



Upgrade 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 © 2011-2021 ForgeRock AS.

Abstract

This guide shows you how to upgrade 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, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

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

DejaVu Fonts

Bitstream Vera Fonts Copyright

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

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

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

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

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

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

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

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

Arev Fonts Copyright

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

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

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

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

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

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

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

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

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. Supported Upgrade Paths	1
2. Planning the Upgrade	3
Routing Around Servers During Downtime	3
Backing Up the Deployment	4
Reviewing REST API Versions Before Upgrading	4
Reviewing Directory Services Certificates Before Upgrading	5
Check your RADIUS Client Secret	5
Customizing Before Upgrading	5
Planning for Rollback	6
Upgrading on a Test Environment First	7
3. Upgrading AM Instances	8
4. Upgrading Components and Services	17
Configuring the User Profile Whitelist	19
Configuring Secret Stores After Upgrade	20
Upgrading Device Recovery Codes	23
Upgrading JDBC Audit Event Handlers	24
5. Migrating Legacy Instances	25
Glossary	26

Overview

The Upgrade Guide covers common aspects of upgrading an AM deployment, whether you are moving to a new maintenance release, upgrading to a new major release, or migrating from a legacy release to a newer AM release.

Release levels, and how much change to expect in a maintenance, minor, or major release, are defined in Release levels. Release levels are identified by version number.




Tip

AM supports several Web Agents and Java Agents versions, so most of the time it is not necessary to upgrade your agents at the same time you upgrade AM.

For a compatibility matrix between the agents and AM, see the following Knowledge Base article.

For information about upgrading Web Agents or Java Agents, see the *ForgeRock Web Agents User Guide* or the *ForgeRock Java Agents User Guide*.

Quick Start

 <p>Before you Upgrade...</p> <p>Review the tasks you need to complete before upgrading AM.</p>	 <p>Upgrading Servers</p> <p>Learn, step by step, how to upgrade the AM core services.</p>	 <p>Upgrading Components</p> <p>Review a list of components and services that you might need to upgrade or reconfigure.</p>
--	---	--

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

Supported Upgrade Paths

The following table contains information about the supported upgrade paths to AM 7.0.2:

Upgrade Paths

Version	Upgrade Supported?
AM 5.x	✓
AM 6.x	✓
AM 7.x	✓

For more information, see [Checking your product versions are supported in the ForgeRock Knowledge Base](#).

If you are upgrading from an *unsupported* version of AM to a later version, you must first upgrade to a supported version. In some cases, you may need to upgrade again depending on the upgrade path.

Caution

The embedded DS server is *not supported for production* in AM 7. If you have an existing site configured with embedded DS, you must migrate it to an external DS store before upgrading to AM 7.

The embedded DS is deprecated in 7 and will be removed in a future release.

+ [How Do I Know if my Deployment Uses the Embedded DS?](#)

- (AM 6 or earlier) Go to Deployment > Servers > *Server Name* > Advanced, and check the value of the `com.sun.identity.sm.sms_object_class_name` advanced property.

If the value is `com.sun.identity.sm ldap.SMSEmbeddedLdapObject`, the server is an evaluation instance of AM, and is using an embedded DS instance as the configuration store.

- In the server where AM is installed, check if the `opens` directory exists under the `/path/to/openam` directory.

You may have migrated it to an external directory and not deleted the directory, though. Check the files in the `opens/logs` directory to determine if the embedded DS is running.

- Go to Deployment > Servers > *Server Name* > Directory Configuration > Server, and check the value of the host name column.

When using an external configuration store, the AM instances point to the FQDN of the load balancer in front of the DS cluster, or to the FQDN of the DS affinity deployment.

When using an embedded configuration store, each AM instance points to its own hostname, since the embedded DS is stored alongside the AM instance.

Use either of the following methods to migrate an embedded data store to an external store before attempting to upgrade to AM 7:

- Migrate data to an external instance of DS by using LDIF files.

Follow the steps in [How do I migrate from an embedded to external DS/OpenDJ in AM/OpenAM \(All versions\)?](#) in the *ForgeRock Knowledge Base*.

- Add a new external DS instance and replicate data from the embedded instance.

Follow the steps in [Add a New Replica Before Upgrade](#) in the *DS Upgrade Guide*.

Chapter 2

Planning the Upgrade

How much you must do to upgrade AM software depends on the magnitude of the differences between the version you currently use and the new version:

- Maintenance releases have a limited effect on current functionality but contain necessary bug and security fixes. You should keep up to date with maintenance releases as the fixes are important, and the risk of affecting service is minimal.
- When upgrading to a new major or minor release, always plan and test the changes before carrying out the upgrade in production. Make sure you read the release notes for intervening versions with care, identifying any changes likely to affect your deployment, and then plan accordingly.

Review the following best practices before you upgrade AM:

- "Routing Around Servers During Downtime"
- "Backing Up the Deployment"
- "Reviewing REST API Versions Before Upgrading"
- "Reviewing Directory Services Certificates Before Upgrading"
- "Customizing Before Upgrading"
- "Planning for Rollback"
- "Upgrading on a Test Environment First"

Routing Around Servers During Downtime

Upgrading servers takes at least one of your AM sites down while the server configurations are being brought up to date with the newer version. Plan for this site to be down, routing client applications to another site until the upgrade process is complete and you have validated the result. Make sure client application owners are well aware of the change, and let them know what to expect.

If you only have a single AM site, make sure the downtime happens in a low usage window, and make sure you let client application owners plan accordingly.

During an upgrade you must restrict access to the AM console: The Upgrade Wizard page does not require authorization; any user with access to the AM console immediately after you deploy the new WAR file can therefore initiate the upgrade process.

Backing Up the Deployment

Always back up your deployment before you upgrade, as you must be able to roll back should something go wrong during the upgrade process.

- Backing up your configuration as described in "*Backing Up Configurations*" in the *Maintenance Guide* is good for production environments.
- In preparation for upgrading AM servers and their configurations, also take LDIF backups of the configuration store data in the directory servers. If possible, stop servers before upgrading and take a file system backup of the deployed servers and also of their configuration directories as well. This can make it easier to roll back from a failed upgrade.

For example, if you deploy AM server in Apache Tomcat under `/openam`, you might take a file system backup of the following directories for each AM server.

- `/path/to/tomcat/webapps/openam/`
- `~/openam/`
- `~/openamcfg/`
- When upgrading tools, keep copies of any tools scripts that you have edited for your deployment. Also back up any trust stores used to connect securely.
- Record any custom advanced server properties configured under Configure > Server Defaults > Advanced or Deployment > Servers > *Server Name* > Advanced in the AM Admin UI. These properties are lost during the upgrade and will need to be added again after the upgrade is complete.

Reviewing REST API Versions Before Upgrading

Upgrading AM may update the default API version of several AM endpoints. After an upgrade, your applications may experience issues connecting to endpoints if they do not specify API version information in REST calls.

By default, an upgraded AM instance responds to REST calls that do not specify version information with the oldest version available for the endpoint. However, the oldest supported version may not be the one required by the application, as API versions become deprecated or unsupported.

To avoid version conflicts between application calls and REST endpoint APIs, consider specifying the protocol and resource version required by the application in the `Accept-API-Version` header when making requests to REST endpoints. For more information, see "Specifying REST API Versions" in the *Getting Started with REST*.

Important

Starting in version 6, AM includes a CSRF protection filter that is enabled by default. REST requests other than GET, HEAD, and OPTIONS made to endpoints under the `json/` root will return 403 Forbidden messages unless one of the `X-Requested-With` or `Accept-API-Version` headers are added to it.

For more information, see "Cross-Site Request Forgery (CSRF) Protection" in the *Security Guide*.

You can configure AM's default REST API behavior. For more information, see "Configuring the Default REST API Version" in the *Getting Started with REST*.

Reviewing Directory Services Certificates Before Upgrading

Before upgrading, review the certificates used to establish secure connections between AM and the DS stores.

If the FQDN value from the `subject` field of a non-wildcard certificate does not match the FQDN obtained from DNS for the DS instance, AM will not be able to establish a secure connection with DS. Additionally, if the DS instance responds to multiple FQDNs, they must be specified in the certificate as well.

This step is critical for the configuration store. If AM cannot establish communication with the configuration store, it will fail to start up.

For more information about setting up DS server certificates, see *Setting Up Server Certificates* in the *ForgeRock Directory Services Security Guide*.

Check your RADIUS Client Secret

If you have configured a RADIUS server service, note that the format of the RADIUS client secret changed in AM versions 6.5.4 and 7.0.2.

Before you upgrade to any of these versions (6.5.4 or 7.0.2), you must do one of the following:

- Back up your current RADIUS client secret and use this secret to *reset* the RADIUS client secret after upgrade.
- After upgrade, create a *new* client secret and update your RADIUS clients with this new secret.

Customizing Before Upgrading

Prepare a `.war` file that contains any customizations you require.

Customizations include any changes you have made to the AM server installation, such as the following:

- Custom plugin and extensions, for example:
 - Custom authentication modules.
 - Custom authentication nodes.
 - Post-authentication plugins.
 - Custom SAML v2.0 attribute mappers.
 - Custom OAuth 2.0 scope validators.

Important

New functionality often changes the samples provided with AM.

Do not copy custom plugins or extensions from a previous version of AM to the `.war` file.

You must customize the samples of the version you are upgrading to before adding them to the `.war` file. For example, download the custom scope validator sample of the version you are upgrading to, customize it, recompile it, and then add the resulting `.jar` file to the `.war` file.

Failure to use the new version's samples as the base for your customizations may result in unexpected behavior.

- Customized JSPs, redesigned login or service pages, additional CSS and visual content, and modified authentication module callback files.
- Any changes to AM classes or APIs.

Tip

We recommend that you recompile any custom implementations you have created with each release of AM, as the method signature or imports for supported and evolving APIs can change in each version.

- Any changes or additional Java class libraries (such as `.jar` files in `WEB-INF/lib`).

Planning for Rollback

Sometimes even a well-planned upgrade operation fails to go smoothly. In such cases, you need a plan to roll back smoothly to the pre-upgrade version.

For AM servers, you can roll back by restoring from file system backup. If you use an external configuration directory service, restore the old configuration from LDIF before restarting the old servers. For more information, see "*Backing Up Configurations*" in the *Maintenance Guide*.

Upgrading on a Test Environment First

Always try upgrading AM in a test environment before applying the upgrade in your production environment.

This will help you gauge the amount of work required without affecting your production environment, and will help smooth out unforeseen problems.

Chapter 3

Upgrading AM Instances

Upgrading AM is a process that consists of upgrading the AM instance or instances in your site and, depending on the version you are upgrading from, updating the configuration of several AM features.

When possible, the upgrade process makes the appropriate changes on AM configuration on your behalf. However, sometimes you will need to perform additional configuration based on your environment needs. *It is imperative that you read the Release Notes and understand the changes introduced in each version before upgrading AM.*

Upgrade AM using the Upgrade wizard, which appears when you replace a deployed AM `.war` file with a newer version and navigate to the deployment URL. The Upgrade wizard brings the AM configuration, including the version number, up to date with the new version.

Tip

The CLI counterpart of the Upgrade wizard is **openam-upgrade-tool-14.1.3.11.jar**, which you install as described in "To Set Up the Configuration Tools" in the *Installation Guide*.

Perform the steps in the following procedure to upgrade AM:

To Upgrade From a Supported Version

Follow these steps to upgrade a site of instances. For information on the versions that are supported for upgrade, see "*Supported Upgrade Paths*".

Caution

- Do *not* perform an upgrade by deploying the new version and then importing an existing configuration by running the **ssoadm import-svc-config** command.
- The embedded DS server is *not supported for production* in AM 7. If you have an existing site configured with embedded DS, you must migrate it to an external DS store before upgrading to AM 7.

The embedded DS is deprecated in 7 and will be removed in a future release.

+ *How Do I Know if my Deployment Uses the Embedded DS?*

- (AM 6 or earlier) Go to Deployment > Servers > *Server Name* > Advanced, and check the value of the `com.sun.identity.sm.sms_object_class_name` advanced property.

If the value is `com.sun.identity.sm.ldap.SMSEmbeddedLdapObject`, the server is an evaluation instance of AM, and is using an embedded DS instance as the configuration store.

- In the server where AM is installed, check if the `opends` directory exists under the `/path/to/openam` directory.

You may have migrated it to an external directory and not deleted the directory, though. Check the files in the `opends/logs` directory to determine if the embedded DS is running.

- Go to Deployment > Servers > *Server Name* > Directory Configuration > Server, and check the value of the host name column.

When using an external configuration store, the AM instances point to the FQDN of the load balancer in front of the DS cluster, or to the FQDN of the DS affinity deployment.

When using an embedded configuration store, each AM instance points to its own hostname, since the embedded DS is stored alongside the AM instance.

Use either of the following methods to migrate an embedded data store to an external store before attempting to upgrade to AM 7:

- Migrate data to an external instance of DS by using LDIF files.

Follow the steps in [How do I migrate from an embedded to external DS/OpenDJ in AM/OpenAM \(All versions\)?](#) in the *ForgeRock Knowledge Base*.

- Add a new external DS instance and replicate data from the embedded instance.

Follow the steps in [Add a New Replica Before Upgrade](#) in the *DS Upgrade Guide*.

1. Ensure you have read "*Planning the Upgrade*" and planned your upgrade accordingly.
2. Prepare your customized AM server `.war` file. For more information, see "*Customizing Before Upgrading*".
3. **Back up your deployment.** For more information, see "*Backing Up the Deployment*".
4. Route client application traffic to another site during the upgrade. For more information, see "*Routing Around Servers During Downtime*".
5. Ensure that an AES 256-bit key called `directenctest` is available to all the instances in the site. It does not need to be the same key that AM provides on the default keystore.

Failure to provide this key will prevent AM from starting up after upgrade.

+ *What Is this Key for?*

AM uses this key to encrypt information stored in the `secure state` in the *Authentication Node Development Guide* of the authentication trees, which is a new and *mandatory* feature of the trees in AM 7 and later.

The upgrade process will map this key to the `am.authn.trees.transientstate.encryption` secret ID.

The way you make the alias available depends on the version of AM you are upgrading from:

- Versions of AM Earlier than 6.5

The key alias must exist in the keystore configured as the AM keystore. Check the path where the keystore and its files are configured by going to Configure > Server Defaults > Security > Key Store.

- AM 6.5

The alias must exist in a secret store configured globally, so that all realms can access it. You can configure additional secrets by realm if required after the upgrade.

You can create a new secret store to provide this secret, or you can add it to one of your existing stores.

The following is an example of how to create the key alias in the AM keystore, or in a keystore configured in a secret store:

```
$ keytool \  
-genseckey \  
-alias directncstest \  
-keyalg AES \  
-keysize 256 \  
-storetype JCEKS \  
-keystore /path/to/keystore.jceks
```

+ *Where Do I Find the Keystore Passwords?*

- (AM versions earlier than 6.5) The passwords are stored in the files configured in Configure > Server Defaults > Security > Key Store.
- (AM 6.5) The passwords are secrets provided by a different secret store. For example, a file system volume secret store.

Make sure that the alias is available to all instances of the site. This may mean, for example, copying the keystore across the site.

After the upgrade, you can rename the key alias or configure a different key in the secret ID mapping, but ensure that this secret ID is always mapped to an existing, resolvable secret or key alias.

6. Stop AM or the container where it runs.
7. Deploy your customized server `.war` file.

When you deploy the new `.war` file, you might have to delete working files left by the old installation. For example, if you deploy on Apache Tomcat, replacing `/path/to/tomcat/webapps/openam.war`, then also recursively delete the `/path/to/tomcat/webapps/openam/` and `/path/to/tomcat/work/Catalina/localhost/openam/` directories before restarting the server.

8. Restart AM or the container where it runs.
9. (Optional) You must update the user store XML schema if you are upgrading from a version of AM earlier than 6 and any of the following statements are true:
 - You have configured knowledge-based (KBA) user self-service questions.
 - You have not configured User Self-Service yet, but you added the `user_store_type_kba.ldif` schema to your external user data store when you configured it.

For more information about LDIFs, see "Supported LDIF Files" in the *Installation Guide*.

To update the user store schema, perform the following steps:

- a. Change directories to the path where you have deployed the `openam.war` file. For example, `/path/to/tomcat/webapps/openam`.
- b. Locate the following files in the `WEB-INF/template/ldif/opensj` path:
 - `opensj_add_kba_attempts.ldif`
 - `opensj_update_aci_kba_attempts.ldif`

Note

If your user data store is not DS, use these files as examples to create files suitable for your environment.

- c. Open the `opensj_update_aci_kba_attempts.ldif` file and replace `@SM_CONFIG_ROOT_SUFFIX` with the base DN defined during the DS installation procedure (for example, `dc=userstore,dc=example,dc=com`).
- d. Update the user data store schema with the two files. For example, to update a DS instance, run the following command:

```
$ /path/to/opendj/bin/ldapmodify \
--hostname 'id.example.com' \
--port 1636 \
--useSsl \
--usePkcs12TrustStore /path/to/opendj/config/keystore \
--trustStorePasswordFile /path/to/opendj/config/keystore.pin \
--bindDN uid=admin \
--bindPassword str0ngAdm1nPa55word \
/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opendj/opendj_add_kba_attempts.ldif \
/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opendj/opendj_update_aci_kba_attempts.ldif
# Processing MODIFY request for cn=schema
# MODIFY operation successful for DN cn=schema
# Processing MODIFY request for dc=userstore,dc=example,dc=com
# MODIFY operation successful for DN dc=userstore,dc=example,dc=com
```

Note that you will need to update the user store schema again in a later step whether you performed this step or not.

10. To upgrade the data in the configuration store, perform one of the following actions in one of the servers in the site:

- Navigate to the AM URL, for example <https://openam.example.com:8443/openam>, and follow the instructions in the Upgrade Wizard for an interactive upgrade.

Important

After deploying AM, but before upgrade, your application container serves AM's upgrader user interface.

We recommend that any external network access to the application container is suspended until the upgrade is complete. When complete, AM prevents access to the upgrader UI itself.

- Use the `openam-upgrade-tool-14.1.3.11.jar` tool for an unattended upgrade:
 1. Install the `openam-upgrade-tool-14.1.3.11.jar` tool as described in "To Set Up the Configuration Tools" in the *Installation Guide*. A `sampleupgrade` file will be expanded in the directory where you install the tool.
 2. Create a configuration file for the `openam-upgrade-tool-14.1.3.11.jar`. You can use the `sampleupgrade` file as a template to create a configuration file, for example `upgrade.properties`.

An upgrade configuration file may resemble the following:

```
$ grep -v "^#" upgrade.properties
SERVER_URL=http://openam.example.com:8080
DEPLOYMENT_URI=/openam
ACCEPT_LICENSES=true
```

3. Upgrade AM by using the tool with the properties file following this example:

```
$ java -jar openam-upgrade-tool-14.1.3.11.jar --file upgrade.properties
Writing Backup; Done.
Upgrading Services
```



```
New service iPlanetAMAuthPersistentCookieService; Done.
New service iPlanetAMAuthOpenIdConnectService; Done.
New service OAuth2Provider; Done.
New service iPlanetAMAuthDevicePrintModuleService; Done.
New service crestPolicyService; Done.
New service RestSecurity; Done.
New service MailServer; Done.
New service dashboardService; Done.
New service iPlanetAMAuthOATHService; Done.
Add Organization schema to sunFAMSAML2Configuration; Done.
Upgrade sunAMAuthHOTPSERVICE; Done.
Upgrade sunAMAuthADService; Done.
Upgrade sunAMAuthOAuthService; Done.
Upgrade iPlanetAMAuthCertService; Done.
Upgrade sunIdentityRepositoryService; Done.
Upgrade iPlanetAMPASSWORDResetService; Done.
Upgrade iPlanetAMSessionService; Done.
Upgrade iPlanetAMAuthService; Done.
Upgrade iPlanetAMAuthLDAPService; Done.
Upgrade sunAMAuthDataStoreService; Done.
Upgrade AgentService; Done.
New sub schema sunIdentityRepositoryService; Done.
New sub schema AgentService; Done.
Delete service sunFAMLibertyInteractionService; Done.
Delete service sunFAMLibertySecurityService; Done.
Creating entitlement application type crestPolicyService; Done.
Creating entitlement application crestPolicyService; Done.
Re-enabling Generic LDAPv3 Data Store; Done.
Upgrading data store embedded; Done.
Updating Platform Properties; Done.
Writing Upgrade Log; Done.

Upgrade Complete.
```

For additional information about the command-line tool, see the reference documentation for `upgrade.jar(1)` in the *Reference*.

4. Restart AM or the container where it runs.
11. Add an access control instruction (ACI) to the configuration store directory to give the AM administrative user server-side sorting privileges.

The ACI should be similar to the following:

```
aci: (targetcontrol="1.2.840.113556.1.4.473")(version 3.0;
acl "Allow server-side sorting"; allow (read)
(userdn = "ldap:///uid=openam,ou=admins,dc=example,dc=com");)
```

See "Preparing Configuration Stores" in the *Installation Guide* for more information about configuring AM configuration stores.

12. Update the identity store schema as follows:
 - a. Log into AM.

- b. Navigate to *Realm Name* > Datastores > *External User Store*.
- c. Click Load Schema before saving, and then click Save to apply your changes.
- d. If you have additional user stores, repeat the previous steps for each user store.

Tip

If you need to manually update your identity store user schema, see "Updating the Schema in an External Identity Repository" in the *Installation Guide*.

13. Depending on the version from which you are upgrading, you might need to update the schema in the Core Token Service (CTS).

If you are updating from a version prior to AM 6.5, apply the following LDIF file:

- `cts-add-ttlextire.ldif`

If you are updating from a version prior to AM 6, also apply the following LDIF files:

- `cts-add-multivalue.ldif`
- `cts-add-multivalue-indices.ldif`

Note

Replace the `@DB_NAME@` variable inside the `cts-add-multivalue-indices.ldif` file with the CTS backend name. For example, replace occurrences of `@DB_NAME@` with `ctsStore`.

•

For example:

```
$ /path/to/openssl/bin/openssl \
--hostname 'cts.example.com' \
--port 1636 \
--useSsl \
--usePkcs12TrustStore /path/to/openssl/config/keystore \
--trustStorePasswordFile /path/to/openssl/config/keystore.pin \
--continueOnError \
--bindDN uid=admin \
--bindPassword str0ngAdm1nPa55word \
/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/sfha/cts-add-multivalue.ldif \
/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/sfha/cts-add-multivalue-indices.ldif \
/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/sfha/cts-add-ttlextire.ldif
```

14. (Optional) If you intend to use push or web authentication, or to use the ForgeRock SDK for device profiling, then you must update your identity store user schema. You can ensure the correct parameters are available by applying the following LDIF files:

- `/path/to/openam/WEB-INF/template/ldif/opensj/push_2fa.ldif`
- `/path/to/openam/WEB-INF/template/ldif/opensj/opensj_webauthndevices.ldif`
- `/path/to/openam/WEB-INF/template/ldif/opensj/opensj_deviceprofiles.ldif`

For example:

```
$ /path/to/opensj/bin/ldapmodify \  
--hostname 'id.example.com' \  
--port 1636 \  
--useSsl \  
--usePkcs12TrustStore /path/to/opensj/config/keystore \  
--trustStorePasswordFile /path/to/opensj/config/keystore.pin \  
--continueOnError \  
--bindDN uid=admin \  
--bindPassword str0ngAdm1nPa55word \  
/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opensj/opensj_webauthndevices.ldif\  
/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opensj/opensj_deviceprofiles.ldif
```

Important

If you do not apply these schema changes, after upgrading AM you should remove the `webauthnDeviceProfilesContainer` object class from the user configuration.

In the AM console, navigate to Realms > *Realm Name* > Identity Stores > *Identity Store Name*. On the User Configuration tab, remove `webauthnDeviceProfilesContainer` from the LDAP User Object Class property, and then save your changes.

Ensure you make the same change for each identity store that does not have the schema change, and in each realm that uses the identity store.

For more information on these LDIF files, and the equivalent files for supported directory servers, see "Supported LDIF Files" in the *Installation Guide*.

15. (Optional) Install a new version of the AM tools as described in "Setting Up Administration Tools" in the *Installation Guide* and in the *ForgeRock Identity Platform Amster User Guide*.

Once the new tools are working, delete the old tools.

16. Review the information in "Upgrading Components and Services" and decide if you need to reconfigure any of AM's services or features.

Note

Reconfigure any custom advanced properties if necessary. These properties are lost during the upgrade, and you will need to add them again in the AM Admin UI.

+ *How Do I Configure Advanced Server Properties?*

- To configure advanced server properties in the AM Admin UI for all AM instances, go to Configure > Server Defaults > Advanced.
- To configure advanced server properties for a particular instance, go to Deployment > Servers > *Server Name* > Advanced.

If the property you want to add or edit is not already configured, add it with its value, then click on the plus (+) button.

If the property you want to add or edit is already configured, click on the pencil (✎) button to edit it. When you are finished, click on the tick (✓) button.

Save your changes.

17. Ensure that the AM scripts are current, and that they contain the modifications that your environment requires.

To avoid overwriting changes done to the original files, the upgrade process *does not* update scripts from earlier versions of AM. Ensure that the scripts in your environment are compatible with the version of AM you upgraded to by performing the following steps:

1. Read the release notes for information about possible changes.
2. Install an AM 7.0.2 test environment, and compare the scripts. New installations always have the current scripts.

18. Validate that the service is performing as expected.
19. Allow client application traffic to flow to the upgraded site.

Chapter 4

Upgrading Components and Services

As part of planning your upgrade, you need to consider that certain changes in later AM versions may have an impact on your environment. Usually, these changes are driven by changes in specification, security policies, or performance.

When possible, the upgrade process makes the appropriate changes on AM configuration. However, sometimes you will need to perform additional configuration based on your environment needs.

In addition to mandatory upgrade steps outlined in "*Upgrading AM Instances*", if you are using features described in the following table you will need to perform additional upgrade tasks:

Critical Changes to Existing Functionality

AM Version	Component or Feature	Change
7.0.2	Decompressed JWTs	By default, AM rejects any JWT that expands to more than 32 KiB (32768 bytes) when decompressed. For more information about changing this default value, see "Controlling the Maximum Size of Compressed JWTs" in the <i>Security Guide</i> .
	Request Body Size	By default, AM rejects incoming requests with a body larger than 1 MB (1048576 bytes) in size. For more information about changing this default value, see "Limiting the Size of the Request Body" in the <i>Security Guide</i> .
	One-Time Passwords in Authentication Nodes	One-time passwords created by the "HOTP Generator Node" in the <i>Authentication and Single Sign-On Guide</i> are now stored in the authentication tree's transient state in the <i>Authentication Node Development Guide</i> . Modify any custom authentication nodes or scripts used by the "Scripted Decision Node" in the <i>Authentication and Single Sign-On Guide</i> to retrieve the one-time passwords from the transient state after upgrading.
7	User Profile Whitelist	The profile attribute whitelist controls the information returned to non-administrative users when accessing <code>json/user</code> endpoints. Common profile attributes are whitelisted by default, but you need to add any custom attribute you want your non-administrative users to see. For more information, see "Configuring the User Profile Whitelist".

AM Version	Component or Feature	Change
	<code>/json/authenticate</code> Endpoint	<p>When a client makes a call to the <code>/json/authenticate</code> endpoint appending a valid SSO token, AM returns the <code>tokenId</code> field empty if <code>HttpOnly</code> cookies are enabled. For example:</p> <pre>{ "tokenId": "", "successUrl": "/openam/console", "realm": "/" }</pre>
	Secure Authentication Tree State Secret ID	<p>An AES 256-bit key called <code>directenctest</code> must be available in the environment during upgrade, but it does not need to be the same key that AM provides on the default keystore.</p> <p>After upgrade, ensure that the <code>am.authn.trees.transientstate.encryption</code> secret ID is always mapped to an existing, resolvable secret or key alias. Failure to do so may result in trees not working as expected.</p>
	The Embedded DS	<p>The embedded DS can only be used for single AM instances, for test and demo purposes. Sites are not supported.</p> <p>Sites using embedded DS servers must be migrated to external DS servers before upgrading.</p>
	SAML v2.0 Secrets	<p>AM 7 migrated SAML v2.0 to use secret stores. The upgrade process only creates the secret store files on the AM instance where you ran the upgrade process. For more information, see "Configuring Secret Stores After Upgrade".</p>
	<code>goto</code> and <code>gotoOnFail</code> Query Parameter Redirection	<p>Redirection URLs for authentication services, agents, and SAML v.2.0 must be configured in the <i>Validation Service</i> in the <i>Authentication and Single Sign-On Guide</i> if they are not in the same scheme, FQDN, and port as AM, or are not relative to AM's URL.</p>
	Web Agents of a Version Earlier than 5.6.3	<p>Several properties that used to be configured as custom properties (<code>com.sun.identity.agents.config.freeformproperties</code>) have been added as regular properties. Due to this change, upgrading to AM 7 will overwrite the value of the original custom properties with the default value of the new UI properties.</p> <p>To work around this issue, perform one of the following actions:</p> <ul style="list-style-type: none"> • Upgrade to Web Agents 5.6.3 or later before upgrading to AM 7. • After upgrading to AM 7, reconfigure the properties that you configured as custom properties in their new UI counterparts.
	Changes on the CTS Reaper Tuning Properties	<p>AM 7 changes the way the CTS reaper searches for expired tokens.</p> <p>After upgrading, retune the CTS Reaper using the information in "Reaper Search Size" in the <i>Core Token Service Guide (CTS)</i>.</p>

AM Version	Component or Feature	Change
	OpenID Connect Clients Authenticating with JWTs	OpenID Connect clients authenticating with JWTs must include in the JWT a jti claim containing a unique identifier, in line with OpenID Connect Core 1.0 incorporating errata set 1.
	Cookie Filter	AM flags cookies as secure if they come through a connection marked as secure, or if they come through HTTPS. See "Managing the Secure Cookie Filter" in the <i>Security Guide</i> .
6.5.0.2 // 6.5.1	OAuth 2.0 Refresh Tokens	<p>AM only issues refresh tokens to clients that have the refresh token grant type configured in their client profile.</p> <p>After an upgrade to 6.5 or later using the UI or the openam-upgrade-tool .jar file, existing OAuth 2.0 clients are configured to use all grant flows, including the Refresh Token Grant flow.</p> <p>To configure the refresh token grant type manually, see "To Configure AM to Issue Refresh Tokens" in the <i>OAuth 2.0 Guide</i>.</p>
6.5	Recovery Codes	Recovery Codes are encrypted, and existing codes are no longer displayed to the user. For more information, see "Upgrading Device Recovery Codes".
	Secret Stores	AM 6.5 introduced secret stores for OAuth 2.0 and the persistent cookie module. The upgrade process only creates the secret store files on the AM instance where you ran the upgrade process. For more information, see "Configuring Secret Stores After Upgrade".
	External Configuration Store	<p>DS 6.5 introduced setup profiles, which pre-configure instances for different usages, such as CTS or configuration data. The default base DN for a DS configuration store instance (ou=am-config) is different than the default used by previous versions of AM (dc=openam,dc=forgerock,dc=org).</p> <p>You should not attempt to run multiple instances of AM where the configuration store base DNs do not match. Use the same configuration store base DNs when configuring external DS 6.5+ instances that will be used simultaneously alongside existing DS 6 or earlier configuration store instances.</p> <p>For more information, see "Preparing Configuration Stores" in the <i>Installation Guide</i>.</p>
6	JSON Endpoints	AM's CSRF protection filter requires that either the X-Requested-With or the Accept-API-Version headers are included on requests to endpoints under the json root. For more information, see "Reviewing REST API Versions Before Upgrading".

Configuring the User Profile Whitelist

AM 7 introduced a 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.

Note that the `kbainfo` attribute is required to be whitelisted for users to manage their KB questions and answers on user self-service flows.

Configuring Secret Stores After Upgrade

AM 6.5 introduced secret stores, which are repositories of cryptographic keys, key pairs, and credentials.

While the configuration for secret stores is available to any upgraded server in the site, **the upgrade process only creates the relevant secret store files on the AM instance where you ran the upgrade process.**

Perform the steps in the following procedure to make the secret stores infrastructure available to the other servers in the site:

To Redeploy Secret Stores to a Site After Upgrade

You can reconfigure the secret stores and their mappings after upgrade. However, we recommend that you follow the steps in this procedure to ensure all secrets are available to all the instances in the site, and later on, you make additional changes to your environment.

The upgrade process creates several secret stores, globally and by realm, depending on the features configured in AM, and depending on the version you are upgrading from:

1. Navigate to Configuration > Secret Stores, and review the global secret stores created for your environment.

+ *Reference: Global Secret Stores Created After Upgrade*

Upgrade from AM 6 or earlier

- **default-keystore**: a keystore-type secret store configured to the path of the AM keystore, as configured on the server where you ran the upgrade process.
- **default-passwords-store**: A filesystem secret store configured as the `/path/to/openam/security/secrets/encrypted` directory.

This directory contains the secrets to open the keystore configured in the **default-keystore**, and its keys.

- **UpgradeGlobalSecrets**: A filesystem secret store configured as the `/path/to/openam/security/secrets/encrypted/encrypted_hmac_key` directory.

This directory contains the secrets used for the OAuth 2.0 server and the Persistent Cookie module.

Upgrade from AM 6.5 or earlier

- **default-keystore**: a keystore-type secret store configured to the path of the AM keystore, as configured on the server where you ran the upgrade process.
- **saml2-metadata-signing-keystore**: A keystore secret store configured to the path of the AM keystore, as configured on the server where you ran the upgrade process.
- **default-encrypted_base64-store**: A file system volume secret store that contains SAML v2.0 secrets that are base64-encrypted.
- **default-encrypted_plain-store**: A file system volume secret store configured as the directory containing secrets related to the **saml2-metadata-signing-keystore** store.

The directory contains the password that was configured in the Metadata signing key password field of the global SAML v2.0 Service Configurations service.

It also contains the password files related to the **default-keystore** store if the **default-passwords-store** store does not exist.

- Ensure that the keystores configured exist on the other servers within the site. You may need to copy the keystores across or make them available in some other way.
 - Ensure that directories configured in file system secret stores and their content exist on the other servers within the site. You may need to copy the directories across or make them available in some other way.
2. Navigate to Realms > *Realm Name* > Secret Stores, and perform the same actions you took for the global secret stores.

+ *Reference: Realm-Based Secret Stores Created After Upgrade*

Realm-based secret stores are created for those features that supported different keystore configurations by realm; for example, SAML v2.0, or the persistent cookie module.

To find the realm-based secret stores, go to Realms > *Realm Name* > Secret Stores. The secrets themselves are stored in the `/path/to/openam/security/secrets/realms/root/realm_name/secret_store_name` directory.

If you have upgraded from a version which had SAML v2.0 Service Configurations at realm level, the upgrade process creates realm-level secret stores for the Metadata signing key alias field and its key, if they were configured.

Repeat this step for each of the realms you have configured.

3. Deploy the new AM `.war` file on the rest of the AM servers.
4. Once the site is up, and before opening the service to the end users, review the secret ID mappings of the new secret stores and make changes to them as you see fit.

For example, the upgrade process may have created stores for features you are not using. Those may have mappings to secrets that do not exist, and you may want to remove them in production environments.

For more information about the available secret IDs, see "Secret ID Default Mappings" in the *Security Guide*.

+ *Reference: SAML v2.0 Mappings After Upgrade*

AM has always been very flexible regarding the configuration of secrets for SAML v2.0; this is why migrating the different combinations may create a high number of secret IDs in your environment.

As a rule of thumb, AM configures providers that were using the same key aliases *in the same order*, to use the same secret IDs. If this rule cannot be satisfied, the upgrade process creates new secretIDs for the provider by assigning it a secret ID identifier.

If you have upgraded from a version which had SAML v2.0 Service Configurations, AM maps the `am.services.saml2.metadata.signing.RSA` secret ID to the alias taken from the relevant

Metadata signing key alias property of the service, either from the global service, or the realm-level services.

Upgrading Device Recovery Codes

This section explains how to upgrade to AM 6.5 and later if you are providing the ability for ForgeRock Authenticator users to access and view device recovery codes.

AM versions earlier than 6.5 do not encrypt the recovery codes stored alongside registered push and OATH devices. This allows the codes to be viewed by users at any time in their dashboard page. However storing credentials in plain text is considered a potential security risk, and from AM 6.5 onwards the recovery codes are displayed once, and then stored in a one-way encryption format, meaning they can never be viewed after their initial display.

After upgrading to AM 6.5 or later, when a user accesses their dashboard page, the stored recovery codes for each registered device will be one-way encrypted, meaning existing codes can no longer be displayed to the user.

This DOES NOT affect the ability to use the existing recovery codes, only the ability to display them in plain text to the user.

If you do not want to encrypt the recovery codes, and therefore retain the ability to show the codes to the user when requested, you can start AM with a Java property, as follows:

To Prevent AM Encrypting Device Recovery Codes

Perform these steps to prevent AM 6.5 and later from encrypting device recovery codes.

Important

It is **STRONGLY** recommended that recovery codes are encrypted.

1. Locate or create the environment settings script for the container in which AM will run.

For example, the environment settings script for Apache Tomcat is located in `/path/to/tomcat/bin/`, and should be named `setenv.bat` (Windows) or `setenv.sh` (Unix).

2. In the relevant environment settings script, add the `org.forgerock.openam.devices.recovery.use_insecure_storage=true` property to the `CATALINA_OPTS` variable. For example:

```
export CATALINA_OPTS="$CATALINA_OPTS -
Dorg.forgerock.openam.devices.recovery.use_insecure_storage=true"
```

For containers other than Apache Tomcat, perform an analogous step to add the Java option to the scripts used to startup the AM instance.

3. Start the container in the usual manner. For example, `./startup.sh`.

AM will not encrypt device recovery codes when created, or when first accessed. By preventing AM from encrypting the stored recovery codes, you should be aware of the following points:

- Users will only see registered devices on their dashboard that are of the same type that they have used to authenticate.

For example, if they authenticated using a registered OATH device, they will not see any registered push or WebAuthn devices on their dashboard. This is to prevent users being able to see recovery codes for devices that they did not authenticate with.

- The option to view the recovery codes for a device has been removed from the user interface.

However, the recovery codes are returned in the JSON response when querying the `/devices/2fa/` endpoint. You will need to provide a customized user interface to display these codes.

- If the container in which AM is running is ever started without the `org.forgerock.openam.devices.recovery.use_insecure_storage=true` property, a query to any of the `/devices/2fa/` endpoints will cause AM to one-way encrypt the recovery codes.

Upgrading JDBC Audit Event Handlers

If you had configured one or more JDBC audit event handlers, make the following changes to the audit tables' schema:

To Upgrade JDBC Audit Event handlers

1. Run the following command on Oracle databases that support AM audit event handlers:

```
ALTER TABLE am_auditaccess ADD (response_detail CLOB NULL);
```

This command adds the `response_detail` column to the `am_auditaccess` table.

2. Run the following commands on MySQL databases that support AM audit event handlers:

```
ALTER TABLE audit.am_auditconfig CHANGE COLUMN configobjectid objectid VARCHAR(255);  
ALTER TABLE audit.am_auditaccess ADD COLUMN response_detail TEXT NULL;
```

The commands change the name of the `configobjectid` column in the `am_auditconfig` table to `objectid` and add the `response_detail` column to the `am_auditaccess` table.

3. If you use databases other than Oracle or MySQL to support AM audit event handlers, review their schema.

If the `am_auditconfig` table has a column named `configobjectid`, change that column's name to `objectid`.

If the `am_auditaccess` table does not have a column named `response_detail`, add that column to the table's schema.

Chapter 5

Migrating Legacy Instances

Rather than upgrade legacy instances (running OpenSSO or Sun Access Manager, or an OpenAM or AM version that is no longer supported), you instead manually migrate from your existing deployment to a new deployment.

For complex legacy deployments, ForgeRock can assist you in the migration process. Send mail to info@forgerock.com for more information.

To Upgrade A Legacy Deployment

1. Prepare your customized AM WAR file.
2. Prepare a new deployment, installing servers from the new, customized WAR file, starting with the instructions in "*Installing Instances*" in the *Installation Guide*.
3. After installation, configure the new servers in the same way as the old servers, adapting as necessary.

You can use the **ssoadm do-batch** command to apply multiple changes with one command.

4. Validate that the new service is performing as expected.
5. Redirect client application traffic from the old deployment to the new deployment.

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

	<p>request. For browser-based clients, AM sets a cookie in the browser that contains the session information.</p> <p>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.</p>
Conditions	<p>Defined as part of policies, these determine the circumstances under which which a policy applies.</p> <p>Environmental conditions reflect circumstances like the client IP address, time of day, how the subject authenticated, or the authentication level achieved.</p> <p>Subject conditions reflect characteristics of the subject like whether the subject authenticated, the identity of the subject, or claims in the subject's JWT.</p>
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 <i>reference</i> to the token to the client, rather than the token itself. This differs from <i>client-based OAuth 2.0 tokens</i> , 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	AM unit for organizing configuration and identity information. 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. Administrators can delegate realm administration. The administrator assigns administrative privileges to users, allowing them to perform administrative tasks within the realm.
Resource	Something a user can access over the network such as a web page. Defined as part of policies, these can include wildcards in order to match multiple actual resources.
Resource owner	In OAuth 2.0, entity who can authorize access to protected web resources, such as an end user.
Resource server	In OAuth 2.0, server hosting protected web resources, capable of handling access tokens to respond to requests for such resources.
Response attributes	Defined as part of policies, these allow 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.