Upgrade Guide

/ ForgeRock Access Management 6.5

Latest update: 6.5.3

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

Abstract

This guide shows you how to upgrade ForgeRock® Access Management. ForgeRock Access Management provides authentication, authorization, entitlement, and federation software.

Table of Contents

Preface

The Upgrade Guide describes how to upgrade ForgeRock Access Management servers, web and Java agents, and tools.

This guide is for anyone who needs to upgrade an AM deployment. This guide assumes you are familiar with installation and configuration, and that you are familiar with the current deployment that you plan to upgrade.

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

About Upgrading

This chapter 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 "ForgeRock Product Release Levels" in the *Release Notes*. Release levels are identified by version number.

1.1. Supported Upgrade Paths

The following table contains information about the supported upgrade paths to AM 6.5.3:

Upgrade Paths

Version	Upgrade Supported?
AM 6.5.x	✔ [a]
AM 6.0.x	✔ [a]
AM 5.5	✔
AM 5.0 (14.0)	✔
OpenAM 13.5.x	✔
OpenAM 13.x	✔

[a]

[a] **Important**
The **Amster-config-upgrader** tool was removed from the AM 6.5.0 and later releases. As a result, after you upgrade your AM servers, you must manually export any Amster configuration files for them to be valid on the upgraded server. The Amster export applies to upgrades *to* AM 6.5.0, or from AM 6.5.0/6.5.0.x to AM 6.5.x (for example, AM 6.5.1 or 6.5.2). For more information, see the BackStage 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.

Upgrading between Enterprise and OEM versions is not supported.

For more information, see Checking your product versions are supported in the *ForgeRock Knowledge Base*.

1.2. 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 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"
- "Testing the Upgrade"

1.2.1. 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 can therefore initiate the upgrade process.

1.2.2. 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 and Restoring Configurations" in the *Setup and 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.

1.2.3. 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 an Explicit REST API Version" in the *Development Guide*.

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 *Setup and Maintenance Guide*.

You can configure AM's default REST API behavior. For more information, see "Configuring the Default REST API Version for a Deployment" in the *Development Guide*.

1.2.4. Reviewing Directory Services Certificates Before Upgrading

Before upgrading, review that 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 Administration Guide.

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

1.2.6. 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 and Restoring Configurations" in the *Setup and Maintenance Guide*.

1.2.7. Testing the Upgrade

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 2
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-14.1.2.2.jar**, which you install as described in "Setting up 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" in the *Release Notes*.

Important

Do *not* perform an upgrade by deploying the new version and then importing an existing configuration by running the **ssoadm import-svc-config** command. Importing an outdated configuration can result in a corrupted installation.

1. Ensure you have read Chapter 1 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. Stop AM or the container where it runs.

6. 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.

7. Restart AM or the container where it runs.

8. (Optional) You must update the user store XML schema if you are upgrading from OpenAM or 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*.
   - You are using the embedded DS instance as a user store.

   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/opendj` path:

      - `opendj_add_kba_attempts.ldif`
      - `opendj_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 `opendj_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:

```
$ ldapmodify \
 --hostname opendj.example.com \
 --port 1389 \
 --bindDN 'cn=Directory Manager' \
 opendj_add_kba_attempts.ldif opendj_update_aci_kba_attempts.ldif
Password for user 'cn=Directory Manager':

 # 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.

9. 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:443/openam`, and follow the instructions in the Upgrade Wizard for an interactive upgrade.
   - Use the `openam-upgrade-tool-14.1.2.2.jar` tool for an unattended upgrade:

     1. Install the `openam-upgrade-tool-14.1.2.2.jar` tool as described in "Setting up 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-14.1.2.2.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-14.1.2.2.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.

10. (Optional) If you installed AM using an external directory server as the configuration store, add an access control instruction (ACI) to the external 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 using an external directory server as the AM configuration store.

11. (Optional) If you installed AM using an external directory server as the user store, update the user 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 external user stores, repeat the previous steps for each user store.

12. (Optional) If you are using an external Core Token Service (CTS) token store, update the schema by applying the following LDIF files to it:

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

  Note

  Ensure to 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`.

- `cts-add-ttlexpire.ldif`

For example:

```
$ ./ldapmodify
 \
--hostname 'config.example.com'
 \
--port 1389
 \
--useStartTLS
 \
--trustAll
 \
--continueOnError
 \
--bindDN 'cn=Directory Manager'
 \
--bindPassword 'str0ngEx4mplePa55word'
 \
/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-ttlexpire.ldif
```

13. If you are using an external identity store, and intend to use push or web authentication, update the schema by applying the following LDIF files to it:

    - `push_2fa.ldif`
    - `opendj_webauthndevices.ldif`

    For example:

```
$ ./ldapmodify
 \
--hostname 'id.example.com'
 \
--port 1389
 \
--useStartTLS
 \
--trustAll
 \
--continueOnError
 \
--bindDN 'cn=Directory Manager'
 \
--bindPassword 'str0ngEx4mplePa55word'
 \
/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opendj/push_2fa.ldif
 \
/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opendj/opendj_webauthndevices.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 external identity store that does not have the schema change, and in each realm that uses the external 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*.

14. (Optional) Install a new version of the AM tools as described in "Installing and Using the Tools" in the *Installation Guide* and in the *ForgeRock Identity Platform Amster User Guide*.

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

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

16. Validate that the service is performing as expected.

17. Allow client application traffic to flow to the upgraded site.

Chapter 3
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
6.5.3	`goto` and `gotoOnFail` Query Parameter Redirection	Redirection URLs for authentication services, agents, and SAML v.2.0 must be configured in the Validation Service if they are not in the same scheme, FQDN, and port as AM, or are not relative to AM's URL.
	`/json/authenticate` Endpoint	When a client makes a call to the `/json/authenticate` endpoint appending a valid SSO token, AM now returns the `tokenId` field **empty** when `HttpOnly` cookies are enabled. For example: `{ "tokenId":"", "successUrl":"/openam/console", "realm":"/" }`
	SAML v2.0 Assertion Consumer Service	SAML v2.0 assertion consumer service URLs must exactly match the the SP's scheme, FQDN, and port.
	SAML v2.0 RelayState Redirection	To redirect to a domain outside of AM's deployment domain, you must add it to the Relay State URL List whitelisting property of the SP or IDP.
6.5.0.2 // 6.5.1	OAuth 2.0 Refresh Tokens	AM only issues refresh tokens to clients that have the `refresh token` grant type configured in their client profile. 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.

AM Version	Component or Feature	Change
		To configure the `refresh token` grant type manually, see "To Configure AM to Issue Refresh Tokens" in the *OAuth 2.0 Guide*.
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	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`). 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. For more information, see "Preparing Configuration Stores" in the *Installation Guide*.
	Amster	The **Amster-config-upgrader** tool was removed. As a result, you need to upgrade AM following the procedures in the *Upgrade Guide* and then, export the configuration from the upgraded instance or site using Amster. For more information, see the following Knowledge Base article.
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".
5	SSO Tokens	AM SSO session tokens are incompatible with SSO tokens from OpenAM. CTS-based (stateful) and client-based (stateless) sessions created by earlier versions of OpenAM are not supported. After upgrading from an earlier version, any existing SSO tokens created by that version will become invalid, and users will need to reauthenticate. In mixed version deployments, earlier versions of OpenAM will not be able to read or process SSO session tokens created by AM 5 or later. This incompatibility only affects SSO session tokens. OAuth 2.0 and OpenID Connect 1.0 tokens are interoperable between versions.

AM Version	Component or Feature	Change
	Realms	Realm paths now must be absolute and include the top-level realm, and DNS aliases and realms specified in the query string are no longer concatenated if used together – the query string overrides the DNS alias. For examples, see "Specifying the Realm in the Login URL" and "Specifying Realms in REST API Calls" in the *Authentication and Single Sign-On Guide*. This change also impacts the user self-service feature when deployed in subrealms. For more information, see "Upgrading User Self-Service in Subrealms".
	Post-Authentication Plugins	AM does not maintain state in post-authentication plugins between login and logout anymore. For more information, see "Upgrading Post-Authentication Plugins".
13.5	User Self-Service	The user self-service feature requires two keys in a JCEKS keystore. For more information, see "Upgrading the Keystore for User Self-Service".

3.1. Configuring Secret Stores After Upgrade

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

The upgrade process configures the following global secret stores in the environment:

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

  This directory contains the secrets to open the keystore configured in the `default-keystore`, and its keys.
- `UpgradeGlobalSecrets`: A filesystem-type secret store configured as the `/path/to/openam/secrets/encrypted_hmac_key` directory.

  This directory contains the secrets used by the AM features that use secret stores, such as the OAuth 2.0 server and the Persistent Cookie module.

  Note that this secret store is always created during the upgrade, even when the features that depend on it are not configured. This is so you can configure any feature after upgrade and have the required secret infrastructure already in place.

The upgrade process will also create realm-based secret stores as required by your environment. To find them, navigate to Realms > *Realm Name* > Secret Stores.

For example, a filesystem-type secret store configured for the realm `mytestrealm` would be configured as the `/path/to/openam/secrets/realms/root/mytestrealm/encrypted_hmac_key` directory.

Tip

You can reconfigure the secret stores after upgrade. The procedure below assumes you want to upgrade your environment using the configuration created by the upgrade process. For more information about secret stores, see "*Configuring Secrets, Certificates, and Keys*" in the *Setup and Maintenance Guide*.

While the configuration for secret stores is available to any upgraded server in the site, **the upgrade process only creates the 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

1. Navigate to Configuration > Secret Stores.
2. Review the `default-keystore` secret store, configured by default to use AM's default keystore.

   The keystore configured in the secret store should already exist on the other servers in the site. If it does not, or if it is in a different directory, copy the keystore file to the correct location.
3. Review the `default-passwords-store` secret store and copy the contents of the `/path/to/openam/secrets/encrypted` directory to the same directory on the rest of the AM servers.
4. Review the `UpgradeGlobalSecrets` secret store and copy the contents of the `/path/to/openam/secrets/encrypted_hmac_key` directory to the same directory on the rest of the AM servers.
5. Navigate to Realms > *Realm Name* > Secret Stores, and review the secret stores:
   - The keystore should already exist in the rest of the servers of the site. If it does not, copy the keystore file to the correct location.
   - Copy the contents of the filesystem-based store to the same directory on the rest of the AM servers.
6. Repeat the previous step for each of the realms.
7. Deploy the new AM `.war` file on the rest of the AM servers.

3.2. 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 XUI 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.

3.3. Upgrading Post-Authentication Plugins

If you have post-authentication plugins that expect state to be maintained by AM between login and logout, you must rewrite and redeploy them.

In versions prior to AM 5, the Keep Authentication Module Objects for Logout Processing option was available in the Core Authentication module. This option, when enabled, directed AM to maintain state information in server memory throughout a session's duration for post-authentication plugin module instances. When logout was triggered, AM invoked the same post-authentication plugin module instance, with state information intact. Therefore, developers could access module state stored at login when users logged out.

AM 6.5 does not maintain state in post-authentication plugins between login and logout. Post-authentication plugins that rely on module state being maintained in AM's memory between login and logout must be rewritten. You can store any information that you want to save between login and logout in a session property. AM stores session properties in the CTS token store after login, and retrieves them from the token store as part of the logout process.

To set a session property, call the `setProperty` method on an `SSOToken` object as demonstrated by the post-authentication plugin sample code in "Building Your Sample Post-Authentication Plugin" in the *Authentication and Single Sign-On Guide*.

3.4. Upgrading User Self-Services

This section covers upgrading user self-service features.

3.4.1. Upgrading the Keystore for User Self-Service

AM's key management system allows the user self-service feature to successfully operate in a multi-instance server deployment behind a load balancer.

When upgrading from a version previous to OpenAM 13.5, AM deploys a JCEKS keystore that includes demo user self-service keys. This keystore is not configured as the default keystore after

the upgrade because your existing deployment might depend on the JKS keystore. For example, you might have deployed SAML v2.0 using key aliases in the JKS keystore.

To help you decide whether to enable a JCEKS keystore after upgrading, see the following table:

User Self-Service Feature Upgrade

Upgrading from:	Enabling JCEKS required?
Versions prior to OpenAM 13.0	No
OpenAM 13.0 with the REST-based user self-service feature disabled	No
OpenAM 13.0 with the legacy user self-service feature enabled	No
OpenAM 13.0 with the REST-based user self-service feature enabled	*Yes*
OpenAM 13.5 with the REST-based user self-service feature enabled	It is already enabled.

You should not use the demo user self-service keys included in the JCEKS keystore for production purposes. Instead, create new key aliases for user self-service and configure them in AM. When moving your keystore from JKS to JCEKS, you must also review your existing use of keys in AM, and add existing keys available in the JKS keystore to the JCEKS keystore. For example, if you have a SAML v2.0 deployment that uses keys in AM's JKS keystore, you need to add the keys to the JCEKS keystore.

See the following sections for details:

- For more information about keystores in AM, how to configure a JCEKS keystore, and how to create new user self-service keys, see "*Configuring Secrets, Certificates, and Keys*" in the *Setup and Maintenance Guide*.
- For more information about configuring user self-service keys in AM, see "Creating a User Self-Service Service Instance" in the *User Self-Service Guide*.

3.4.2. Upgrading User Self-Service in Subrealms

AM 5.0 altered the method for specifying the realm in URLs. Upgrading from a previous version which has user self-service enabled in a subrealm requires that this new method is applied to the URLs used in confirmation emails, as follows:

To Upgrade User Self-service in a Subrealm

1. Log in to the AM console of the upgraded instance as an administrator, for example `amAdmin`.
2. Navigate to Realms > *Subrealm Name* > Services > User Self-Service, and then click the Advanced Configuration tab.

Note that to view the Advanced Configuration tab you may need to click the small downwards-pointing triangle icon.

3. On the Advanced Configuration tab, alter the following properties to include a `realm` parameter, as in the following examples:

   User Registration Confirmation Email URL

   ```
   https://openam.example.com:8443/openam/XUI/?realm=${realm}#register/
   ```

   Forgotten Password Confirmation Email URL

   ```
   https://openam.example.com:8443/openam/XUI/?realm=${realm}#passwordReset/
   ```

4. Save your changes.

A clean install of AM will include a `realm` parameter in these properties by default.

3.5. 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.

3.6. Upgrading Web Agents and Java Agents

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 Access Management Web Agents User Guide* or the *ForgeRock Access Management Java Agents User Guide*.

Chapter 4

Migrating Legacy Servers

Rather than upgrade legacy servers (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 server `.war` file.
2. Prepare a new deployment, installing servers from the new, customized `.war` file, starting with the instructions in "*Installing and Starting Servers*" 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.

Chapter 5
Reference

5.1. Command-Line Tool Reference

Name

upgrade.jar — upgrade AM using a configuration file

Synopsis

`upgrade.jar` {options}

Description

This executable jar file, openam-upgrade-tool-14.1.2.2.jar, lets you perform a silent upgrade on a deployed AM server by applying settings from a configuration file or using arguments. This capability allows you to include the `upgrade.jar` from a command line or in an upgrade script.

Options

The following options are supported.

`-f | --file *configuration-file*`

Upgrade a deployed AM web application archive using the specified configuration file. Upgrade configuration files are described in the sections below. Also, you can specify the system properties on the command line, instead of using the configuration file. See Example 2 below.

`--acceptLicense`

Auto-accept the software license agreement and suppress the display of the licence acceptance screen to the user. If the configuration file contains the `ACCEPT_LICENSES` property, it will have precedence over the command-line option.

`-? | --help`

Display the usage message.

Upgrade Configuration File

Base your configuration on the `sampleupgrade` file delivered with AM, and using the hints in this section, or the comments included in the file.

Upgrade Properties

SERVER_URL

URL to the web container where AM runs, such as `http://openam.example.com:8080`.

DEPLOYMENT_URI

URI where AM is deployed on the web container, such as `/openam`.

ACCEPT_LICENSES

Optional boolean property that can be set to always auto-accept the software license agreement and suppress displaying the license acceptance screen to the user. A value of `true` auto-accepts the license; any other value will be assumed to equal `false`, resulting in the presentation of the license. Default value is `false`. This property takes precedence over the `--acceptLicense` option, which can also be passed in to the application with the openam-upgrade-tool-14.1.2.2.jar file.

Examples

The following example shows a configuration file and the commands to upgrade a server using the `upgrade.jar`. The configuration file is saved as `/tmp/upgrade.txt`.

```
SERVER_URL=http://openam.example.com:8080
DEPLOYMENT_URI=/openam
ACCEPT_LICENSES=true
```

```
$JAVA_HOME/bin/java -jar ~/openam/tools/openam-upgrade-tool-14.1.2.2.jar \
 -f /tmp/upgrade.txt
```

The following example shows how to specify system properties with the `upgrade.jar`.

```
SERVER_URL=http://openam.example.com:8080
DEPLOYMENT_URI=/openam
ACCEPT_LICENSES=true
```

```
$JAVA_HOME/bin/java -jar ~/openam/tools/openam-upgrade-tool-14.1.2.2.jar \
 -DSERVER_URL=http://openam.example.com:8080 -DDEPLOYMENT_URI=/openam
```

The following example shows the use of the `--acceptLicense` option with the `upgrade.jar`.

```
SERVER_URL=http://openam.example.com:8080
DEPLOYMENT_URI=/openam
```

```
$JAVA_HOME/bin/java -jar ~/openam/tools/openam-upgrade-tool-14.1.2.2.jar \
 -DSERVER_URL=http://openam.example.com:8080 -DDEPLOYMENT_URI=/openam \
 --acceptLicense
```

Appendix A. Getting Support

For more information or resources about AM and ForgeRock Support, see the following sections:

A.1. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

- The ForgeRock Knowledge Base offers a large and increasing number of up-to-date, practical articles that help you deploy and manage ForgeRock software.

  While many articles are visible to community members, ForgeRock customers have access to much more, including advanced information for customers using ForgeRock software in a mission-critical capacity.

- ForgeRock product documentation, such as this document, aims to be technically accurate and complete with respect to the software documented. It is visible to everyone and covers all product features and examples of how to use them.

A.2. Using the ForgeRock.org Site

The ForgeRock.org site has links to source code for ForgeRock open source software, as well as links to the ForgeRock forums and technical blogs.

If you are a *ForgeRock customer*, raise a support ticket instead of using the forums. ForgeRock support professionals will get in touch to help you.

A.3. Getting Support and Contacting ForgeRock

ForgeRock provides support services, professional services, training through ForgeRock University, and partner services to assist you in setting up and maintaining your deployments. For a general overview of these services, see https://www.forgerock.com.

ForgeRock has staff members around the globe who support our international customers and partners. For details on ForgeRock's support offering, including support plans and service level agreements (SLAs), visit https://www.forgerock.com/support.

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	In general terms, a service exposing protected resources. 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.
Application type	Application types act as templates for creating policy applications. Application types define a preset list of actions and functional logic, such as policy lookup and resource comparator logic.

	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 *reference* 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	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	Group of AM servers configured the same way, accessed through a load balancer layer. The load balancer handles failover to provide service-level availability. The load balancer can also be used to protect AM services.
Standard metadata	Standard federation configuration information that you can share with other access management software.
Stateless Service	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. All AM services are stateless unless otherwise specified. See also Client-based sessions and CTS-based sessions.
Subject	Entity that requests access to a resource 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.
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.