Upgrade

This 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 ForgeRock product release levels. Release levels are identified by version number.

TIP

AM supports several versions of the web and Java agents, so you don't usually need to upgrade your agents when you upgrade AM.

You can find information on the supported AM versions in Web Agents or Java Agents.

Learn more about upgrading web and Java agents in the Web Agents documentation or the Java Agents documentation.

Before you upgrade

Review the tasks you need to complete before upgrading AM.

Upgrading servers

Learn, step by step, how to upgrade the AM core services.

Upgrading components

Review components and services that you

might need to upgrade or reconfigure.

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.

Supported upgrade paths

This table contains information about the supported upgrade paths to AM 7.2.2:

Upgrade paths

Version	Upgrade Supported?
AM 6.5.x	✔
AM 7.x	✔

For details, refer to the Ping Identity Product Support Lifecycle Policy in the *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.

IMPORTANT

IMPORTANT

The embedded DS server is *no longer supported in production*. If you have an existing site configured with an embedded DS server, you must migrate it to an external DS store before upgrading to AM 7.2.2.

The embedded DS server was deprecated in AM 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 7.2.2:

- 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 in AM 6.5? in the *Knowledge Base*.

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

  See When Adding New Servers in the DS documentation.

Plan 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. 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:

- Route around servers during downtime
- Back up the deployment
- Review REST API versions before upgrading
- Review Directory Services certificates before upgrading
- Customize before upgrading
- Plan for rollback
- Upgrade in a test environment first

Route 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 admin UI immediately after you deploy the new WAR file can therefore initiate the upgrade process.

Back 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 Back up configurations 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.

Review 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.

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 Protect against CSRF attacks.

You can configure AM's default REST API behavior. For more information, see Configuring the Default REST API Version.

Review 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 Key Management in the DS documentation.

Customize 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

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 or APIs.

TIP

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`).

Plan 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 Back up configurations.

Upgrade in 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.

Upgrade AM instances

Upgrading AM involves 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 go 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-14.1.3.27.jar`**, which is installed when you set up the configuration tools.

Perform the steps in the following procedure to upgrade AM:

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.

IMPORTANT

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.
- 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 AM 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 one of the following methods to migrate an embedded data store to an external store before attempting to upgrade AM:

  - 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 in AM 6.5? in the *Knowledge Base*.
  - Add a new external DS instance and replicate data from the embedded instance.

Follow the steps in When Adding New Servers in the DS documentation.

1. Before you start, make sure that you have planned your upgrade.
2. Prepare your customized AM server `.war` file.

   For more information, see Customize before upgrading.
3. **Back up your deployment**.

   For more information, see Back up the deployment.
4. Route client application traffic to another site during the upgrade.

   For more information, see Route 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 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 directenctest \
-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. You must update the identity 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 Set up directory schemas with LDIF.

     To update the user store schema, perform the following steps:
   - Change directories to the path where you have deployed the `openam.war` file.

     For example, `/path/to/tomcat/webapps/openam`.

- 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.

- 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`).
- 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.ld
if
# 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. (DS identity stores only) Update the identity store XML schema if you are upgrading from a version of AM earlier than 7.1.

NOTE

NOTE

This step is required to use the Save Retry Limit to User feature in the Retry Limit Decision node, which is enabled by default.

Even if you are not using the node now, ForgeRock recommends that you update the schema should you decide to use it in the future. If you cannot update the identity store schema at this point, you must disable the feature.

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

- Change directories to the path where you have deployed the `openam.war` file. For example, `/path/to/tomcat/webapps/openam`.
- Locate the `opendj_retry_limit_node_count.ldif` file in the `WEB-INF/template/ldif/opendj` path.

  You will apply more changes to the user schema in a later step. That step will also update your identity user store for this feature if you are not using DS.
- Update the identity store schema.

  For example:

```
$ /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 \
--continueOnError \
--bindDN uid=admin \
--bindPassword str0ngAdm1nPa55word \
/path/to/tomcat/webapps/openam/WEB-
INF/template/ldif/opendj/opendj_retry_limit_node_count.ldi
f
```

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

IMPORTANT

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

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

- The AM upgrade wizard uses three libraries that should be removed after upgrade for security reasons.

  When your upgrade is complete, remove the following .jar files from the `WEB-INF/lib` directory:

  - `click-extras-2.3.0.jar`
  - `click-nodeps-2.3.0.jar`
  - `velocity-1.7.jar`

  These files are used *only* by the install and upgrade wizards. Removing them will have no effect on your installed instance.

  You must also remove the references to `click-servlet` from the deployment descriptor file. Edit `/path/to/openam/WEB-INF/web.xml` to remove the following mappings:

```
<servlet>
    <servlet-name>click-servlet</servlet-name>
    <servlet-class>org.apache.click.ClickServlet</servlet-
class>
</servlet>

...

<servlet-mapping>
    <servlet-name>click-servlet</servlet-name>
    <url-pattern>*.htm</url-pattern>
</servlet-mapping>
```

- Use the `openam-upgrade-tool-14.1.3.27.jar` tool for an unattended upgrade:

  i. Install the `openam-upgrade-tool-14.1.3.27.jar` tool as described in To Set Up the Configuration Tools.

  A `sampleupgrade` file will be expanded in the directory where you install the tool.

  ii. Create a configuration file for the `openam-upgrade-tool-14.1.3.27.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
```

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

```
$ java -jar openam-upgrade-tool-14.1.3.27.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, refer to the reference documentation for upgrade.jar.

iv. Restart AM or the container where it runs.

12. 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");)
```

    Refer to Prepare configuration stores for more information about configuring AM configuration stores.

13. Update the identity store schema as follows:

    - Log into AM.
    - Go to **Realms** > ***Realm Name*** > **Datastores** > ***External User Store***.
    - Click **Load Schema** before saving, and click Save to apply your changes.
    - 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, refer to Update the schema in an external identity repository.

14. 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-ttlexpire.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/opendj/bin/ldapmodify \
--hostname 'cts.example.com' \
--port 1636 \
--useSsl \
--usePkcs12TrustStore /path/to/opendj/config/keystore \
--trustStorePasswordFile /path/to/opendj/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-ttlexpire.ldif
```

15. 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/opendj/push_2fa.ldif`
    - `/path/to/openam/WEB-INF/template/ldif/opendj/opendj_webauthndevices.ldif`
    - `/path/to/openam/WEB-INF/template/ldif/opendj/opendj_deviceprofiles.ldif`

    For example:

    ```
    $ /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 \
    ```

```
--continueOnError \
--bindDN uid=admin \
--bindPassword str0ngAdm1nPa55word \
/path/to/tomcat/webapps/openam/WEB-
INF/template/ldif/opendj/opendj_webauthndevices.ldif\
/path/to/tomcat/webapps/openam/WEB-
INF/template/ldif/opendj/opendj_deviceprofiles.ldif
```

IMPORTANT

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

In the AM admin UI, go to **Realms** > ***Realm Name*** > **Identity Stores** > ***Identity Store Name***. On the **User Configuration** tab, remove `webauthnDeviceProfilesContainer` from the **LDAP User Object Class** property, and 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 Set up directory schemas with LDIF.

16. Install a new version of the AM tools as described in Set up administration tools and in the ForgeRock Identity Platform Amster User Guide.

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

17. Review the information in Upgrade components and services and decide if you need to reconfigure any of AM's services or features.

NOTE

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 for all instances of the AM environment, go to **Configure > Server Defaults > Advanced** in the AM admin UI.
- 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 already configured, click on the pencil (🖉) button to edit it. When you are finished, click on the tick (✓) button.

Click **Save Changes**.

18. 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:

    a. Read the release notes for information about possible changes.

    b. Install an AM 7.2.2 test environment, and compare the scripts.

       New installations always have the current scripts.

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

Upgrade components and services

As part of planning your upgrade, consider that some changes in later AM versions will 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 the mandatory upgrade steps outlined in Upgrade AM instances, if you are using features described in these lists, you will need to perform additional upgrade

tasks:

▼ Changes in AM 7.1

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 Control the size of compressed JWTs.

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 Limit the size of the request body.

Pre-Approval for Redirection URIs Enforced

This change affects AM when acting as an OAuth 2.0 and OpenID Connect client.

If a redirection URI uses a scheme, host, or port that differs from that of AM, you must now add it to the global validation service to ensure that it is pre-approved. This is described in Success and failure redirection URLs.

Otherwise, AM rejects the URI, and redirection fails.

Subject Claim in Access and ID Tokens

The subject claim of access tokens and ID tokens has changed formats to ensure that it is locally unique. The new format is *not enforced after upgrading* to AM 7.1, but new installations default to it.

The `org.forgerock.security.oauth2.enforce.sub.claim.uniqueness` advanced server property controls the format of the `sub` claim.

Before enabling it, ensure that your clients can use the new `sub` claim format, or a combination of the `sub` and the `subname` claims.

The Retry Limit Decision node

The new **Save Retry Limit to User** feature in this node is enabled by default after upgrade and requires upgrading the identity store schema.

Ensure you update the schema following the instructions in Upgrade AM instances, or disable the feature. ForgeRock recommends keeping it enabled for security reasons.

Failure to take any of the actions will break the authentication journey for trees using this node.

One-Time Passwords in Authentication Nodes

One-time passwords created by the HOTP Generator node are now stored in the authentication tree's transient state.

Modify any custom authentication nodes or scripts used by the Scripted Decision node to retrieve the one-time passwords from the transient state after upgrading to AM 7.1.

▼ Changes in AM 7

User Profile Whitelist

The profile attribute allowlist controls the information returned to non-administrative users when accessing `json/user` endpoints.

Common profile attributes are allowlisted by default, but you need to add any custom attribute you want your non-administrative users to see. For more information, see Configure the user profile allowlist.

`/json/authenticate` Endpoint

When a client makes a call to the `/json/authenticate` endpoint appending a valid SSO token, AM returns the `tokenId` field **empty** if `HttpOnly` cookies are enabled. For example:

```
{
    "tokenId":"",
    "successUrl":"/openam/console",
    "realm":"/alpha"
}
```

Secure Authentication Tree State Secret ID

An AES 256-bit key called `directenctest` 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.

After upgrade, ensure that the `am.authn.trees.transientstate.encryption` 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.

The Embedded DS

The embedded DS can only be used for single AM instances, for test and demo purposes. Sites are not supported.

Sites using embedded DS servers must be migrated to external DS servers before upgrading.

SAML v2.0 Secrets

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 Configure secret stores after upgrade.

`goto` and `gotoOnFail` Query Parameter Redirection

Redirection URLs for authentication services, agents, and SAML v.2.0 must be configured in Success and failure redirection URLs if they are not in the same scheme, FQDN, and port as AM, or are not relative to AM's URL.

Web Agents of a Version Earlier than 5.6.3

Several properties that used to be configured as custom properties (`com.sun.identity.agents.config.freeformproperties`) 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.

To work around this issue, perform one of the following actions:

- 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

AM 7 changes the way the CTS reaper searches for expired tokens.

After upgrading, retune the CTS Reaper using the information in Reaper Search Size.

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 Secure cookie filter.

▼ Changes in AM 6.5.0.2 and 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.

To configure the `refresh token` grant type manually, see To Configure AM to Issue Refresh Tokens.

▼ Changes in AM 6.5

Recovery Codes

Recovery Codes are encrypted, and existing codes are no longer displayed to the user. For more information, see Upgrade 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 Configure 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 to the default used by previous versions of AM (`dc=openam`, `dc=forgerock`, `dc=org`).

Do not 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 Prepare a configuration store.

▼ Changes in AM 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.

Configure the user profile allowlist

AM 7 introduced a profile attribute allowlist.

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

Common profile attributes are allowlisted by default. You must add any custom attributes that you want non-administrative users to see.

The allowlist can be set globally, or per realm, in the user self-service service. To modify the list:

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

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

Configure secret stores after upgrade

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

IMPORTANT

- The upgrade process does not create all the demo key aliases created during a fresh install. This includes, for example, the aliases required for the IoT service. If you want to use services that require these demo aliases, add the corresponding keys and mappings after you have upgraded.
- Although 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 run the upgrade process**.

Follow these steps to make the secret stores available to other servers in the site:

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. Go 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.

IMPORTANT

When you upgrade from an EOSL version (for example, AM 6.0.x) to a supported version (AM 7.2.x and later), the upgrade process adds multiple keystore entries for each SAML signing certificate that was present in your default keystore. To work around this issue, remove the duplicate entries manually after the upgrade. This issue does not occur when you are upgrading from AM 6.5 or later.

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. Go 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.

   ▼ *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.

Upgrade 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:

Prevent AM from 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 you encrypt recovery codes.

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=tr
```

```
ue"
```

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. When preventing AM from encrypting the stored recovery codes, 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.

Upgrade JDBC audit event handlers

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

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.

Migrate legacy instances

Rather than upgrade legacy instances (running 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.

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 Install AM.
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.

Was this helpful?