FORGEROCK®

Upgrade Guide

ForgeRock Access Management 5

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, policy agents, and tools.

This guide is for anyone who needs to upgrade an ForgeRock Access Management 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™ is the only offering for access management, identity management, user-managed access, directory services, and an identity gateway, designed and built as a single, unified platform.

The platform includes the following components that extend what is available in open source projects to provide fully featured, enterprise-ready software:

- ForgeRock Access Management (AM)
- ForgeRock Identity Management (IDM)
- ForgeRock Directory Services (DS)
- ForgeRock Identity Gateway (IG)

Chapter 1
About Upgrading

This chapter covers common aspects of upgrading an OpenAM 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 OpenAM release.

Release levels, and how much change to expect in a maintenance, minor, or major release, are defined in Section A.1, "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 5:

Table 1.1. Upgrade Paths

Version	Upgrade Supported?
OpenAM 9.0.x	No
OpenAM 9.5.x	No
OpenAM 10.0.x	No
OpenAM 11.0.x	No
OpenAM 12.0.x	Yes
OpenAM 13.x.x	Yes
Access Management 5	Yes [a]

[a]

[a] **Caution**
Access Management is incompatible with SSO session tokens from OpenAM.
Storage and processing of SSO tokens changed in AM 5, meaning both stateful and stateless SSO sessions created in 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 re-authenticate.
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.

If your pre-AM 5 deployment relies on long-lived SSO session tokens, and re-authenticating your users will be problematic, raise a ForgeRock Support issue for more information and assistance regarding upgrading to AM 5 or later.

Note

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 OpenAM 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.
- These suggestions are true both for OpenAM server components, and also for policy agents.

To upgrade from an OpenAM server, use the Upgrade wizard. The OpenAM server Upgrade wizard appears when you replace a deployed OpenAM server `.war` file with a newer version and browse to the deployment URL. The Upgrade wizard brings the OpenAM configuration, including the version number, up to date with the new version. The CLI counterpart of the Upgrade wizard is openam-upgrade-tool-14.0.0.jar, which you install as described in Section 2.3.2, "Setting up Configuration Tools" in the *Installation Guide*.

1.3. Best Practices for Upgrades

Be prepared before you begin an upgrade, even if the upgrade is for a maintenance release.

1.3.1. Route Around Servers During Downtime

Upgrading servers takes at least one of your OpenAM 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 OpenAM 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.

1.3.2. 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 Section 8.1, "Backing Up and Restoring Configurations" in the *Setup and Maintenance Guide* is good for production environments.
- In preparation for upgrading OpenAM 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 OpenAM server in Apache Tomcat under `/openam`, you might take a file system backup of the following directories for each OpenAM server.

  - `/path/to/tomcat/webapps/openam/`
  - `~/openam/`
  - `~/.openamcfg/`
- When upgrading web policy agents, take a file system backup of the policy agent installation and configuration directories.

  When upgrading Java EE policy agents, it can be easier to uninstall the new version and reinstall the old version than to restore from file system backup.
- 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.

1.3.3. Apply Customization Before Upgrading

Before you upgrade OpenAM servers, prepare a .war file that contains any customizations you require.

Customizations include any changes you have made to the OpenAM server installation, such as the following.

- Plugins and extensions such as custom authentication modules, response attribute providers, post authentication plugins, SAML v2.0 attribute mappers, and OAuth 2.0 scope implementations.

  These are described in the Development Guide.

Important

If you are upgrading from OpenAM 12.x and you have custom authentication modules, you must upgrade their service definitions to contain `resourceName` attributes in the `Schema` and `SubSchema` elements.

For an example of a service definition compatible with AM 5, see Section 10.1.6, "The Sample Auth Service Configuration" in the *Authentication and Single Sign-On Guide*.

- Customized JSPs, redesigned login or service pages, additional CSS and visual content, and modified authentication module callback files.

  These are described in the UI Customization Guide.

- Any changes to OpenAM classes.

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

1.3.4. 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 OpenAM 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 Section 8.1, "Backing Up and Restoring Configurations" in the *Setup and Maintenance Guide*.

For web policy agents, you can roll back by restoring from file system backup. If you used configuration only available to newer agents, restore the pre-upgrade configuration before restarting the old agents.

For Java EE policy agents, uninstall the newer agents and reinstall the older agents, including the old configurations.

Chapter 2 Upgrading Servers

This chapter covers upgrade from core server 12.0.0 or later to the current version. For other components, see Chapter 3, "*Upgrading Components*".

OpenAM server upgrade relies on the Upgrade Wizard to make the necessary changes to the configuration store. You must then restart OpenAM or the container in which it runs. Even a version number change requires that you run the Upgrade Wizard, so needing to run the Upgrade Wizard says nothing about the significance of the changes that have been made to OpenAM. You must run the Upgrade Wizard even for maintenance releases.

Make sure you try upgrading OpenAM in a test environment before applying the upgrade in your production environment.

- Procedure 2.1, "To Upgrade From a Supported Version"
- Procedure 2.2, "To Complete Upgrade from OpenAM 13.0.x"

Procedure 2.1. To Upgrade From a Supported Version

Follow these steps to upgrade a site of servers. For information on the versions that are supported for upgrade, see Section 2.7, "Supported Upgrade Paths" in the *Release Notes*.

During the upgrade process, you must take the OpenAM servers in the site out of production, instead redirecting client application traffic elsewhere. This is required because upgrade involves making changes to OpenAM's configuration model. If the upgrade fails, you must be able to roll back before the configuration changes impact other sites.

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. Prepare your customized OpenAM server `.war` file.
2. **Back up your deployment**.
3. Route client application traffic to another site during the upgrade.
4. For servers in the site, stop OpenAM, or if necessary stop the container where OpenAM runs.

5. For servers in the site, 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.

6. For servers in the site, restart OpenAM or the container where it runs.

7. 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 OpenAM 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-14.0.0.jar` tool for an unattended upgrade:

     1. Install the `openam-upgrade-tool-14.0.0.jar` tool as described in Section 2.3.2, "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-14.0.0.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 OpenAM by using the tool with the properties file following this example:

        ```
        $ java -jar openam-upgrade-tool-14.0.0.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 OpenAM or the container where it runs.

8. If you installed OpenAM using an external directory server as the configuration store, add an access control instruction (ACI) to the external directory to give the OpenAM 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 Section 1.4.2, "Preparing an External Configuration Data Store" in the *Installation Guide* for more information about using an external directory server as the OpenAM configuration store.

9. If you want to configure the upgraded system for the Core Token Service (CTS), read Chapter 3, "*Implementing the Core Token Service*" in the *Installation Guide*. For a list of supported directory services, see the Section 2.5, "Data Store Requirements" in the *Release Notes*

10. Referral policies are not supported in AM 5. If your OpenAM deployment has referral policies, the following warning message will appear when you upgrade your server to AM 5:

    ```
    Referrals found that require removing
    ```

    OpenAM will take the following actions during the upgrade:

    - Removing all referral policies from your OpenAM configuration.

    - Copying resource types and policy sets associated with removed referral policies to the realms targeted by the referral policies.

For example, suppose you had an OpenAM 12 deployment with a referral policy in realm A, and that referral policy referred to policies in realm B. During an upgrade, OpenAM would delete the referral policy in realm A and copy all the resource types and policy sets associated with the deleted referral policy from realm A to realm B.

After upgrading to AM 5, you are responsible for reconfiguring AM so that policy evaluation that previously depended upon referrals continues to function correctly. You might need to take one or both of the following actions:

- Reconfiguring your policy agent with the realm and policy set that contain policies to be evaluated when that agent requests a policy decision from OpenAM. Previously, you might have configured the agent to use a realm that contained a referral policy. Because referral policies are not supported in AM 5, this is not possible.

  Note

  The agent configuration UI refers to a policy set as an application.

  For more information about configuring an agent with a realm and policy set, see Section 2.4.2, "Working With Realms and Policy Agents" in the *Setup and Maintenance Guide*.

- Copying or moving a policy or a group of policies. AM 5 has REST API endpoints that let you copy and move policies. This functionality might be helpful when migrating away from policy deployments that use referral policies. For more information about the REST endpoints that let you copy and move policies, see Section 2.2.6.6, "Copying and Moving Policies" in the *Authorization Guide*.

11. If you have post authentication plugins that expect state to be maintained by OpenAM 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 OpenAM to maintain state information in server memory throughout a session's duration for post authentication plugin module instances. When logout was triggered, OpenAM 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 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 Section 10.3.2, "Building Your Sample Post Authentication Plugin" in the *Authentication and Single Sign-On Guide*.

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

Procedure 2.2. To Complete Upgrade from OpenAM 13.0.x

If you configured one or more JDBC audit event handlers in OpenAM 13.0.x, make the following changes to the audit tables' schema:

1. Run the following command on Oracle databases that support OpenAM 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 OpenAM 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 OpenAM audit event handlers, review their schema.

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

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

Chapter 3
Upgrading Components

This chapter concerns upgrading policy agents, tools, and services.

3.1. Upgrading Web Policy Agents

To upgrade Web Policy agents, perform the following procedure:

Procedure 3.1. To Upgrade Web Policy Agents

1. Refer to the *OpenAM Web Policy Agent Release Notes* for information about changes and new policy agent functionality.

2. Back up the policy agent installation and configuration directories. For example:

   ```
   cp -r /path/to/web_agents/apache24_agent /path/to/backup
   cp -r /path/to/apache/httpd/conf /path/to/backup
   ```

   If the configuration if stored centrally in OpenAM, back it up as described in Section 8.1, "Backing Up and Restoring Configurations" in the *Setup and Maintenance Guide*.

3. Redirect client traffic away from the protected application.

4. Stop the webserver where the policy agent is installed.

5. Remove the old policy agent.

   For example, to remove an old web policy agent installed in Apache HTTP server, see Removing Apache Web Policy Agents in the *OpenAM Web Policy Agent Guide*. If the uninstall process has changed, refer to the version of the *Web Policy Agent Guide* that corresponds to your web policy agent.

6. Install the new policy agent.

For example, to install the new policy agent in Apache HTTP server, see Installing Web Policy Agents in Apache HTTP Server in the *OpenAM Web Policy Agent Guide*.

7. Start the webserver where the policy agent is installed.

8. Validate that the policy agent is performing as expected.

   For example, navigate to a protected page on the web site and confirm whether you can access it according to your configuration.

9. Allow client traffic to flow to the protected application.

3.2. Upgrading Java EE Policy Agents

To upgrade the Java EE Policy Agents, perform the following procedure:

Procedure 3.2. To Upgrade Java EE Policy Agents

1. Refer to the *OpenAM Java EE Policy Agent Release Notes* for information about changes and new policy agent functionality.

2. Back up the policy agent installation and configuration directories. For example:

```
cp -r /path/to/j2ee_agents/tomcat_v7_agent /path/to/backup
cp -r /path/to/tomcat/webapps/agentapp /path/to/backup
```

   If the configuration if stored centrally in OpenAM, back it up as described in Section 8.1, "Backing Up and Restoring Configurations" in the *Setup and Maintenance Guide*.

3. Redirect client traffic away from the protected application.

4. Stop the container where the policy agent is installed.

5. Remove the old policy agent.

   For example, to remove an old policy agent installed in Apache Tomcat, see Remove Tomcat Policy Agent Software in the *OpenAM Java EE Policy Agent User's Guide*. If the uninstall process has changed, refer to the version of the *Java EE Policy Agent User's Guide* that corresponds to your web policy agent.

6. Install the new policy agent.

   For example, to install a policy agent in Apache Tomcat, see Installing Java EE Agents in Apache Tomcat in the *OpenAM Java EE Policy Agent User's Guide*.

7. Start the container where the policy agent is installed.

8. Validate that the policy agent is performing as expected.

For example, navigate to a protected page on the web site and confirm whether you can access it according to your configuration.

9. Allow client traffic to flow to the protected application.

3.3. Upgrading Tools

To upgrade the tools, perform the following procedure:

Procedure 3.3. To Upgrade Tools

1. Install the new version of the tools as described in Section 2.3, "Installing and Using the Tools" in the *Installation Guide*.

2. Once the new tools are working, you can delete the old tools.

3.4. Upgrading User Self Services

This section covers upgrading user self-service features.

3.4.1. Upgrading the Keystore for User Self-Service

OpenAM'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, OpenAM deploys a JCEKS keystore that includes demo user self-service keys. This keystore is not configured as the OpenAM 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 to AM 5, see the following table:

Table 3.1. 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

Upgrading from:	Enabling JCEKS required?
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 OpenAM. When moving your keystore from JKS to JCEKS, you must also review your existing use of keys in OpenAM, 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 OpenAM'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 OpenAM, how to configure a JCEKS keystore, and how to create new user self-service keys, see Chapter 5, "*Setting Up Keys and Keystores*" in the *Setup and Maintenance Guide*.
- For more information about configuring user self-service keys in OpenAM, see Section 2.1, "Configuring the Signing and Encryption Key Aliases" in the *User Self Service Guide*.

3.4.2. Upgrading User Self-Service in Subrealms

AM 5 alters 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:

Procedure 3.4. 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

   `http://openam.example.com:8080/openam/XUI/?realm=${realm}#register/`

   Forgotten Password Confirmation Email URL

   `http://openam.example.com:8080/openam/XUI/?realm=${realm}#passwordReset/`

4. Save your changes.

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

Chapter 4
Migrating Legacy Servers

Rather than upgrade legacy servers (running OpenSSO or Sun Access Manager, or an OpenAM 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.

Procedure 4.1. To Upgrade A Legacy Deployment

1. Prepare your customized OpenAM server `.war` file.
2. Prepare a new deployment, installing servers from the new, customized `.war` file, starting with the instructions in Chapter 2, "*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 OpenAM using a configuration file

Synopsis

`upgrade.jar` {options}

Description

This executable jar file, openam-upgrade-tool-14.0.0.jar, lets you perform a silent upgrade on a deployed OpenAM 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 OpenAM 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 OpenAM, and using the hints in this section, or the comments included in the file.

Upgrade Properties

SERVER_URL

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

DEPLOYMENT_URI

URI where OpenAM 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-14.0.0.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-14.0.0.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-14.0.0.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-14.0.0.jar \
 -DSERVER_URL=http://openam.example.com:8080 -DDEPLOYMENT_URI=/openam \
 --acceptLicense
```

Appendix A. Getting Support

For more information or resources about OpenAM 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.
- ForgeRock core 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.

Core documentation therefore follows a three-phase review process designed to eliminate errors:

- Product managers and software architects review project documentation design with respect to the readers' software lifecycle needs.
- Subject matter experts review proposed documentation changes for technical accuracy and completeness with respect to the corresponding software.
- Quality experts validate implemented documentation changes for technical accuracy, completeness in scope, and usability for the readership.

The review process helps to ensure that documentation published for a ForgeRock release is technically accurate and complete.

Fully reviewed, published core documentation is available at http://backstage.forgerock.com/. Use this documentation when working with a ForgeRock Identity Platform release.

A.2. Joining the ForgeRock Community

Visit the Community resource center where you can find information about each project, download trial builds, browse the resource catalog, ask and answer questions on the forums, find community events near you, and find the source code for open source software.

A.3. Getting Support and Contacting ForgeRock

ForgeRock provides support services, professional services, classes 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, visit https://www.forgerock.com, or send an email to ForgeRock at info@forgerock.com.

Glossary

Access control	Control to grant or to deny access to a resource.
Account lockout	The act of making an account temporarily or permanently inactive after successive authentication failures.
Actions	Defined as part of policies, these verbs indicate what authorized subjects can do to resources.
Advice	In the context of a policy decision denying access, a hint to the policy enforcement point about remedial action to take that could result in a decision allowing access.
Agent administrator	User having privileges only to read and write policy agent profile configuration information, typically created to delegate policy agent profile creation to the user installing a policy agent.
Agent authenticator	Entity with read-only access to multiple agent profiles defined in the same realm; allows an agent to read web service profiles.
Application	In general terms, a service exposing protected resources. In the context of OpenAM policies, the application is a template that constrains the policies that govern access to protected resources. An application can have zero or more policies.
Application type	Application types act as templates for creating policy applications. 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	OpenAM authentication unit that handles one way of obtaining and verifying credentials.
Authorization	The act of determining whether to grant or to deny a principal access to a resource.
Authorization Server	In OAuth 2.0, issues access tokens to the client after authenticating a resource owner and confirming that the owner authorizes the client to access the protected resource. OpenAM can play this role in the OAuth 2.0 authorization framework.
Auto-federation	Arrangement to federate a principal's identity automatically based on a common attribute value shared across the principal's profiles at different providers.
Bulk federation	Batch job permanently federating user profiles between a service provider and an identity provider based on a list of matched user identifiers that exist on both providers.
Circle of trust	Group of providers, including at least one identity provider, who have agreed to trust each other to participate in a SAML v2.0 provider federation.
Client	In OAuth 2.0, requests protected web resources on behalf of the resource owner given the owner's authorization. OpenAM can play this role in the OAuth 2.0 authorization framework.
Conditions	Defined as part of policies, these determine the circumstances under which which a policy applies. Environmental conditions reflect circumstances like the client IP address, time of day, how the subject authenticated, or the authentication level achieved.

	Subject conditions reflect characteristics of the subject like whether the subject authenticated, the identity of the subject, or claims in the subject's JWT.
Configuration datastore	LDAP directory service holding OpenAM configuration data.
Cross-domain single sign-on (CDSSO)	OpenAM capability allowing single sign-on across different DNS domains.
Delegation	Granting users administrative privileges with OpenAM.
Entitlement	Decision that defines which resource names can and cannot be accessed for a given subject in the context of a particular application, which actions are allowed and which are denied, and any related advice and attributes.
Extended metadata	Federation configuration information specific to OpenAM.
Extensible Access Control Markup Language (XACML)	Standard, XML-based access control policy language, including a processing model for making authorization decisions based on policies.
Federation	Standardized means for aggregating identities, sharing authentication and authorization data information between trusted providers, and allowing principals to access services across different providers without authenticating repeatedly.
Fedlet	Service provider application capable of participating in a circle of trust and allowing federation without installing all of OpenAM on the service provider side; OpenAM lets you create Java Fedlets.
Hot swappable	Refers to configuration properties for which changes can take effect without restarting the container where OpenAM runs.
Identity	Set of data that uniquely describes a person or a thing such as a device or an application.
Identity federation	Linking of a principal's identity across multiple providers.
Identity provider (IdP)	Entity that produces assertions about a principal (such as how and when a principal authenticated, or that the principal's profile has a specified attribute value).
Identity repository	Data store holding user profiles and group information; different identity repositories can be defined for different realms.
Java EE policy agent	Java web application installed in a web container that acts as a policy agent, filtering requests to other applications in the container with policies based on application resource URLs.

Metadata	Federation configuration information for a provider.
Policy	Set of rules that define who is granted access to a protected resource when, how, and under what conditions.
Policy Agent	Agent that intercepts requests for resources, directs principals to OpenAM for authentication, and enforces policy decisions from OpenAM.
Policy Administration Point (PAP)	Entity that manages and stores policy definitions.
Policy Decision Point (PDP)	Entity that evaluates access rights and then issues authorization decisions.
Policy Enforcement Point (PEP)	Entity that intercepts a request for a resource and then enforces policy decisions from a PDP.
Policy Information Point (PIP)	Entity that provides extra information, such as user profile attributes that a PDP needs in order to make a decision.
Principal	Represents an entity that has been authenticated (such as a user, a device, or an application), and thus is distinguished from other entities. When a Subject successfully authenticates, OpenAM associates the Subject with the Principal.
Privilege	In the context of delegated administration, a set of administrative tasks that can be performed by specified subjects in a given realm.
Provider federation	Agreement among providers to participate in a circle of trust.
Realm	OpenAM unit for organizing configuration and identity information. Realms can be used for example when different parts of an organization have different applications and user data stores, and when different organizations use the same OpenAM deployment. Administrators can delegate realm administration. The administrator assigns administrative privileges to users, allowing them to perform administrative tasks within the realm.
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 OpenAM to return additional information in the form of "attributes" with the response to a policy decision.
Role based access control (RBAC)	Access control that is based on whether a user has been granted a set of permissions (a role).
Security Assertion Markup Language (SAML)	Standard, XML-based language for exchanging authentication and authorization data between identity providers and service providers.
Service provider (SP)	Entity that consumes assertions about a principal (and provides a service that the principal is trying to access).
Session	The interval that starts with the user authenticating through OpenAM and ends when the user logs out, or when their session is terminated. For browser-based clients, OpenAM manages user sessions across one or more applications by setting a session cookie. See also Stateful session and Stateless session.
Session high availability	Capability that lets any OpenAM 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 OpenAM after successful authentication. For a Stateful session, 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 OpenAM servers configured the same way, accessed through a load balancer layer. The load balancer handles failover to provide service-level availability. Use sticky load balancing based on `amlbcookie` values to improve site performance. The load balancer can also be used to protect OpenAM services.
Standard metadata	Standard federation configuration information that you can share with other access management software.
Stateful session	An OpenAM session that resides in the Core Token Service's token store. Stateful sessions might also be cached in memory on one or

	more OpenAM servers. OpenAM tracks stateful 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.
Stateless session	An OpenAM session for which state information is encoded in OpenAM and stored on the client. The information from the session is not retained in the CTS token store. For browser-based clients, OpenAM sets a cookie in the browser that contains the session information.
Subject	Entity that requests access to a resource When a subject successfully authenticates, OpenAM associates the subject with the Principal that distinguishes it from other subjects. A subject can be associated with multiple principals.
User data store	Data storage service holding principals' profiles; underlying storage can be an LDAP directory service, a relational database, or a custom `IdRepo` implementation.
Web policy agent	Native library installed in a web server that acts as a policy agent with policies based on web page URLs.