparing Policy and Application Stores"

Task	Resources
application data on its configuration store, you may want to migrate that data to the new store(s).	
Configure the Store(s) Configure the newly-installed store(s) so that AM can use them.	"Setting Up Policy and Application Stores."

Preparing Policy and Application Stores

This section explains how to prepare a DS server for use as an external policy or application data store.

In a new AM deployment, you can install new instances of DS for policy and/or application data. See "Installing and Configuring Directory Services for Policy and/or Application Data".

In deployments where there is existing policy and/or application data, you may prefer to migrate this data from the existing location alongside the configuration data, into new separate DS instances. See "Migrating Policy and/or Application Data to a DS Instance"

Installing and Configuring Directory Services for Policy and/or Application Data

The following instructions show how to install and set up the DS server.

To Install and Configure Directory Services for Policy and/or Application Data

Directory Services 6.5 added support for *setup profiles* to greatly simplify initial configuration.

Using a setup profile will create the backend, schema, bind user, and indexes required for use with policy and/or application data. Note that policy and application data has the same schema requirements as configuration data.

1. To install DS using a setup profile, follow the steps in *DS for AM Configuration Data* in the *Directory Services 7 Installation Guide*.

Important

Ensure that you specify the same base DN value as your configuration store when using a setup profile to create a policy or application store. For example:

```
--set am-config/baseDn:"dc=example,dc=com"
```

Creating a separate DS backend for policies and/or applications in the configuration store is not supported.

2. Share the store certificate with the AM container to prepare for TLS/LDAPS. Application and policy stores should communicate over secure connections for security reasons.

DS 7 or later is configured to require secure connections by default; therefore, share its certificate with the AM container before continuing.

+ *Sharing the DS Certificate with AM*

1. Export the DS server certificate:

```
$ /path/to/openssl/bin/dskeymgr export-ca-cert \
--deploymentKey $DEPLOYMENT_KEY \
--deploymentKeyPassword password \
--alias ds-ca-cert \
--outputFile ds-ca-cert.pem
```

Note that `$DEPLOYMENT_KEY` is a Unix variable that contains the DS deployment key, so that it is not logged in the user's command history.

The default DS server certificate only has the hostname you supplied at setup time, and `localhost`, as the value of the `SubjectAlternativeName` attribute; however, certificate hostname validation is strict.

Ensure that the certificate matches the hostname (or the FQDN) of the DS server before continuing.

2. Import the DS certificate into the AM truststore:

```
$ keytool \
-importcert \
--alias ds-ca-cert \
--file ds-ca-cert.pem \
--keystore /path/to/openam/security/keystores/truststore
```

For more information on configuring AM's truststore, see "Preparing a Truststore" in the *Installation Guide*.

3. You have successfully installed and prepared the DS directory server for use as an external policy or application store.

You can use the instance created in these procedures for both policy and application data simultaneously. To have separate instances for policies and applications, repeat the previous step to create an additional external policy or application store.

Proceed to configuring AM to use the prepared DS directory servers as external policy or application stores, see "*Policy and Application Stores*".

The bind DN of the default AM service account used to authenticate to the external store has the form:

```
uid=am-config,ou=admins,Base DN
```

Migrating Policy and/or Application Data to a DS Instance

If you are upgrading existing AM instances to use external DS instances for policy and/or application data, you may want to migrate that existing data into the new instances. Migrating policy and/or application data to a separate store allows tuning and scaling of the more dynamic data, separately from the more static nature of configuration data.

This section covers the following high-level methods for migrating policy and/or application data to new DS instances:

- "Exporting and Importing Data in DS by using LDIF"
- "Backing Up and Restoring a DS Instance"

Exporting and Importing Data in DS by using LDIF

This section gives an overview of migrating data to new DS instances by using the `import-ldif` and `export-ldif` commands.

The top-level steps are as follows:

1. Install a DS instance to store the policy and/or application data, by following the steps in "Installing and Configuring Directory Services for Policy and/or Application Data".
2. Use the **export-ldif** command to create an LDIF file of the structure of the new DS instance.

For example:

```
$ /path/to/openssl/bin/export-ldif \  
--hostname 'config.example.com' \  
--port 4444 \  
--usePkcs12TrustStore /path/to/openssl/config/keystore \  
--trustStorePasswordFile /path/to/openssl/config/keystore.pin \  
--bindDN uid=admin \  
--bindPassword strongAdminPa55word \  
--backendId cfgStore \  
--ldifFile /tmp/Apps_and_or_Policies.ldif
```

3. Use the **export-ldif** command to append to the previous LDIF file your existing policy and/or application data.

Tip

The **export-ldif** command has a `--includeBranch` option to limit the data exported to just policy and/or application data.

See the following commands to export application data, policy data, or both:

Policy and Application Data

```
$ /path/to/openssl/bin/export-ldif \
--hostname 'config.example.com'\
--port 4444 \
--usePkcs12TrustStore /path/to/openssl/config/keystore \
--trustStorePasswordFile /path/to/openssl/config/keystore.pin \
--bindDN uid=admin \
--bindPassword strongAdminPa55word \
--includeBranch ou=sunamhiddenrealmdelegationpermissions,ou=services,dc=example,dc=com \
--includeBranch ou=AgentService,ou=services,dc=example,dc=com \
--includeBranch ou=sunEntitlementService,ou=services,dc=example,dc=com \
--includeBranch ou=sunEntitlementIndexes,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMCOTConfigService,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMIDFFMetadataService,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMSAML2MetadataService,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMWSFederationMetadataService,ou=services,dc=example,dc=com \
--backendId appData \
--appendToLdif \
--ldifFile /tmp/Apps_and_or_Policies.ldif
```

Policy Data

```
$ /path/to/openssl/bin/export-ldif \
--hostname 'config.example.com'\
--port 4444 \
--usePkcs12TrustStore /path/to/openssl/config/keystore \
--trustStorePasswordFile /path/to/openssl/config/keystore.pin \
--bindDN uid=admin \
--bindPassword strongAdminPa55word \
--includeBranch ou=sunEntitlementService,ou=services,dc=example,dc=com \
--includeBranch ou=sunEntitlementIndexes,ou=services,dc=example,dc=com \
--includeBranch ou=sunamhiddenrealmdelegationpermissions,ou=services,dc=example,dc=com \
--backendId appData \
--appendToLdif \
--ldifFile /tmp/Apps_and_or_Policies.ldif
```

Application Data

```
$ /path/to/openssl/bin/export-ldif \
--hostname 'config.example.com'\
--port 4444 \
--usePkcs12TrustStore /path/to/openssl/config/keystore \
--trustStorePasswordFile /path/to/openssl/config/keystore.pin \
--bindDN uid=admin \
--bindPassword strongAdminPa55word \
--includeBranch ou=AgentService,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMCOTConfigService,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMIDFFMetadataService,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMSAML2MetadataService,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMWSFederationMetadataService,ou=services,dc=example,dc=com \
--backendId appData \
--appendToLdif \
--ldifFile /tmp/Apps_and_or_Policies.ldif
```

Replace `dc=example,dc=com` in the commands above with your base DN value.

4. To also export policy and/or application data from subrealms, repeat the previous step, but altered to include the path to the subrealm in the included branches.

For example, to export data from a realm named `mySubRealm`, add `ou=services,o=mySubRealm`, into each of the included branches, as shown below:

```
$ /path/to/openssl/bin/export-ldif \
--hostname 'config.example.com' \
--port 4444 \
--usePkcs12TrustStore /path/to/openssl/config/keystore \
--trustStorePasswordFile /path/to/openssl/config/keystore.pin \
--bindDN uid=admin \
--bindPassword strongAdminPa55word \
--includeBranch
o=sunamhiddenrealmdelegationsservicepermissions,<replaceable>ou=services,o=mySubRealm,</
replaceable>ou=services,dc=example,dc=com \
--includeBranch ou=AgentService,<replaceable>ou=services,o=mySubRealm,</
replaceable>ou=services,dc=example,dc=com \
--includeBranch ou=sunEntitlementService,<replaceable>ou=services,o=mySubRealm,</
replaceable>ou=services,dc=example,dc=com \
--includeBranch ou=sunEntitlementIndexes,<replaceable>ou=services,o=mySubRealm,</
replaceable>ou=services,dc=example,dc=com \
--includeBranch ou=sunFMCOTConfigService,<replaceable>ou=services,o=mySubRealm,</
replaceable>ou=services,dc=example,dc=com \
--includeBranch ou=sunFMIDFFMetadataService,<replaceable>ou=services,o=mySubRealm,</
replaceable>ou=services,dc=example,dc=com \
--includeBranch ou=sunFMSAML2MetadataService,<replaceable>ou=services,o=mySubRealm,</
replaceable>ou=services,dc=example,dc=com \
--includeBranch ou=sunFMWSFederationMetadataService,<replaceable>ou=services,o=mySubRealm,</
replaceable>ou=services,dc=example,dc=com \
--backendId appData \
--appendToLdif \
--ldifFile /tmp/Apps_and_or_Policies.ldif
```

Repeat this step for each subrealm that contains application and/or policy data you want to transfer to an external store.

5. Edit the LDIF file to include the required top-level `ou=services` entry. For example, after the entry for `dn: dc=example,dc=com` add the following:

```
dn: ou=services,dc=example,dc=com
objectclass: top
objectclass: organizationalunit
objectclass: sunServiceComponent
ou: services
```

Note that the example above requires a blank line both before and after, and you must replace `dc=example,dc=com` with your base DN value.

The start of the resulting LDIF will resemble the following:

```
dn: dc=example,dc=com
objectClass: top
objectClass: untypedObject
dc: openam
aci: (targetattr="*)(version 3.0;acl "Allow CRUDQ operations";allow (search, read, write, add,
delete)(userdn = "ldap:///uid=am-config,ou=admins,dc=example,dc=com");)
aci: (targetcontrol="2.16.840.1.113730.3.4.3")(version 3.0;acl "Allow persistent search"; allow
(search, read)(userdn = "ldap:///uid=am-config,ou=admins,dc=example,dc=com");)
aci: (targetcontrol="1.2.840.113556.1.4.473")(version 3.0;acl "Allow server-side sorting"; allow
(read)(userdn = "ldap:///uid=am-config,ou=admins,dc=example,dc=com");)
entryUUID: 5beee9ea-1c31-3129-a8f9-c79da3102f26

dn: ou=services,dc=example,dc=com
objectClass: top
objectClass: organizationalunit
objectClass: sunServiceComponent
ou: services

dn: ou=admins,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
ou: admins
...
...
```

- Repeat the previous step to create the parent entries for any subrealms that you exported.

For example, before the `ou=admins,dc=example,dc=com` line, add the following for a subrealm named `mySubRealm`:

```
dn: o=mySubRealm,ou=services,dc=example,dc=com
objectClass: sunRealmService
objectClass: top
o: mySubRealm

dn: ou=services,o=mySubRealm,ou=services,dc=example,dc=com
objectClass: organizationalUnit
objectClass: sunServiceComponent
objectClass: top
ou: services
```

Note that the example above requires a blank line both before and after, and you must replace `dc=example,dc=com` with your base DN value.

The start of the resulting LDIF will resemble the following:

```
dn: dc=example,dc=com
objectClass: top
objectClass: untypedObject
dc: openam
aci: (targetattr="*)(version 3.0;acl "Allow CRUDQ operations";allow (search, read, write, add,
delete)(userdn = "ldap:///uid=am-config,ou=admins,dc=example,dc=com");)
aci: (targetcontrol="2.16.840.1.113730.3.4.3")(version 3.0;acl "Allow persistent search"; allow
(search, read)(userdn = "ldap:///uid=am-config,ou=admins,dc=example,dc=com");)
aci: (targetcontrol="1.2.840.113556.1.4.473")(version 3.0;acl "Allow server-side sorting"; allow
(read)(userdn = "ldap:///uid=am-config,ou=admins,dc=example,dc=com");)
entryUUID: 5beee9ea-1c31-3129-a8f9-c79da3102f26
```



```
dn: ou=services,dc=example,dc=com
objectclass: top
objectclass: organizationalunit
objectclass: sunServiceComponent
ou: services

dn: o=mySubRealm,ou=services,dc=example,dc=com
objectClass: sunRealmService
objectClass: top
o: mySubRealm

dn: ou=services,o=mySubRealm,ou=services,dc=example,dc=com
objectClass: organizationalUnit
objectClass: sunServiceComponent
objectClass: top
ou: services

dn: ou=admins,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
ou: admins
...
...
```

7. Use the **import-ldif** command to add the updated LDIF data to the new instance. For example:

```
$ /path/to/openssl/bin/import-ldif \
--hostname external.example.com \
--port 4444 \
--useSsl \
--usePkcs12TrustStore /path/to/openssl/config/keystore \
--trustStorePasswordFile /path/to/openssl/config/keystore.pin \
--bindDN uid=admin \
--bindPassword strongAdminPa55word \
--backendId cfgStore \
--ldifFile /tmp/Apps_and_or_Policies.ldif
```

8. Configure AM to use the new store for policy and/or application data, by following the steps in "Setting Up Policy and Application Stores".

For more information and the specific steps for importing and exporting LDIF data, see [Data Storage](#) in the *DS Configuration Guide*.

Backing Up and Restoring a DS Instance

This section gives an overview of migrating data to new DS instances by using the DS **backup** and **restore** commands.

The top-level steps are as follows:

1. Use the **backup** command to make a backup copy of the existing configuration store.
2. Install a DS instance to store the policy and/or application data, by following the steps in "Installing and Configuring Directory Services for Policy and/or Application Data".

Note that the DS instance **MUST** have the same base DN, and backend name, as the instance from which the backup was created.

3. Use the **restore** command to restore your policy and/or application data to the new store.
4. Configure AM to use the new store for policy and/or application data, by following the steps in "Setting Up Policy and Application Stores".

For more information and the specific steps for performing backup and restore operations in DS, see *Backup and Restore* in the *DS Maintenance Guide*, and *To Initialize a Replica From Backup Files* in the *DS Configuration Guide*.

Setting Up Policy and Application Stores.

This section covers setting up policy and application stores in AM.

Setting up a policy and/or application store in AM requires two procedures:

1. Configuring the connection between AM and the directory server.

See "To Connect AM to a Policy or Application Store".

2. Enabling a realm to use the newly configured directory server.

See "To Enable a Realm to use a Policy or Application Store".

To Connect AM to a Policy or Application Store

Perform the steps in this procedure to add a connection in AM to the policy or application store.

1. In the AM console, navigate to Configure > Global Services > External Data Stores.
2. On the Secondary Configurations tab, click Add a Secondary Configuration.
3. Complete the form as follows:
 - a. In the Name field, provide a name for the data store, for example, **myPolicyStore**
 - b. In the Host Urls field, enter one or more connection strings to the stores to use. The format for each connection string is **HOST:PORT**, for example **policies1.example.com:636**.

AM will use the first connection string in the list, unless the server is unreachable, in which case it will try the subsequent connection strings in the order they are defined.

- c. Enter the Bind DN and Bind Password of the service account AM uses to authenticate to the data store. The account needs sufficient privileges to read and write to the root suffix of the data store.

- d. Specify whether to use SSL and/or Start TLS connectivity to the data store by enabling the relevant option.
- e. Specify whether to access the data stores by using multiple directory instances in an affinity deployment, rather than a single master directory instance using an active/passive deployment.

If you enable this option, specify each of the directory server instances that form the affinity deployment in the Host Urls field.

4. To save your changes, click Create.

AM will attempt to contact the data store using the specified settings. If successful, AM will attempt to make the required schema and structure changes in the data store. If the service account specified by the Bind DN property does not have permissions to alter schema and structure, you will need to manually apply the required settings. See *"Preparing External Stores"* in the *Installation Guide*.

If AM was able to contact the data store using the specified settings, the connection is saved and made available for use as a policy or application store.

5. (Optional) To edit the connection settings to a store, perform the following steps:
 - a. On the Secondary Configuration tab, click the name of the data store.
 - b. Edit the configuration as required, and then click Save Changes.
6. (Optional) Repeat the steps above to add any additional policy or application stores.

You can now configure AM to use the new store. See *"To Enable a Realm to use a Policy or Application Store"*.

To Enable a Realm to use a Policy or Application Store

Perform the following steps to configure a realm in AM to use a policy or application store.

Important

Changing the policy or application store will cause any existing policies or applications to become unavailable to the realm.

Either recreate the policies or applications manually, or use Amster to export the existing instances, then import them back after changing the stores.

1. In the AM console, navigate to Realms > *Realm name* > Services.
2. Configure the External Data Stores service in the realm:

- If the External Data Stores service has not yet been added to the realm, click Add a Service, and then select External Data Stores.
 - If the External Data Stores service has already been added to the realm, click External Data Stores to edit the configuration.
3. On the External Data Stores page, select the name of the store to use as the Policy Data Store and/or Application Data Store, and then click Save Changes.

Note

If you choose the **Default Datastore** option for either property, AM will resort to using the configuration data store that was specified during the installation of AM.

Changes take effect immediately. New policies or applications are created in the relevant data store, if configured.

To Remove a Policy or Application Store

To be able to remove a policy or application store from AM it cannot be in use by any realm.

Perform the following steps to remove an policy or application store from a realm in AM, and delete a store from the AM instance.

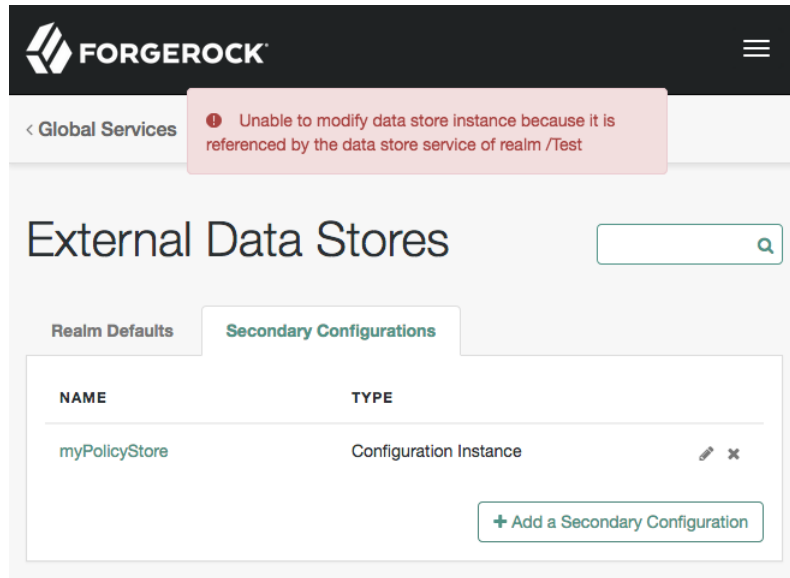
1. For each realm that is using the store, in the administration console, navigate to Realms > *Realm Name* > Services > External Data Stores, and change each of the drop-downs to either **Default Datastore**, or an alternative data store.

Save your changes.

2. Navigate to Configure > Global Services > External Data Stores > Secondary Configurations. Click the name of the store to remove, and click the delete icon.

If the data store is still in use, you will see an error message as follows:

Unable to modify data store instance because it is referenced by the data store service of realm */Realm Name*



If you receive the error, repeat the first step to remove the unwanted store from the listed realm, and then repeat this step.

Chapter 5

Load Balancers

Load balancer configuration requirements differ depending on where you configure AM to store sessions and authentication sessions, as shown in the following table:

Load Balancing Requirements by Session Storage Location

Storage Location	Load Balancing Requirement	Comments
CTS token store	None. Session stickiness recommended for performance	<p>Although the CTS token store is the authoritative source for CTS-based sessions and authentication sessions, AM servers cache the session or authentication session when:</p> <ul style="list-style-type: none"> • Authenticating a user • Satisfying a session request <p>An AM site configured to use CTS-based sessions achieves the best performance when the server that originally authenticated a user continually manages that user's session, since it does not need to retrieve it from the CTS token store.</p> <p>In the same way, an AM realm configured to use CTS-based authentication sessions achieves the best performance when the same server manages every request for the same authentication flow.</p>
Client	None. Session stickiness recommended for performance	<p>Although the user's session or authentication session resides in a JWT stored on the client which is passed to AM server along with the request, client-based sessions should be signed and/or encrypted for security reasons. Because decrypting and verifying the session on each request may be an expensive operation depending on the algorithms configured, AM caches the decrypt sequence in memory to improve performance.</p> <p>Therefore, an AM site configured to use client-based sessions achieves best performance when the same server manages every request for the same session or authentication session.</p>
In AM's memory ^a	Session stickiness required	<p>Deployments where AM stores authentication sessions in memory require sticky load balancing to route all requests pertaining to a particular authentication flow to the same AM server. If a request reaches a different AM server, the authentication flow will start anew.</p>

^aAuthentication sessions only.

For more information about AM session and authentication sessions, see *"Introducing Sessions"* in the *Sessions Guide*.

Session storage location can be heterogeneous within the same AM deployment to suit the requirements of each of your realms. If your deployment uses a substantial number of CTS-based sessions, follow the recommendations for deployments configured for CTS-based sessions.

Configuring Site Sticky Load Balancing

Configure AM for sticky load balancing as follows:

To Configure Site Sticky Load Balancing

1. Navigate to Deployment > Sites.
2. Ensure you have a site created and that your servers are part of it. For more information, see *"To Configure a Site With the First Server"* in the *Installation Guide*.
3. Ensure that the `amlbcookie` cookie has a unique value for each AM server:
 - a. Navigate to Deployment > Servers > *Server Name* > Advanced and review the value of the `com.ipplanet.am.lbcookie.value` property. By default, the cookie value is set to the AM server ID.

Keep the value of the `amlbcookie` cookie set to the AM server ID to improve server performance when using Web Agents.

If you have replaced the value of the this property and you need to match the AM server URLs with their corresponding server IDs, query the `global-config/servers` endpoint. For example:

```
$ curl \
--header 'Accept: application/json' \
--header "iPlanetDirectoryPro: AQIC5...NDU1*" \
--header "Accept-API-Version: resource=1.0, protocol=2.1" \
'https://openam.example.com:8443/openam/json/global-config/servers?_queryFilter=true'
{
  "result": [
    {
      "_id": "01",
      "_rev": "-1541617246",
      "siteName": null,
      "url": "https://openam.example.com:8443/openam"
    }
  ],
  "resultCount": 1,
  "totalPagedResults": -1,
  "totalPagedResultsPolicy": "NONE"
}
```

In the example above, the server ID for server `https://openam.example.com:8443/openam` is `01`.

Changes take effect only after you restart the AM server.

- b. Restart the AM server. You can then check the cookie value by logging in to the AM console, and examining the `amlbcookie` cookie in your browser.

Tip

To change the name of the sticky load balancing cookie, see "Changing the Name of the Sticky Load Balancing Cookie" in the *Security Guide*.

4. Repeat the previous steps for each of the AM servers that should be part of the site.
5. Configure your load balancer to perform sticky load balancing based on the `amlbcookie` value.

In other words, the load balancer layer must keep track of which `amlbcookie` cookie value corresponds to which AM server.

When the load balancer receives a request, it inspects the value of the `amlbcookie` cookie, and then forwards the request to the corresponding AM server.

Load Balancer Offloading

When traffic to and from the load balancer is protected with HTTPS, you must terminate the SSL connection on the load balancer. Decrypting the traffic in the load balancer makes it possible to use cookie-based session stickiness.

You then either re-encrypt the traffic from the load balancer to AM, or make connections from the load balancer to AM over HTTP.

If you configure the load balancer in passthrough mode instead, session stickiness would not be possible.

Request Forwarding Caveats

Sticky load balancing based on the value of the `amlbcookie` cookie does not guarantee request forwarding to the corresponding AM server in all cases. For example, ForgeRock® Common REST API calls do not typically use cookies. Therefore, load balancers are not able to route these calls to the AM server on which a user's session is cached.

The AM server that does not hold the user's session in cache must locate the user's session by retrieving it from the CTS token store.

Handling HTTP Request Headers

HTTP requests can include information needed for access management, such as the client IP address used for adaptive risk-based authentication.

Configure your load balancer or proxy to pass the information to AM by using request headers. For example, the load balancer or proxy can send the client IP address by using the `X-Forwarded-For` HTTP request header.

Also configure AM to consume and to forward the headers as necessary. For example, to configure AM to look for the client IP address in the `X-Forwarded-For` request header, set the advanced configuration property `com.sun.identity.authentication.client.ipAddressHeader` to `X-Forwarded-For` under Deployment > Servers > *Server Name* > Advanced.

In a site configuration where one AM server can forward requests to another AM server, you can retain the header by adding it to the advanced configuration property `openam.retained.http.request.headers`. If `X-Forwarded-For` is the only additional header to retain, set `openam.retained.http.request.headers` to `X-DSAMEVersion,X-Forwarded-For`, for example.

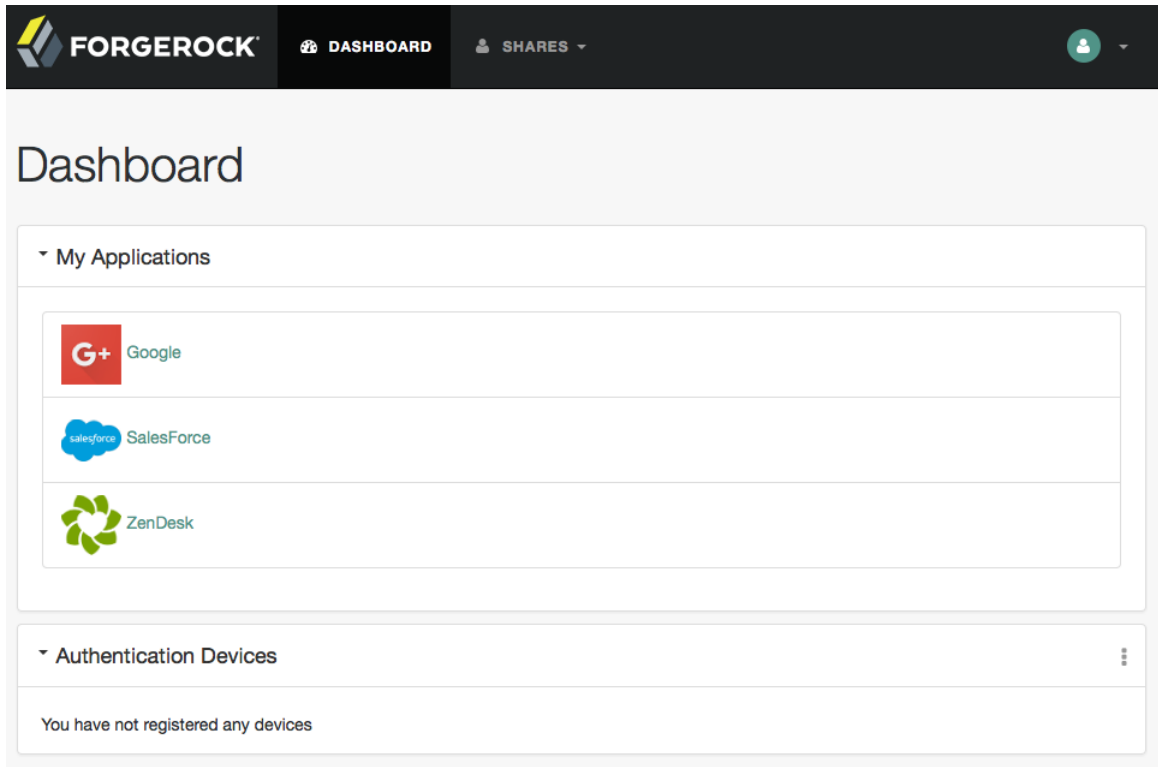
Configure these properties under Deployment > Servers > *Server Name* > Advanced.

Chapter 6

Dashboards

The Dashboard Service provides the end user with an interface to access applications secured by AM, both cloud-based applications like Salesforce and internal applications protected by web or Java agents. The Dashboard Service uses SSO to log in 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.

User Dashboard Screen



The Dashboard Service is meant to give users a single place to access their applications. Keep in mind that this does not limit user access, only what appears on the user dashboard.

There are three stages to setting up the Dashboard Service:

- Setup the Dashboard Service and add applications.
- Configure the service for the realms.
- Assign users applications so that they appear on the users' dashboards. This can be done manually or through a provisioning solution.

Once the Dashboard Service is configured for a user, the user can access their dashboard after logging in through the XUI under `/XUI/?realm=/#dashboard/`.

When making a request to the UI, specify the realm or realm alias as the value of a `realm` parameter in the query string, or the DNS alias in the domain component of the URL. If you do not use a realm alias, then you must specify the entire hierarchy of the realm, starting at the Top Level Realm. For example `https://openam.example.com:8443/openam/XUI/?realm=/customers/europe#login/`.

For example, the full URL depending on the deployment might be at `https://openam.example.com:8443/openam/XUI/?realm=/myrealm#dashboard/`.

Implementing the Dashboard Service

Making some applications universally available ensures that all users have the same basic applications. However, some of your applications should be protected from the majority of your users. You will need to single out which users will include the application on their dashboard.

There are three default applications in the Dashboard Service: Google, Salesforce, and Zendesk.

- To set up the Dashboard Service, see "To Set Up the Dashboard Service and Add Applications".
- To configure the Dashboard Service, see "To Configure the Dashboard Service for a Realm".
- To add an application to a user's dashboard, see "To Enable An Application for a User".
- To remove an application from a user's dashboard, see "To Remove User Access to an Application".

To Set Up the Dashboard Service and Add Applications

You can add applications to the Dashboard Service with the following steps. All fields except the dashboard class name and ICF Identifier are required for the application to work properly from the dashboard:

1. Log in to the AM console as AM Administrator, `amAdmin`.
2. Navigate to Configure > Global Services > Dashboard > Secondary Configurations, and then click Add a Secondary Configuration to add an application to the Dashboard Service.
3. Provide a unique name for the application.

4. Add a Dashboard Class Name that identifies how the end user will access the app, such as `SAML2ApplicationClass` for a SAML v2.0 application.
5. Add a Dashboard Name for the application.
6. Add a Dashboard Display Name. This name is what the end user will see, such as Google.
7. Add the Dashboard Icon you would like the end user to see for the application. Either use a fully qualified URL or an appropriate relative URL so that the icon is rendered properly on the user dashboard.
8. Add the Dashboard Login URL to point to the location the end user will go to once they click on the icon.
9. Leave the ICF Identifier blank.
10. Click Add when you are done.

To Configure the Dashboard Service for a Realm

You must configure the Dashboard Service and add applications to a realm before you can access them. The following instructions show you how to add an application to a single realm. Before you begin, make sure you have the name of the application as it appears on the Secondary Configuration Instance table under Configure > Global Services > Dashboard:

1. Select Realms > *Realm Name* > Services, and then click Add a Service.
2. Select the Dashboard service, and then click Create.
3. Add or remove the applications you would like to appear on the Dashboard service for the realm.
4. Click Save Changes when you are done.

To Enable An Application for a User

Use the following steps to enable access to an application from the user's dashboard:

1. Select Realms > *Realm Name* > Identities and click the user identifier to edit the user's profile.
2. Under Services, click Dashboard.
3. Add the application beside the user name under the user's Assigned Dashboard list.
4. Click Save.

To Remove User Access to an Application

Removing user access to an application does not delete the user's identity profile. The following steps walk you through removing an application from a user's dashboard:

1. Select Realms > *Realm Name* > Identities and click the user identifier to edit the user's profile.
2. Under Services, click Dashboard.
3. Delete the application beside the user name under the user's Assigned Dashboard list.
4. Click Save.

Displaying Dashboard Applications

AM lets administrators configure online applications to display applications on user Dashboards. You can use the exposed REST API to display information about the online applications.

`/dashboard/assigned`

This endpoint retrieves the list of applications assigned to the authenticated user.

```
$ curl \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
--header "Accept-API-Version: resource=1.0" \
https://openam.example.com:8443/openam/json/realms/root/dashboard/assigned
{
  "google":{
    "dashboardIcon":[
      "Google.gif"
    ],
    "dashboardName":[
      "Google"
    ],
    "dashboardLogin":[
      "http://www.google.com"
    ],
    "ICFIdentifier":[
      ""
    ],
    "dashboardDisplayName":[
      "Google"
    ],
    "dashboardClassName":[
      "SAML2ApplicationClass"
    ]
  }
}
```

`/dashboard/available`

This endpoint retrieves the list of applications available in the authenticated user's realm. The example is based on two of the default Dashboard applications: Google and Salesforce.

```
$ curl \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
--header "Accept-API-Version: resource=1.0" \
https://openam.example.com:8443/openam/json/realms/root/dashboard/available
{
  "google":{
```

```

    "dashboardIcon": [
      "Google.gif"
    ],
    "dashboardName": [
      "Google"
    ],
    "dashboardLogin": [
      "http://www.google.com"
    ],
    "ICFIdentifier": [
      ""
    ],
    "dashboardDisplayName": [
      "Google"
    ],
    "dashboardClassName": [
      "SAML2ApplicationClass"
    ]
  },
  "salesforce": {
    "dashboardIcon": [
      "salesforce.gif"
    ],
    "dashboardName": [
      "Salesforce"
    ],
    "dashboardLogin": [
      "http://salesforce.com"
    ],
    "ICFIdentifier": [
      ""
    ],
    "dashboardDisplayName": [
      "Salesforce"
    ],
    "dashboardClassName": [
      "SAML2ApplicationClass"
    ]
  }
}

```

/dashboard/defined

This endpoint retrieves the list of all applications available defined for the AM Dashboard service. The example is based on the three default Dashboard applications: Google, Salesforce, and Zendesk.

```

$ curl \
--header "iplanetDirectoryPro: AQIC5w...2NzEz*" \
--header "Accept-API-Version: resource=1.0" \
https://openam.example.com:8443/openam/json/realms/root/dashboard/defined
{
  "google": {
    "dashboardIcon": [
      "Google.gif"
    ],
    "dashboardName": [
      "Google"
    ],

```

```

    "dashboardLogin": [
      "http://www.google.com"
    ],
    "ICFIdentifier": [
      "idm magic 34"
    ],
    "dashboardDisplayName": [
      "Google"
    ],
    "dashboardClassName": [
      "SAML2ApplicationClass"
    ]
  },
  "salesforce": {
    "dashboardIcon": [
      "salesforce.gif"
    ],
    "dashboardName": [
      "SalesForce"
    ],
    "dashboardLogin": [
      "http://www.salesforce.com"
    ],
    "ICFIdentifier": [
      "idm magic 12"
    ],
    "dashboardDisplayName": [
      "Salesforce"
    ],
    "dashboardClassName": [
      "SAML2ApplicationClass"
    ]
  },
  "zendesk": {
    "dashboardIcon": [
      "ZenDesk.gif"
    ],
    "dashboardName": [
      "ZenDesk"
    ],
    "dashboardLogin": [
      "http://www.ZenDesk.com"
    ],
    "ICFIdentifier": [
      "idm magic 56"
    ],
    "dashboardDisplayName": [
      "ZenDesk"
    ],
    "dashboardClassName": [
      "SAML2ApplicationClass"
    ]
  }
}

```

If your application runs in a user-agent such as a browser, you can rely on AM to handle authentication.

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 identities 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 agent profile configuration information, typically created to delegate agent profile creation to the user installing a web or Java agent.
Agent authenticator	Entity with read-only access to multiple agent profiles defined in the same realm; allows an agent to read web service profiles.
Application	<p>In general terms, a service exposing protected resources.</p> <p>In the context of AM policies, the application is a template that constrains the policies that govern access to protected resources. An application can have zero or more policies.</p>
Application type	<p>Application types act as templates for creating policy applications.</p> <p>Application types define a preset list of actions and functional logic, such as policy lookup and resource comparator logic.</p>

	Application types also define the internal normalization, indexing logic, and comparator logic for applications.
Attribute-based access control (ABAC)	Access control that is based on attributes of a user, such as how old a user is or whether the user is a paying customer.
Authentication	The act of confirming the identity of a principal.
Authentication chaining	A series of authentication modules configured together which a principal must negotiate as configured in order to authenticate successfully.
Authentication level	Positive integer associated with an authentication module, usually used to require success with more stringent authentication measures when requesting resources requiring special protection.
Authentication module	AM 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. AM can play this role in the OAuth 2.0 authorization framework.
Auto-federation	Arrangement to federate a principal's identity automatically based on a common attribute value shared across the principal's profiles at different providers.
Bulk federation	Batch job permanently federating user profiles between a service provider and an identity provider based on a list of matched user identifiers that exist on both providers.
Circle of trust	Group of providers, including at least one identity provider, who have agreed to trust each other to participate in a SAML v2.0 provider federation.
Client	In OAuth 2.0, requests protected web resources on behalf of the resource owner given the owner's authorization. AM can play this role in the OAuth 2.0 authorization framework.
Client-based OAuth 2.0 tokens	After a successful OAuth 2.0 grant flow, AM returns a token to the client. This differs from CTS-based OAuth 2.0 tokens, where AM returns a <i>reference</i> to token to the client.
Client-based sessions	AM sessions for which AM returns session state to the client after each request, and require it to be passed in with the subsequent

request. For browser-based clients, AM sets a cookie in the browser that contains the session information.

For browser-based clients, AM sets a cookie in the browser that contains the session state. When the browser transmits the cookie back to AM, AM decodes the session state from the cookie.

Conditions

Defined as part of policies, these determine the circumstances under which which a policy applies.

Environmental conditions reflect circumstances like the client IP address, time of day, how the subject authenticated, or the authentication level achieved.

Subject conditions reflect characteristics of the subject like whether the subject authenticated, the identity of the subject, or claims in the subject's JWT.

Configuration datastore

LDAP directory service holding AM configuration data.

Cross-domain single sign-on (CDSSO)

AM capability allowing single sign-on across different DNS domains.

CTS-based OAuth 2.0 tokens

After a successful OAuth 2.0 grant flow, AM returns a *reference* to the token to the client, rather than the token itself. This differs from [client-based OAuth 2.0 tokens](#), where AM returns the entire token to the client.

CTS-based sessions

AM [sessions](#) that reside in the Core Token Service's token store. CTS-based sessions might also be cached in memory on one or more AM servers. AM tracks these sessions in order to handle events like logout and timeout, to permit session constraints, and to notify applications involved in SSO when a session ends.

Delegation

Granting users administrative privileges with AM.

Entitlement

Decision that defines which resource names can and cannot be accessed for a given identity 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 AM.

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 AM on the service provider side; AM lets you create Java Fedlets.
Hot swappable	Refers to configuration properties for which changes can take effect without restarting the container where AM 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 agent	Java web application installed in a web container that acts as a policy enforcement point, 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	Java, web, or custom agent that intercepts requests for resources, directs principals to AM for authentication, and enforces policy decisions from AM.
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 Subject successfully authenticates, AM associates the Subject with the Principal .
Privilege	In the context of delegated administration, a set of administrative tasks that can be performed by specified identities in a given realm.
Provider federation	Agreement among providers to participate in a circle of trust.
Realm	<p>AM unit for organizing configuration and identity information.</p> <p>Realms can be used for example when different parts of an organization have different applications and identity stores, and when different organizations use the same AM deployment.</p> <p>Administrators can delegate realm administration. The administrator assigns administrative privileges to users, allowing them to perform administrative tasks within the realm.</p>
Resource	<p>Something a user can access over the network such as a web page.</p> <p>Defined as part of policies, these can include wildcards in order to match multiple actual resources.</p>
Resource owner	In OAuth 2.0, entity who can authorize access to protected web resources, such as an end user.
Resource server	In OAuth 2.0, server hosting protected web resources, capable of handling access tokens to respond to requests for such resources.
Response attributes	Defined as part of policies, these allow AM 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).
Authentication Session	The interval while the user or entity is authenticating to AM.
Session	The interval that starts after the user has authenticated and ends when the user logs out, or when their session is terminated. For browser-based clients, AM manages user sessions across one or more applications by setting a session cookie. See also CTS-based sessions and Client-based sessions.

Session high availability	Capability that lets any AM server in a clustered deployment access shared, persistent information about users' sessions from the CTS token store. The user does not need to log in again unless the entire deployment goes down.
Session token	Unique identifier issued by AM after successful authentication. For a CTS-based sessions , the session token is used to track a principal's session.
Single log out (SLO)	Capability allowing a principal to end a session once, thereby ending her session across multiple applications.
Single sign-on (SSO)	Capability allowing a principal to authenticate once and gain access to multiple applications without authenticating again.
Site	<p>Group of AM servers configured the same way, accessed through a load balancer layer. The load balancer handles failover to provide service-level availability.</p> <p>The load balancer can also be used to protect AM services.</p>
Standard metadata	Standard federation configuration information that you can share with other access management software.
Stateless Service	<p>Stateless services do not store any data locally to the service. When the service requires data to perform any action, it requests it from a data store. For example, a stateless authentication service stores session state for logged-in users in a database. This way, any server in the deployment can recover the session from the database and service requests for any user.</p> <p>All AM services are stateless unless otherwise specified. See also Client-based sessions and CTS-based sessions.</p>
Subject	<p>Entity that requests access to a resource</p> <p>When an identity successfully authenticates, AM associates the identity with the Principal that distinguishes it from other identities. An identity can be associated with multiple principals.</p>
Identity store	Data storage service holding principals' profiles; underlying storage can be an LDAP directory service or a custom IdRepo implementation.
Web Agent	Native library installed in a web server that acts as a policy enforcement point with policies based on web page URLs.