FORGEROCK®

Installation Guide

/ ForgeRock Access Management 6.5

Latest update: 6.5.5

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

Abstract

Guide showing you how to install ForgeRock® Access Management. ForgeRock Access Management provides authentication, authorization, entitlement, and federation software.

Table of Contents

Preface

This guide shows you how to install ForgeRock Access Management for access and federation management.

This guide covers the install, upgrade, and uninstall procedures that you theoretically perform only once per version. This guide aims to provide you with at least some idea of what happens behind the scenes when you perform the steps.

This guide is written for anyone installing AM to manage and federate access to web applications and web-based resources.

Almost anyone can learn something from this guide, though a background in access management and maintaining web application software can help. You do need some background in managing services on your operating systems and in your application servers. You can nevertheless get started with this guide, and then learn more as you go along.

Unless you are planning a throwaway evaluation or test installation, read the Release Notes before you get started.

About ForgeRock Identity Platform™ Software

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

Chapter 1

Preparing for Deployment

This chapter covers prerequisites for deploying AM software, including how to prepare your environment and how to set up your application server to run AM. At the end of the chapter you will deploy AM into a web container, ready for initial installation, in "*Installing and Starting Servers*".

Preparing the Environment

This section covers setting up the environment in which to run AM.

The topics covered in this section are:

- Preparing a Fully Qualified Domain Name
- Preparing a Java Environment
- Setting Maximum File Descriptors and Processes Per User
- Enabling RSA SecurID Support

For more information about supported operating systems and Java requirements, see "Operating System Requirements" and "Java Requirements" in the *Release Notes*.

Preparing a Fully Qualified Domain Name

AM requires that you provide a fully qualified domain name (FQDN) when you configure it. Before you set up AM, be sure that your system has an FQDN, such as `openam.example.com`. For evaluation purposes, you can give your system an alias using the `/etc/hosts` file on UNIX systems or `%SystemRoot%\system32\drivers\etc\hosts` on Windows. For production deployments, make sure the FQDN is properly assigned using DNS.

Do not use the hostname `localhost` for AM, not even for testing purposes. AM relies on browser cookies, which are returned based on the domain name. You can set the cookie domain name value to an empty string for host-only cookies or to any non-top level domain. For example, if you install AM and use `openam.example.com` as the host, you can set the cookie domain name as `example.com`.

Important

Do not configure a top-level domain as your cookie domain as browsers will reject them.

Top-level domains are browser-specific. Some browsers, like Firefox, also consider special domains like Amazon's web service (for example, ap-southeast-2.compute.amazonaws.com) to be a top-level domain.

Check the effective top-level domain list at *https://publicsuffix.org/list/effective_tld_names.dat* to ensure that you do not set your cookie to a domain in the list.

Preparing a Java Environment

AM software depends on a Java runtime environment. Check the output of the **java -version** command to make sure your version is supported according to "Java Requirements" in the *Release Notes*.

The suggestions in this section pertain to AM deployments with the following characteristics:

- The deployment has a dedicated DS server for the Core Token Service. The host running this directory server is a high-end server with a large amount of memory and multiple CPUs.
- The AM server is configured to use CTS-based sessions.

Important

It is important to keep your Java software up-to-date with the latest supported version. Make sure that your `JAVA_HOME` environment variable always points to the latest supported Java version.

Settings For Oracle Java Environments

When using an Oracle Java environment set at least the following options:

`-server`

Use `-server` rather than `-client`.

`-Xmx1g` (minimum)

AM requires at least a 1 GB heap. If you are including the embedded DS, AM requires at least a 2 GB heap, as 50% of that space is allocated to DS. Higher volume and higher performance deployments require additional heap space.

`-XX:MetaspaceSize=256m`

Set the metaspace memory size to 256 MB.

`-XX:MaxMetaspaceSize=256m`

Set the maximum metaspace memory size to 256 MB.

For additional JVM tuning and security recommendations, see "Tuning Java Virtual Machine Settings".

Settings For IBM Java Environments

When using an IBM Java environment, set at least the following options:

`-DamCryptoDescriptor.provider=IBMJCE`
`-DamKeyGenDescriptor.provider=IBMJCE`

Use the IBM Java Cryptography Extensions.

`-Xmx1g` **(minimum)**

AM requires at least a 1 GB heap. If you are including the embedded DS, AM requires at least a 2 GB heap, as 50% of that space is allocated to DS. Higher volume and higher performance deployments require additional heap space.

Settings for OpenJDK Java Environment

When using an OpenJDK Java environment set at least the following options.

`-Xmx1024m` **(minimum)**

AM requires at least a 1 GB heap. If you are including the embedded DS, AM requires at least a 2 GB heap, as 50% of that space is allocated to DS. Higher volume and higher performance deployments require additional heap space. Recommended: `-Xmx2048m`.

`-XX:MetaspaceSize=256m`

Set the initial metadata space size to 256 MB.

Settings for Configuring a JVM Proxy

To configure AM to make requests through a proxy server, set the following HTTP or HTTPS proxy-related options depending on the protocol configured between AM and the proxy:

HTTPS Options

`-Dhttps.proxyHost`

IP address or hostname of the proxy server. For example, `proxy.example.com`.

`-Dhttps.proxyPort`

Port number of the proxy server. For example, `8443`.

`-Dhttps.nonProxyHosts`

A pipe-separated (|) list of IP addresses or hostnames that should be reached directly, bypassing the proxy configuration. For example, `localhost|internal.example.com`.

Use wildcards (*) at the beginning or the end of the address or hostname. For example, `*.example.com` or `internal*`.

HTTP Options

`-Dhttp.proxyHost`

IP address or hostname of the proxy server. For example, `proxy.example.com`.

`-Dhttp.proxyPort`

Port number of the proxy server. For example, `8080`.

`-Dhttp.nonProxyHosts`

A pipe-separated (|) list of IP addresses or hostnames that should be reached directly, bypassing the proxy configuration. For example, `localhost|internal.example.com`.

Use wildcards (*) at the beginning or the end of the address or hostname. For example, `*.example.com` or `internal*`.

Note

ForgeRock's ClientHandler code, such as the Google reCAPTCHA user self-service feature or the social authentication providers, require setting the advanced server property `org.forgerock.openam.httpclienthandler.system.proxy.enabled` to `true` to enable proxy support.

Tuning Java Virtual Machine Settings

This section gives some initial guidance on configuring the JVM for running AM. These settings provide a strong foundation to the JVM before a more detailed garbage collection tuning exercise, or as best practice configuration for production:

Heap Size Settings

JVM Parameters	Suggested Value	Description
`-Xms` & `-Xmx`	At least 1 GB (2 GB with embedded DS), in production environments at least 2 GB to 3 GB. This setting depends on the available physical memory, and on whether a 32- or 64-bit JVM is used.	-
`-server`	-	Ensures the server JVM is used
`-XX:MetaspaceSize` & `-XX:MaxMetaspaceSize`	Set both to 256 MB	Controls the size of the metaspace in the JVM
`-Dsun.net.client.defaultReadTimeout`	60000	Controls the read timeout in the Java HTTP client implementation

JVM Parameters	Suggested Value	Description
		This applies only to the Sun/ Oracle HotSpot JVM.
`-Dsun.net.client.defaultConnectTimeout`	High setting: 30000 (30 seconds)	Controls the connect timeout in the Java HTTP client implementation When you have hundreds of incoming requests per second, reduce this value to avoid a huge connection queue. This applies only to the Sun/ Oracle HotSpot JVM.

Security Settings

JVM Parameters	Suggested Value	Description
`-Dhttps.protocols`	`TLSv1.2`	Controls the protocols used for outbound HTTPS connections from AM. Specify one or more of the following values, separated by commas: • TLSv1 • TLSv1.1 • TLSv1.2 • TLSv1.3[a] This setting applies only to Sun/Oracle Java environments.
`-Dorg.forgerock.openam.ldap.secure.protocol.version`	`TLSv1.2`	Controls the protocol AM uses to connect to various external resources. Specify one or more of the following values, separated by commas: • TLSv1 • TLSv1.1 • TLSv1.2 • TLSv1.3[b]

[a]Supported from JDK 11.
[b]Supported from JDK 11.

Note

For `-Dhttps.protocols`, specify the protocol version(s) Java clients can use to connect to AM.

For `-Dorg.forgerock.openam.ldap.secure.protocol.version`, see "Securing Communications" for a list of external resources to which communication is affected.

Specify a single protocol if AM will only use that protocol when connecting to affected external resources. For example, a value of `TLSv1.2` configures AM to only use the TLSv1.2 protocol to connect.

Specify a comma-separated list with multiple protocols if AM will use the most secure protocol supported by the external resources. For example, if you are using at least JDK 11 you could specify a value of `TLSv1,TLSv1.1,TLSv1.2,TLSv1.3`, which configures AM to attempt to use the TLSv1.3 protocol to connect to external configuration and user data stores. If a TLSv1.3 connection is not supported, AM attempts to use TLSv1.2 to connect, then TLSv1.1, and if still not supported, AM uses TLSv1.

Garbage Collection Settings

JVM Parameters	Suggested Value	Description
`-verbose:gc`	-	Verbose garbage collection reporting
`-Xloggc:`	`$CATALINA_HOME/logs/gc.log`	Location of the verbose garbage collection log file
`-XX:+PrintClassHistogram`	-	Prints a heap histogram when a SIGTERM signal is received by the JVM
`-XX:+PrintGCDetails`	-	Prints detailed information about garbage collection
`-XX:+PrintGCTimeStamps`	-	Prints detailed garbage collection timings
`-XX:+HeapDumpOnOutOfMemoryError`	-	Out of Memory errors generate a heap dump automatically
`-XX:HeapDumpPath`	`$CATALINA_HOME/logs/heapdump.hprof`	Location of the heap dump
`-XX:+UseConcMarkSweepGC`	-	Use the concurrent mark sweep garbage collector
`-XX:+CMSClassUnloadingEnabled`	-	Allow class unloading during CMS sweeps

Setting Maximum File Descriptors and Processes Per User

AM is not file-descriptor intensive. However, each DS instance in your environment should have access to, at least, 65,536 file descriptors to handle multiple client connections.

Ensure that every DS instance is allocated enough file descriptors. For example, use the **ulimit -n** command to check the limits for a particular user:

```
$ su - opendj
$ ulimit -n
65536
```

The embedded DS runs inside the AM process space. When AM is configured with the embedded DS instance, you must ensure that the user running the AM processes has, at least, 65,536 file descriptors.

It may also be neccessary to increase the number of processes available to the user running the AM processes.

For example, use the **ulimit -u** command to check the process limits for a user:

```
$ su - openam
$ ulimit -u
2048
```

Important

Before increasing the file descriptors for the DS instance, ensure that the total amount of file descriptors configured for the operating system is higher than 65,536.

Otherwise, if the DS instance uses all of the file descriptors, the operating system will run out of file descriptors. This may prevent other services from working, including those required for logging in the system.

Refer to your operating system's documentation for instructions on how to display and increase the file descriptors or process limits for the operating system and for a given user.

For more information about setting up your environment for DS, see Choosing an Operating System in the *Directory Services 6.5 Install Guide*.

Enabling RSA SecurID Support

To use the SecurID authentication module, you must first build an AM war file that includes the supporting library, for example `authapi-2005-08-12.jar`, which you must obtain from RSA. The `authapi-2005-08-12.jar` file also requires a dependency file, `crypto.jar`, which you can also obtain from RSA.

1. Unpack the AM `.war` file. For example:

```
$ mkdir /tmp/openam
$ cd /tmp/openam/
$ jar -xf ~/Downloads/openam/AM-6.5.5.war
```

2. Obtain the `authapi.jar` (for example, `authapi-2005-08-12.jar`) and its dependency file, `crypto.jar` from RSA. Then, copy `authapi-2005-08-12.jar` into the `WEB-INF/lib` directory. For example:

```
$ cp /path/to/authapi-2005-08-12.jar WEB-INF/lib/
```

3. Pack up the AM `.war` file. For example:

   ```
   $ jar -cf ../openam.war *
   ```

4. Deploy the new `.war` file. For more information, see "Deploying".

   In this example, the `.war` file to deploy is `/tmp/openam.war`.

Preparing a Web Application Container

This section covers setting up a web application container in which to run AM.

The topics covered in this section are:

- Preparing Apache Tomcat
- Preparing for JBoss and WildFly
- Preparing Oracle WebLogic
- Preparing IBM WebSphere
- Enabling CORS Support
- Preparing AES Key Wrap Encryption

Note

If a Java Security Manager is enabled for your web application container, add permissions before installing AM.

For a list of supported web application containers, see "Web Application Container Requirements" in the *Release Notes*.

Preparing Apache Tomcat

AM examples often use Apache Tomcat (Tomcat) as the deployment container. Tomcat is installed on `openam.example.com`, and listens on the default ports without a Java Security Manager enabled.

JVM start up

AM core services require a minimum JVM heap size of 1 GB, and a metadata space size of up to 256 MB. If you are including the embedded DS, AM requires at least a 2 GB heap, as 50% of that space is allocated to DS. See "Preparing a Java Environment" for details.

Set the `CATALINA_OPTS` environment variable in Tomcat's start-up script or service with the appropriated tuning for your environment. For example:

```
CATALINA_OPTS="-server -Xmx2g -XX:MetaspaceSize=256m -XX:MaxMetaspaceSize=256m"
```

Some versions of Internet Explorer and Microsoft Edge support the `Expires` header attribute instead of the `Max-Age` header attribute, which may cause SAML 2.0 and agent logout sequences to fail.

If you have set the `org.apache.catalina.STRICT_SERVLET_COMPLIANCE` Tomcat property to `true`, add the `org.apache.tomcat.util.http.ServerCookie.ALWAYS_ADD_EXPIRE` property to Tomcat's start-up sequence to add the `Expires` attribute to the headers:

```
CATALINA_OPTS="-server -Xmx2g -XX:MetaspaceSize=256m -XX:MaxMetaspaceSize=256m \
                -Dorg.apache.tomcat.util.http.ServerCookie.ALWAYS_ADD_EXPIRES=true"
```

Slashes in Resource Names

Some AM resources have names that can contain slash characters (/), for example, in policy names, application names, and SAML v2.0 entities. These slash characters can cause unexpected behavior when running AM on Tomcat.

One possible workaround is to configure Tomcat to allow encoded slash characters by adding the `org.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH=true` property to the `CATALINA_OPTS` variable; however, this is not recommended for production deployments (see the warning below). For example:

```
CATALINA_OPTS= "-server -Xmx2g -XX:MetaspaceSize=256m -XX:MaxMetaspaceSize=256m \
                -Dorg.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH=true"
```

Warning

It is strongly recommended that you do *not* enable `org.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH` when running AM in production as it introduces a security risk.

For more information, see How do I safely enable the org.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH setting in AM/OpenAM (All Versions)? in the *ForgeRock Knowledge Base*.

Cookie Domains

You can set the cookie domain name value to an empty string for host-only cookies or to any non-top level domain. For example, if you install AM and use `openam.example.com` as the host, you can set the cookie domain name as `example.com`. For information about configuring the cookie domain during installation, see "To Custom Configure an Instance".

Encoding and Security

ForgeRock recommends that you edit the Tomcat <Connector> configuration to set `URIEncoding="UTF-8"`. UTF-8 URI encoding ensures that URL-encoded characters in the paths of URIs are correctly decoded by the container. This is particularly useful if your applications use the AM REST APIs and some identifiers, such as user names, contain special characters.

You should also ensure the `sslProtocol` property is set to `TLS`, which disables the potentially vulnerable SSL v3.0 protocol.

<Connector> configuration elements are found in the configuration file, `/path/to/tomcat/conf/server.xml`. The following excerpt shows an example <Connector> with the `URIEncoding` and `sslProtocol` attributes set appropriately:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
 maxThreads="150" scheme="https" secure="true"
 clientAuth="false" sslProtocol="TLS" URIEncoding="UTF-8" />
```

Embedded DS

When you set up AM with the embedded DS, make sure that Tomcat's `autoDeploy` attribute is set to `false`. If `autoDeploy` is set to `true`, the host dynamically deploys and updates any web application, for example, when a new `.war` file is dropped into the `appBase`.

Preparing for JBoss and WildFly

You can deploy AM on JBoss AS, JBoss EAP, and WildFly. The procedures listed here provide steps for configuring JBoss AS, JBoss EAP, and WildFly for AM.

After configuring JBoss or WildFly, you then prepare AM for deployment by making a few changes to the contents of the AM `.war` archive.

- "To Prepare JBoss or WildFly"
- "To Prepare for JBoss and WildFly"

To Prepare JBoss or WildFly

1. Stop JBoss or WildFly.
2. The default JVM settings do not allocate sufficient memory to AM. This step shows one method that you can use to modify the JVM settings. For other methods, see either the JBoss Application Server Official Documentation Page or the JVM Settings page in the WildFly documentation

   a. Open the `standalone.conf` file in the `/path/to/jboss/bin` directory for JBoss or WildFly in standalone mode.

   b. Check the JVM settings associated with `JAVA_OPTS`.

      Change the JVM heap size to `-Xmx1g`. The default JVM heap size for some versions of JBoss might already exceed the recommended value. If you are using the embedded version of DS, the minimum heap size may be higher. For details on the JVM options to use, see "Preparing a Java Environment".

      Change the metaspace size to `-XX:MaxMetaspaceSize=256m` if the default size does not exceed this amount.

   c. Set the following JVM `JAVA_OPTS` setting in the same file:

```
-Dorg.apache.tomcat.util.http.ServerCookie.ALWAYS_ADD_EXPIRES=true
```

Verify that the headers include the `Expires` attribute rather than only `Max-Age`, as some versions of Internet Explorer and Microsoft Edge do not support `Max-Age`.

3. Set up Wildfly for Social Authentication, by performing the following steps:

    a. Ensure the Wildfly server is running.

    b. Go to the Wildfly Path.

    c. In the `$JBOSS_HOME/bin` directory, run the `jboss-cli.sh` script file:

    ```
    $ ./bin/jboss-cli.sh
    ```

    d. Type "connect" to connect to the server.

    e. Enable use of the equals (**=**) symbol in cookies by running the following command:

    For example:

    ```
    [standalone@localhost:9990 /] /subsystem=undertow/server=default-server/
    http-listener=default:write-attribute(name=allow-equals-in-cookie-value,
    value=true)
    {
       "outcome" => "success",
       "response-headers" => {
          "operation-requires-reload" => true,
          "process-state" => "reload-required"
        }
    }
    ```

    f. Restart Wildfly.

4. Now deploy the `openam.war` file into the appropriate deployment directory. The directory varies depending on whether you are running in standalone or domain mode.

To Prepare for JBoss and WildFly

To prepare AM to run with JBoss or WildFly, you should make some changes to the default AM `war` file. JBoss and WildFly deploy applications from different temporary directories every time you restart the container, which would require reconfiguring AM. To avoid problems, change the AM `war` file as follows:

1. If you have not already done so, create a temporary directory and expand the `AM-6.5.5.war` file. For example:

```
$ cd /tmp
$ mkdir /tmp/openam ; cd /tmp/openam
$ jar xvf ~/Downloads/AM-6.5.5.war
```

2. Locate the `bootstrap.properties` file in the `WEB-INF/classes` directory of the expanded `war` archive. Update the `# configuration.dir=` line in this file to specify a path with read and write permissions, and then save the change.

```
# This property should also be used when the system user that
# is running the web/application server process does not have
# a home directory. i.e. System.getProperty("user.home") returns
# null.

configuration.dir=/my/readwrite/config/dir
```

3. If you are deploying AM on JBoss AS or JBoss EAP, remove the `jboss-all.xml` file from the `WEB-INF` directory of the expanded `war` archive.

   Be sure *not* to remove this file if you are deploying AM on WildFly.

4. If you are deploying AM on Wildfly 12, remove the `jul-to-slf4j-1.7.21.jar` file from the `WEB-INF/lib` directory of the expanded `war` archive.

5. Rebuild the `openam.war` file.

```
$ jar cvf ../openam.war *
```

6. If you plan to deploy multiple cookie domains with WildFly, you must configure the `com.sun.identity.authentication.setCookieToAllDomains` property after you have installed the AM server. See "Handling Multiple Cookie Domains When Using Wildfly" for more information.

Preparing Oracle WebLogic

To deploy AM in Oracle WebLogic, perform the following steps:

1. Update the JVM options as described in "Preparing a Java Environment".

2. Customize the `AM-6.5.5.war` file as described in "To Prepare for Oracle WebLogic".

To Prepare for Oracle WebLogic

To prepare AM to run in Oracle WebLogic, you must change the AM `war` file. Be sure to make these changes whenever you deploy a new `war` file, or as part of an AM upgrade.

Prepare for deployment in Oracle WebLogic as follows:

1. Create a temporary directory and expand the `AM-6.5.5.war` file. For example:

```
$ cd /tmp
$ mkdir /tmp/openam ; cd /tmp/openam
$ jar xvf ~/Downloads/AM-6.5.5.war
```

2. Ensure the installer is able to locate configuration files by specifying a folder which the user running Oracle WebLogic can read and write:

   a. Locate the `bootstrap.properties` file in the `WEB-INF/classes` directory of the expanded `war` file.

   b. Update the `# configuration.dir=` line in the `bootstrap.properties` file to specify a path with read and write permissions. For example:

```
# This property should also be used when the system user that
# is running the web/application server process does not have
# a home directory. i.e. System.getProperty("user.home") returns
# null.

configuration.dir=/my/readwrite/config/dir
```

   If installing on Windows, the specified path should have slashes **/** and not backslashes \.

3. Rebuild the `AM-6.5.5.war` file:

```
$ jar cvf ../Edited-AM-6.5.5.war *
```

4. Ensure Oracle WebLogic passes responsibility for account management over to AM, as follows:

   a. Ensure Oracle WebLogic is not running.

   b. Navigate to the Oracle WebLogic `config.xml` configuration file for the domain in which AM runs.

   For example, `$HOME/Oracle/Middleware/Oracle_Home/user_projects/domains/base_domain/config/config.xml`.

   c. In the `<security-configuration>` section, add the following line before the closing `</security-configuration>` tag:

```
<enforce-valid-basic-auth-credentials>false</enforce-valid-basic-auth-credentials>
```

   For example:

```
    <enforce-valid-basic-auth-credentials>false</enforce-valid-basic-auth-credentials>
</security-configuration>
```

5. Configure the Oracle WebLogic domain to use the canonical path of resource files by performing the following steps:

   a. In the Oracle WebLogic console, in the Domain Structure panel, click the name of the domain. For example, `base_domain`.

   b. On the Configuration tab, click the Web Applications tab, and then enable the Archived Real Path Enabled property.

   c. Save your changes, and restart Oracle WebLogic for them to take effect.

6. You can now deploy your `Edited-AM-6.5.5.war` in Oracle WebLogic.

Preparing IBM WebSphere

To deploy AM in IBM WebSphere, perform the following steps:

1. Update the JVM options as described in "Preparing a Java Environment".
2. Customize the `AM-6.5.5.war` file as described in "To Prepare for Oracle WebLogic".
3. After deploying AM, configure WebLogic as described in "To Prepare WebSphere".

To Prepare for IBM WebSphere

To prepare AM to run in WebSphere, change the AM `war` file to ensure that the AM upgrade process is able to find the AM configuration files. Be sure to make this change whenever you deploy a new `war` file as part of an AM upgrade.

Change the AM `war` file as follows:

Note

If installing on Windows, the specified paths should have slashes **/** and not backslashes \.

1. Create a temporary directory and expand the `AM-6.5.5.war` file. For example:

   ```
   $ cd /tmp
   $ mkdir /tmp/openam ; cd /tmp/openam
   $ jar xvf ~/Downloads/AM-6.5.5.war
   ```

2. Locate the `bootstrap.properties` file in the `WEB-INF/classes` directory of the expanded `war` file.

   Update the `# configuration.dir=` line in the `bootstrap.properties` file to specify a path with read and write permissions. For example:

   ```

   # This property should also be used when the system user that
   # is running the web/application server process does not have
   # a home directory. i.e. System.getProperty("user.home") returns
   # null.

   configuration.dir=/my/readwrite/config/dir
   ```

3. (Optional) If you are using an IBM JDK, replace the default `WEB-INF/template /keystore/keystore.jceks` keystore file with one generated using the IBM JDK, by performing the following steps:

   a. Generate a new, empty `keystore.jceks` keystore file in IBM JDK format:

      ```
      $ keytool -genkey -storetype jceks -keystore keystore.jceks -storepass changeit -keypass changeit
      ```

   b. Copy the new `keystore.jceks` keystore file into the expanded .war file, overwriting the existing `WEB-INF/template/keystore/keystore.jceks` keystore file:

```
$ cp keystore.jceks /tmp/openam/WEB-INF/template/keystore/keystore.jceks
```

4. Rebuild the `AM-6.5.5.war` file:

```
$ jar cvf ../AM-6.5.5.war *
```

To Prepare WebSphere

In addition to preparing the AM .war file, perform the following steps to configure WebSphere for AM *after you deploy AM into WebSphere*:

1. Load classes from AM bundled libraries before loading classes from libraries delivered with WebSphere:

   a. In the WebSphere administration console, navigate to Application > Application Type > WebSphere enterprise applications > *AM Name* > Class loading and update detection.

   b. Set Class loader order > Classes loaded with local class loader first (parent last).

   c. Ensure that the value of the *WAR class loader policy* property is set to the default value: `Class loader for each WAR file in application`.

   d. Save your work.

2. (Optional) If your environment uses SOAP, perform the following steps to add SOAP-related properties to the JVM:

   a. In the WebSphere administration console, select Servers.

   b. Expand Server Type, and select WebSphere application servers.

   c. Select your WebSphere server name.

   d. Expand Java and Process Management, and select Process Definition.

   e. Under the Additional Properties section, select Java Virtual Machine.

   f. Locate the Generic JVM argument box and add the following properties:

   ```
   -
   Djavax.xml.soap.MessageFactory=com.sun.xml.internal.messaging.saaj.soap.ver1_1.SOAPMessageFactory1_1Impl
   -Djavax.xml.soap.SOAPFactory=com.sun.xml.internal.messaging.saaj.soap.ver1_1.SOAPFactory1_1Impl
   -
   Djavax.xml.soap.SOAPConnectionFactory=com.sun.xml.internal.messaging.saaj.client.p2p.HttpSOAPConnectionFact
   -Djavax.xml.soap.MetaFactory=com.sun.xml.internal.messaging.saaj.soap.SAAJMetaFactoryImpl
   -Dcom.ibm.websphere.webservices.DisableIBMJAXWSEngine=true
   ```

   g. Save your work.

Enabling CORS Support

Cross-origin resource sharing (CORS) allows requests to be made across domains from user agents. AM supports CORS, but CORS is not enabled by default.

To enable CORS support, edit the deployment descriptor file before deploying AM. CORS support is implemented as a servlet filter, and so you add the filter's configuration to the deployment descriptor file.

To Enable CORS Support

1. If you have not yet deployed AM:

   a. Unpack the AM `.war` file. For example:

   ```
   $ mkdir /tmp/openam
   $ cd /tmp/openam/
   $ jar -xf ~/Downloads/openam/AM-6.5.5.war
   ```

   b. Open the deployment descriptor file `WEB-INF/web.xml` in a text editor.

2. If you have already deployed AM:

   - Open the deployment descriptor file `web.xml` in a text editor. The location of the file depends on your web application container, for example in Tomcat it might be located at: `/path/to/tomcat/webapps/openam/WEB-INF/web.xml`.

3. In the deployment descriptor file, add a `<filter-mapping>` element containing the name and a URL pattern for the filter. The URL pattern specifies the endpoints to which AM applies the CORS filter.

   To enable CORS support for all endpoints, use the following example:

   ```
   <filter-mapping>
       <filter-name>CORSFilter</filter-name>
       <url-pattern>/*</url-pattern><!-- CORS support for all endpoints -->
   </filter-mapping>
   ```

   To enable CORS support for individual endpoints instead of all endpoints, add multiple `<filter-mapping>` elements, one for each endpoint:

```
<filter-mapping>
  <filter-name>CORSFilter</filter-name>
  <url-pattern>/uma/*</url-pattern>
</filter-mapping>

<filter-mapping>
  <filter-name>CORSFilter</filter-name>
  <url-pattern>/json/*</url-pattern>
</filter-mapping>

<filter-mapping>
  <filter-name>CORSFilter</filter-name>
  <url-pattern>/oauth2/*</url-pattern>
</filter-mapping>
```

4. In the deployment descriptor file, add a `<filter>` element to configure the filter.

   The available parameters for the `<filter>` element are as follows:

   `<filter-name>`

   Specifies the name for the filter. Must match the name specified in the `<filter-mapping>` elements.

   `<filter-class>`

   Specifies the Java class the implements the CORS filter. Should be set to the default `org.forgerock.openam.cors.CORSFilter`.

   `methods` (required)

   A comma-separated list of HTTP methods allowed when making CORS requests to AM.

   Example: `GET,POST,PUT,PATCH,OPTIONS,DELETE`

   `origins` (required)

   A comma-separated list of the origins allowed when making CORS requests to AM. Wildcards are not supported - each value should be an exact match for the origin of the CORS request.

   Example: `http://example.com,https://example.org:8433`

> **Tip**
>
> During development you may not be using fully-qualified domain names as the origin of a CORS request, for example using the `file://` protocol locally. If so, you can add these non-FQDN origins to the list. For example, `http://example.com,https://example.org:8433,file://,null`.

`allowCredentials` (optional)

Whether to take allow requests with credentials in either HTTP cookies or HTTP authentication information. Accepts `false` (the default) or `true`.

Set to `true` if you send `Authorization` headers as part of the CORS requests, or need to include information in cookies when making requests.

`headers` (optional)

A comma-separated list of request header names allowed when making CORS requests to AM.

Example: `iPlanetDirectoryPro,X-OpenAM-Username,X-OpenAM-Password,Accept-API-Version,Content-Type,If-Match,If-None-Match`

By default, the following simple headers are explicitly allowed:

- `Cache-Control`
- `Content-Language`
- `Expires`
- `Last-Modified`
- `Pragma`

If you do not specify a value for this property, the presence of any header in the CORS request, other than the simple headers listed above, will cause the request to be rejected.

Headers commonly used when accessing an AM server include the following:

Commonly Used Headers

Header	Information
`iPlanetDirectoryPro`	Used for session information. See "Implementing Client-Based Sessions" in the *Authentication and Single Sign-On Guide*.
`X-OpenAM-Username`,`X-OpenAM-Password`	Used to pass credentials in REST calls that use the HTTP POST method.

Header	Information
	See "Authentication and Logout using REST" in the *Authentication and Single Sign-On Guide*.
`Accept-API-Version`	Used to request a specific AM endpoint version. See "REST API Versioning" in the *Authentication and Single Sign-On Guide*.
`Content-Type`	Required for cross-origin calls to AM REST API endpoints.
`If-Match`,`If-None-Match`	Used to ensure the correct version of a resource will be affected when making a REST call, for example when updating an UMA resource. See "To Update an UMA Resource" in the *User-Managed Access (UMA) 2.0 Guide* .

Caution

If you need to accept all origins by allowing the use of `Access-Control-Allowed-Origin=*` headers, do not allow `Content-Type` headers. Allowing the use of both types of headers exposes AM to cross-site request forgery (CSRF) attacks.

`expectedHostname` (optional)

The name of the host expected in the `Host` header of CORS requests to AM. The request will be refused if the specified value does not match.

If not specified, any host value is accepted.

If the AM server is behind a load-balancer, specify the public name of the load balancer.

Example: `openam.example.com:8080`

`exposeHeaders` (optional)

A comma-separated list of response header names that AM returns in the `Access-Control-Expose-Headers` header.

User agents can make use of any headers that are listed in this property, as well as the simple response headers, which are as follows:

- `Cache-Control`
- `Content-Language`
- `Expires`
- `Last-Modified`

- `Pragma`
- `Content-Type`

User agents must filter out all other response headers.

Example: `Access-Control-Allow-Origin,Access-Control-Allow-Credentials,Set-Cookie`

`maxAge` (optional)

The maximum length of time that the browser is allowed to cache the pre-flight response, in seconds.

The default is `600`.

The following is an example excerpt from a configured `web.xml` file that could be used during testing and development:

```
<filter-mapping>
  <filter-name>CORSFilter</filter-name>
  <url-pattern>/uma/*</url-pattern>
</filter-mapping>

<filter-mapping>
  <filter-name>CORSFilter</filter-name>
  <url-pattern>/json/*</url-pattern>
</filter-mapping>

<filter-mapping>
  <filter-name>CORSFilter</filter-name>
  <url-pattern>/oauth2/*</url-pattern>
</filter-mapping>

<filter>
  <filter-name>CORSFilter</filter-name>
  <filter-class>org.forgerock.openam.cors.CORSFilter</filter-class>
  <init-param>
    <param-name>methods</param-name>
    <param-value>POST,PUT,OPTIONS</param-value>
  </init-param>
  <init-param>
    <param-name>origins</param-name>
    <param-value>http://localhost:8000,null,file://,https://example.org:8433</param-value>
  </init-param>
  <init-param>
    <param-name>allowCredentials</param-name>
    <param-value>true</param-value>
  </init-param>
  <init-param>
    <param-name>headers</param-name>
    <param-value>X-OpenAM-Username,X-OpenAM-Password,X-Requested-With,Accept,iPlanetDirectoryPro</
param-value>
  </init-param>
  <init-param>
```

```
    <param-name>expectedHostname</param-name>
    <param-value>openam.example.com:8080</param-value>
  </init-param>
  <init-param>
    <param-name>exposeHeaders</param-name>
    <param-value>Access-Control-Allow-Origin,Access-Control-Allow-Credentials,Set-Cookie</param-value>
  </init-param>
  <init-param>
    <param-name>maxAge</param-name>
    <param-value>1800</param-value>
  </init-param>
</filter>
```

5. Save your changes.

6. If you have not yet deployed AM:

   a. Pack up the AM `.war` file to deploy.

      ```
      $ jar -cf ../openam.war *
      ```

   b. Deploy the new `.war` file.

      In this example, the `.war` file to deploy is `/tmp/openam.war`.

      For more information, see "Deploying".

7. If you have already deployed AM:

   - Restart AM or the web container where it runs.

For more details on CORS, see the *Cross-Origin Resource Sharing* specification.

Preparing AES Key Wrap Encryption

By default, AM uses the Java Cryptography Extension (JCE) encryption class to encrypt and decrypt system passwords and keys used in the configuration, and by other components, such as agents.

If your deployment requires a more secure encryption algorithm, AM supports the Advanced Encryption Standard (AES) Key Wrap algorithm (RFC3394). AM's implementation of AES Key Wrap uses the Password-Based Key Derivation Function 2 (PBKDF2) (RFC2898) with HMAC-SHA1. This allows administrators to choose key size hash algorithms, such as SHA256, SHA384, or SHA512.

Important

The AES Key Wrap Encryption algorithm is only enabled when installing AM. There is no current upgrade path for existing installations.

The Security Token Service (STS) feature does not support the AES Key Wrap Encryption algorithm. Make sure that you do not deploy this feature in an AM instance configured to use the AES Key Wrap Encryption algorithm.

You must also update the **ssoadm** command to work with AES key wrap encryption. See "To Configure ssoadm for AES Key Wrap Encryption".

Warning

When implementing AES Key Wrap Encryption, take special care when selecting your encryption key iteration count.

If you select a large iteration count of 20,000, for example, you can effectively slow down a brute-force attack when passwords are of low quality (less than 20 characters and non-randomized). The trade off is that an iteration count of 20,000 can also negatively impact AM startup times if there are many agents in your deployment.

Determine the optimal iteration count for your deployment based on your security and performance requirements. Set the iteration count to a large number only if absolutely necessary.

AM does not have an iteration count requirement. However, it will log a warning if both of the following conditions are true:

- The number of iterations is less than 10,000.
- The AM encryption key is less than 20 characters long.

To Configure AES Key Wrap Encryption for Tomcat

- Edit your container startup scripts, for example `setenv.sh`, to set the following JVM system properties in Tomcat:

  ```
  JAVA_OPTS="$JAVA_OPTS -
  Dcom.iplanet.security.encryptor=org.forgerock.openam.shared.security.crypto.AESWrapEncryption"
  JAVA_OPTS="$JAVA_OPTS -Dorg.forgerock.openam.encryption.key.iterations=10000"
  JAVA_OPTS="$JAVA_OPTS -Dorg.forgerock.openam.encryption.key.size=256"
  JAVA_OPTS="$JAVA_OPTS -Dorg.forgerock.openam.encryption.key.digest=SHA512"
  JAVA_OPTS="$JAVA_OPTS -Dorg.forgerock.openam.encryption.padshortinputs"
  ```

  Only the first line in the example is required. The other lines are configurable to meet the needs of your deployment. Key sizes greater than 128 bits require that the JCE Unlimited Strength policy files be installed in your system. PBKDF2 using SHA256, SHA384, and SHA512 is only available on Java 8.

  For systems running Java 17, the `org.forgerock.openam.encryption.padshortinputs` property pads short inputs (less than 8 bytes). If you are using Java 17 with AES Key Wrap Encryption, enable this system property and re-encrypt any short system passwords that have already been encrypted. If you do not do this, AM will be unable to decrypt the short values.

Note

You cannot change these configuration parameters once AM has been installed.

To Configure ssoadm for AES Key Wrap Encryption

After you enable AES key wrap encryption, update the **ssoadm** command to work with the new encryption settings.

- Add the following properties to the `/path/to/ssoadm/setup` and `/path/to/ssoadm/bin/ssoadm` commands:

```
-Dcom.iplanet.security.encryptor=org.forgerock.openam.shared.security.crypto.AESWrapEncryption
-Dorg.forgerock.openam.encryption.key.iterations=10000
-Dorg.forgerock.openam.encryption.key.size=256
-Dorg.forgerock.openam.encryption.key.digest=SHA512
```

Downloading and Deploying

This section covers acquiring the AM software and deploying it into a web application container.

The topics covered in this section are:

- Obtaining Software
- Deploying

Obtaining Software

The *ForgeRock BackStage* website hosts downloads, including a `.zip` file with all of the AM components, the `.war` file, AM tools, the configurator, web and Java agents, Identity Gateway, and documentation. Verify that you review the Software License and Subscription Agreement presented before you download AM files.

For each release of the AM, you can download the entire package as a `.zip` file, only the AM `.war` file, or only the administrative tools as a `.zip` archive. The Archives only have the AM source code used to build the release.

After you download the `.zip` file, create a new directory for AM, and unzip the `.zip` file to access the content.

```
$ cd ~/Downloads
$ mkdir openam ; cd openam
$ unzip ~/Downloads/AM-6.5.5.zip
```

When you unzip the archive of the entire package, you get ldif, license, and legal directories in addition to the following files:

Distribution Files

File	Description
`AM-6.5.5.war`	The distribution `.war` file includes the core server code with an embedded DS server, which stores configuration data and simplifies deployments. The distribution includes an administrative graphical user interface (GUI) Web console. During installation, the `.war` file accesses properties to obtain the fully qualified domain name, port, context path, and the location of the configuration folder. These properties can be obtained from the `boot.json` file in the AM installation directory, from environment variables, or from a combination of the two. This file is also available to download individually.
`AM-Soap-STS-Server-6.5.5.war`	AM provides a SOAP-based security token service (STS) server that issues tokens based on the WS-Security protocol[a].
`SSOAdminTools-5.1.2.24.zip`	AM provides an `ssoadm` command-line tool that allows administrators to configure and maintain AM as well as create their own configuration scripts. The `zip` distribution file contains binaries, properties file, script templates, and setup scripts for UNIX and windows servers.
`SSOConfiguratorTools-5.1.2.24.zip`	AM provides configuration and upgrade tools for installing and maintaining your server. The `zip` distribution file contains libraries, legal notices, and supported binaries for these configuration tools. Also, you can view example configuration and upgrade properties files that can be used as a template for your deployments.
`Fedlet-6.5.5.zip`	AM provides an AM Fedlet, a light-weight SAML v2.0 service provider. The Fedlet lets you set up a federated deployment without the need of a fully-featured service provider.
`IDPDiscovery-6.5.5.war`	AM provides an IdP Discovery Profile (SAMLv2 binding profile) for its IdP Discovery service. The profile keeps track of the identity providers for each user.
`sample-trees-6.5.5.zip`	Clean installs of AM with an embedded data store provide ready-made sample authentication trees to demonstrate how they can be put together. These sample trees are not installed by default if you are upgrading an existing instance of AM. The `sample-trees-6.5.5.zip` file contains the sample trees in JSON files, ready for import by *Amster* command-line interface. For information on importing files by using Amster, see Importing Configuration Data in the *Amster 6.5 User Guide*.

[a]AM also provides REST-based STS service endpoints, which you can directly utilize on the AM server.

Deploying

After you have downloaded AM software, deploy it to your installed application container.

Note that deploying AM only extracts the files into the application container, prior to installation and configuration. Deploying AM also makes LDIF files available, which can be used to prepare external data stores for use with AM.

To Deploy an Instance

The `AM-6.5.5.war` file contains the AM server. How you deploy the `.war` file depends on your web application container.

1. Deploy the `.war` file on your container.

   For example, copy the file to deploy on Apache Tomcat.

   ```
   $ cp AM-6.5.5.war /path/to/tomcat/webapps/openam.war
   ```

   During trials or development, you can change the file name to `openam.war` when deploying in Tomcat, so that the deployment URI is `/openam`.

   When installing AM in a production environment, do not use a predictable deployment URI such as `/openam` or `/opensso`.

   Note

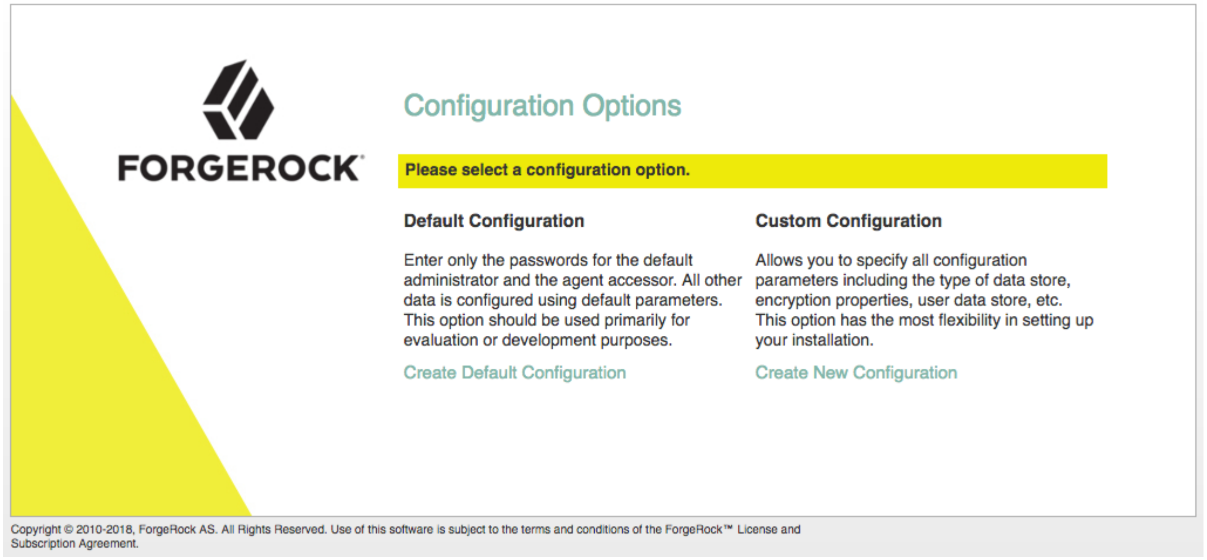
   You change the file name to something other than `openam.war` when deploying in Tomcat so that the deployment URI is not `/openam`. For helpful hints on avoiding obvious deployment defaults, see "Avoiding Obvious Defaults".

   Important

   To properly configure AM, AM requires a deployment URI with a non-empty string after `/`. Do not deploy AM at the root context. Do not rename the `.war` file to `ROOT.war` before deploying on Tomcat, for example.

   It can take several seconds for AM to be deployed in your container.

2. Navigate to the initial configuration screen, for example `https://openam.example.com:8443/openam`.



Configuration Options

Please select a configuration option.

Default Configuration

Enter only the passwords for the default administrator and the agent accessor. All other data is configured using default parameters. This option should be used primarily for evaluation or development purposes.

Create Default Configuration

Custom Configuration

Allows you to specify all configuration parameters including the type of data store, encryption properties, user data store, etc. This option has the most flexibility in setting up your installation.

Create New Configuration

Copyright © 2010-2018, ForgeRock AS. All Rights Reserved. Use of this software is subject to the terms and conditions of the ForgeRock™ License and Subscription Agreement.

AM is now ready for installation. To proceed, take on of the following actions:

- Configure external data stores using the files created during deployment. See "*Preparing External Stores*".
- Use the embedded data stores for evaluation purposes, and skip ahead to installing AM. See "Installing a Single Server".

Chapter 2
Preparing External Stores

This chapter covers setting up external LDAP directory servers for storing configuration data and identity profiles.

AM ships with an embedded DS instance that you can install as part of the AM configuration process. By default, AM installs the embedded directory server instance and its configuration settings in the `$HOME` folder of the user running AM and runs the embedded directory server instance in the same JVM memory space as AM.

AM stores both identity profiles, that is information about your users, devices, and things, as well as data managed by the Core Token Service (CTS) pertaining to authenticated identities—AM CTS-based sessions, logout blacklists, and several types of authentication tokens—in the same embedded DS server that holds the AM configuration.

AM connects to the embedded DS server as directory superuser, bypassing access control evaluation because AM manages the directory as its private store.

The embedded DS server is best suited to evaluation, development and testing purposes. In production environments, external stores are recommended.

You can use an external directory server instead of the embedded DS instance for storing the following data:

- Configuration data. The properties and settings used by the AM instance. See "Preparing Configuration Stores".
- Identity profiles. Information about the users, devices, or things that will be authenticating to your systems. You can also configure AM to access existing directory servers to obtain identity profiles. See "Preparing Identity Repositories".
- Policy data. Policy-related data, such as policies, policy sets, and resource types. See "Preparing Policy and Application Stores".
- Application data. Application-related data, such as web and Java agent configuration, federation entities and configuration, and OAuth 2.0 client definitions. See "Preparing Policy and Application Stores".
- CTS data. Information about sessions, OAuth 2.0 tokens, and session blacklists and whitelists. See "*Implementing the Core Token Service*".
- UMA-related data. Information about resources, labels, audit messages, and pending requests. See "Preparing External UMA Data Stores" in the *User-Managed Access (UMA) 2.0 Guide*.

The following table shows which directory servers are supported for storing different data:

Supported Directory Servers

Directory Server	Versions	Config	Apps/ Policies	CTS	Identities	UMA
Embedded ForgeRock Directory Services [a]	6.5.6	✓	✓	✓	✓	✓
External ForgeRock Directory Services	Any ForgeRock-supported version	✓	✓	✓	✓	✓
Oracle Unified Directory	11g R2				✓	
Oracle Directory Server Enterprise Edition	11g				✓	
Microsoft Active Directory	2012 R2, 2016				✓	
IBM Tivoli Directory Server	6.3				✓	

[a]Demo and test environments only

The procedure for preparing external directory servers for use by AM is similar for each of the different data types. The steps to perform are as follows:

1. If it does not yet exist, install the external directory server software, for example Directory Services.
2. As the directory administrator user:
   a. Apply the relevant schema to the directory.
   b. Create indexes to optimize data retrieval from the directory server.
   c. Create a user account with the minimum required privileges for AM to bind to the directory server with and access neccessary data.

To prepare external stores for installation of AM, see the following sections:

- "Preparing Configuration Stores"
- "Preparing Identity Repositories"
- "Preparing Policy and Application Stores"

Preparing Configuration Stores

This section explains how to prepare a single DS server as an external configuration data store.

To Install and Configure Directory Services for Configuration Data

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

Using a setup profile will create the backend, schema, bind user, and indexes required for use with configuration data.

1. To install DS using a setup profile, follow the steps in "To Use DS for AM Configuration Data" in the *Directory Services 6.5 Installation Guide*.
2. Proceed to installation of AM to use the prepared DS directory server as an external configuration store. See "To Custom Configure an Instance".

   The default bind account to use when installing AM to use the external configuration store is:

   ```
   uid=am-config,ou=admins,ou=am-config
   ```

Preparing Identity Repositories

AM accesses user identity data from one or more identity repositories.

In most deployments, AM connects to existing LDAP directory servers for user identity data, as it shares data in an identity repository with other applications.

AM ships with an embedded Directory Services server that you can install as part of the AM configuration process. If you are configuring AM to share data with other applications, or if you expect your deployment will have a large amount of users, connect AM to an external identity repository.

Note

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

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

To prepare external identity repositories, see the following sections:

- If you are installing a new Directory Services instance for identity data, see "Installing and Configuring Directory Services for Identity Data".
- If you are connecting AM to an existing identity repository, see "Configuring Existing Directory Servers for Identity Data".

Installing and Configuring Directory Services for Identity Data

This section shows how to install and set up a new DS server instance for identity data.

To Install and Configure Directory Services for Identity Data

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

Using a setup profile will create the backend, schema, bind user, and indexes required for use with identity data.

1. To install DS using a setup profile, follow the steps in *To Use DS for AM Identity Data* in the *Directory Services 6.5 Installation Guide*.

2. The following steps update the permissions in an external Directory Services identity repository. For other directory servers, apply the relevant LDIF files using the appropriate tools.

   a. Edit the `opendj_userinit.ldif` LDIF file in the `/path/to/openam/WEB-INF/template/ldif/opendj` directory, replacing all variables that are surrounded by *at* (@) symbols with a value specific to your directory server.

      For example, in the `opendj_userinit.ldif` LDIF file you must replace all instances of *@userStoreRootSuffix@* with the root suffix you specified when configuring the external DS identity store, the default being `ou=identities`.

      Tip

      For information on downloading and deploying the AM .war file, see "Downloading and Deploying".

   b. Use the **ldapmodify** command to add the updated LDIF data to the external instance. For example:

```
$ ./ldapmodify \
--hostname 'id.example.com' \
--port 1389 \
--useStartTLS \
--trustAll \
--continueOnError \
--bindDN uid=admin \
--bindPassword 'str0ngEx4mplePa55word' \
/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opendj/opendj_userinit.ldif
```

      For more information on this LDIF file, and equivalent files for supported directory servers, see "*Supported LDIF Files*".

3. (Optional) If you intend to use push or web authentication, you may need to update the directory server 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`

For example:

```
$ ./ldapmodify \
--hostname 'id.example.com' \
--port 1389 \
--useStartTLS \
--trustAll \
--continueOnError \
--bindDN 'cn=Directory Manager' \
--bindPassword 'str0ngEx4mplePa55word' \
/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opendj/push_2fa.ldif \
/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opendj/opendj_webauthndevices.ldif
```

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

4. Proceed to configuring the identity store in AM. See "*Setting Up Identity and Data Stores*" in the *Setup and Maintenance Guide*.

   The bind account to use when configuring the identity store in AM is `uid=am-identity-bind-account,ou=admins,ou=identities`.

Configuring Existing Directory Servers for Identity Data

It is common for AM to access identity data from an existing directory server. AM requires a user account to connect to the directory server, and AM LDAP schema to update entries with AM-related identity data.

For a list of supported external directory servers, see "Supported Directory Servers".

The following sections show you how to prepare an existing identity repository for use in AM:

1. "Creating a Directory Server User for AM Connections"
2. "Updating the Schema in an External Identity Repository"

Creating a Directory Server User for AM Connections

AM connects to an external directory server as a user that you specify in the AM identity repository configuration. This user is known as the *AM bind account*.

Specifying the directory administrator as the AM bind account is not recommended for production deployments as it would give AM directory administrator privileges to the identity repository.

Instead, create a separate AM bind account with fewer access privileges than the directory administrator so that you can assign the appropriate level of privileges for the AM bind account.

You need to consider two areas of permission for the AM bind account:

Schema Update Privileges

AM needs to update the directory schema when you configure a new identity repository and when you upgrade AM software. If the AM bind account has schema update privileges, AM can update the schema dynamically during identity repository configuration and during AM upgrades. If the AM bind account does not have schema update privileges, you must update the schema manually before configuring a new identity repository and before upgrading AM.

Directory Read and Write Access Privileges

If you want AM to create, update, and delete user entries, then the AM bind account must have full read and write access to the identity data in the directory. If you are using an external identity repository as a read-only user directory, then the AM bind account needs read privileges only.

The level of access privileges you give the AM bind account is specific to each AM deployment. Work with your directory server administrator to determine the appropriate level of privileges as part of the process of preparing your external identity repositories.

The following procedure gives an example of creating a bind account in Active Directory:

To Create a Bind Account in Active Directory for AM Connections

Perform these steps to create a user that AM can use to connect to Active Directory. These steps are one example, consult with your Active Directory administrator on how best to create an account.

1. Create a new user account in the Active Directory domain. For example, in Windows 2012:
    a. In the Active Directory Users and Computers tool, on the Action menu, select New, and then click User.
    b. Provide descriptive values for the new user, and a logon name such as `AM-Bind-Account`.
    c. Click Next.
    d. Enter a strong password for the bind account, disable the *User must change password at next logon* option, and then click Next.
    e. Review the details of the new account, and then click Finish.
2. Give the new bind account access to the directory data:
    a. In the Active Directory Users and Computers tool, right-click the domain that contains your users, and then select Delegate Control.
    b. Add the bind account you created in the previous step, and then click Next.
    c. Select the tasks you want to allow the AM bind account to perform from the list.

       For example, to allow read and write access to users, enable the *Create, delete, and manage inetOrgPerson accounts* task.

d. After assigning the tasks you want to allow the AM bind account to perform, click Finish.

You can now proceed to setting up the neccessary schema in the directory server.

The bind account to use when configuring the identity store in AM is the full DN of the user, for example `uid=AM-Bind-Account,ou=Users,dc=example,dc=org`.

Updating the Schema in an External Identity Repository

AM can add the necessary LDAP schema definitions itself, if it has sufficient privileges to do so, or you can apply the LDAP schema definition LDIF files manually if required. See the following procedures:

- "To Prepare an External Identity Repository With Manual Schema Updates"
- "To Prepare an External Identity Repository With Automatic Schema Updates"

To Prepare an External Identity Repository With Manual Schema Updates

If the AM bind account does not have permission to update schema then you must configure existing external data stores by using manual schema updates. To do this, you must update the directory server schema of the external identity repository manually at the following times:

- Before you configure the identity repository as part of initial AM configuration.
- Before you configure an identity repository after initial AM configuration.
- Whenever you upgrade AM.

A number of LDIF files are provided in the AM `.war` file for supported identity directory servers. The path is `/path/to/openam/WEB-INF/template/ldif/{directory_type}`, where `{directory_type}` is one of the following:

- `ad` for Microsoft Active Directory
- `adam` for Microsoft Active Directory Lightweight Directory Services
- `odsee` for Oracle Directory Server Enterprise Edition
- `opendj` for ForgeRock Directory Services and Oracle Unified Directory
- `tivoli` for IBM Tivoli Directory Server

For more information on the LDIF files, see "*Supported LDIF Files*".

The following steps update the schema in an Active Directory identity repository. For other directory servers, apply the relevant LDIF files using the appropriate tools.

1. Edit the LDIF files in the `/path/to/openam/WEB-INF/template/ldif/ad` directory, replacing any variables that are surrounded by *at* (@) symbols with a value specific to your directory server.

For the Active Directory LDIF files you must replace all instances of *@userStoreRootSuffix@* with the root suffix used by your Active Directory identity store, for example `DC=example,DC=org`.

> **Tip**
>
> For information on downloading and deploying the AM .war file, see "Downloading and Deploying".

2. Using an Active Directory administrator account, add the required AM schema extensions to your external identity repository.

   For example, in PowerShell, run the **ldifde** command to import the user, device print, and dashboard schema extensions:

```
PS C:\Users\Administrator> cd \path\to\openam\WEB-INF\template\ldif\ad

PS C:\path\to\openam\WEB-INF\template\ldif\ad> ldifde -i -f .\ad_user_schema.ldif
Connecting to "domain.example.org"
Logging in as current user using SSPI
Importing directory from file ".\ad_user_schema.ldif"
Loading entries.................................................................
64 entries modified successfully.

The command has completed successfully
PS C:\path\to\openam\WEB-INF\template\ldif\ad> ldifde -i -f .\ad_deviceprint.ldif
Connecting to "domain.example.org"
Logging in as current user using SSPI
Importing directory from file ".\ad_deviceprint.ldif"
Loading entries.................................................................
6 entries modified successfully.

The command has completed successfully
PS C:\path\to\openam\WEB-INF\template\ldif\ad> ldifde -i -f .\ad_dashboard.ldif
Connecting to "domain.example.org"
Logging in as current user using SSPI
Importing directory from file ".\ad_dashboard.ldif"
Loading entries.................................................................
6 entries modified successfully.

The command has completed successfully
```

3. (Optional) If you intend to use push or web authentication, apply the following LDIF files:

   - `/path/to/openam/WEB-INF/template/ldif/ad/ad_pushdevices.ldif`
   - `/path/to/openam/WEB-INF/template/ldif/ad/ad_webauthndevices.ldif`

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

4. Proceed to configuring the identity store in AM. See "*Setting Up Identity and Data Stores*" in the *Setup and Maintenance Guide*.

To Prepare an External Identity Repository With Automatic Schema Updates

If the bind account has permission to update schema then you can allow AM to update the schema automatically.

To allow AM to update the schema, you must first configure AM to be able to access the directory server, and enable the Load Schema option, by performing the following steps:

1. Configure the directory server in AM by following the instructions in "To Configure an Identity Data Store" in the *Setup and Maintenance Guide*.

   > **Important**
   >
   > Enable the Load Schema option before clicking Save Changes, and AM will apply the neccessary schema to the directory server.

   The schema is loaded as part of configuring the identity store in AM. No further configuration is required.

2. Verify that the new identity repository is correctly configured in AM. See "Testing External Identity Repository Access" in the *Setup and Maintenance Guide*.

Preparing Policy and Application Stores

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

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

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

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

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

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

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

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

1. To install DS using a setup profile, follow the steps in *To Use DS for AM Configuration Data* in the *Directory Services 6.5 Installation Guide*.

   Important

   Ensure that you specify the same base DN value as your configuration store (whether it is embedded or external) when using a setup profile to create a policy or application store. For example:

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

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

2. You have successfully installed and prepared the DS directory server for use as an external policy or application store.

   You can use the instance created in these procedures for both policy and application data simultaneously. To have separate instances for policies and applications, repeat the previous step to create an additional external policy or application store.

   Proceed to configuring AM to use the prepared DS directory servers as external policy or application stores, see "Setting Up External Data Stores" in the *Setup and Maintenance Guide*.

   The default bind account to use when configuring AM to use the external store takes the format:

   ```
   uid=am-config,ou=admins,Base DN
   ```

Migrating Policy and/or Application Data to a DS Instance

If you are upgrading existing AM instances to use external DS instances for policy and/or application data, you may want to migrate that existing data into the new instances. Migrating policy and/or application data to a separate store allows tuning and scaling of the more dynamic data, separately from the more static nature of configuration data.

This section covers the following high-level methods for migrating policy and/or application data to new DS instances:

- "Exporting and Importing Data in DS by using LDIF"
- "Backing Up and Restoring a DS Instance"

Exporting and Importing Data in DS by using LDIF

This section gives an overview of migrating data to new DS instances by using the `import-ldif` and `export-ldif` commands.

The top-level steps are as follows:

1. Install a DS instance to store the policy and/or application data, by following the steps in "Installing and Configuring Directory Services for Policy and/or Application Data".

2. Use the **export-ldif** command to create an LDIF file of the structure of the new DS instance.

   For example:

```
$ ./export-ldif \
--hostname external.example.com \
--port 5444 \
--bindDN "cn=Directory Manager" \
--bindPassword forgerock \
--backendId cfgStore \
--ldifFile /tmp/Apps_and_Policies.ldif
```

3. Use the **export-ldif** command to append to the previous LDIF file your existing policy and/or application data.

   Tip

   The **export-ldif** command has a `--includeBranch` option to limit the data exported to just policy and/or application data.

   See the following commands to export application data, policy data, or both:

   Policy and Application Data

```
$ /path/to/ds/bin/export-ldif \
--hostname 'config.example.com'\
--port 4444 \
--useSsl \
--usePkcs12TrustStore /path/to/opendj/config/keystore \
--trustStorePasswordFile /path/to/opendj/config/keystore.pin \
--bindDN uid=admin \
--bindPassword ${ds.admin.password} \
--includeBranch o=sunamhiddenrealmdelegationservicepermissions,ou=services,dc=example,dc=com \
--includeBranch ou=AgentService,ou=services,dc=example,dc=com \
--includeBranch ou=sunEntitlementService,ou=services,dc=example,dc=com \
--includeBranch ou=sunEntitlementIndexes,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMCOTConfigService,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMIDFFMetadataService,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMSAML2MetadataService,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMWSFederationMetadataService,ou=services,dc=example,dc=com \
--backendId appData \
--appendToLdif \
--ldifFile /tmp/Apps_and_or_Policies.ldif
```

   Policy Data

```
$ /path/to/ds/bin/export-ldif \
--hostname 'config.example.com'\
--port 4444 \
--useSsl \
--usePkcs12TrustStore /path/to/opendj/config/keystore \
--trustStorePasswordFile /path/to/opendj/config/keystore.pin \
--bindDN uid=admin \
--bindPassword ${ds.admin.password} \
--includeBranch ou=sunEntitlementService,ou=services,dc=example,dc=com \
--includeBranch ou=sunEntitlementIndexes,ou=services,dc=example,dc=com \
--includeBranch o=sunamhiddenrealmdelegationservicepermissions,ou=services,dc=example,dc=com \
--backendId appData \
--appendToLdif \
--ldifFile /tmp/Apps_and_or_Policies.ldif
```

Application Data

```
$ /path/to/ds/bin/export-ldif \
--hostname 'config.example.com'\
--port 4444 \
--useSsl \
--usePkcs12TrustStore /path/to/opendj/config/keystore \
--trustStorePasswordFile /path/to/opendj/config/keystore.pin \
--bindDN uid=admin \
--bindPassword ${ds.admin.password} \
--includeBranch ou=AgentService,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMCOTConfigService,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMIDFFMetadataService,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMSAML2MetadataService,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMWSFederationMetadataService,ou=services,dc=example,dc=com \
--backendId appData \
--appendToLdif \
--ldifFile /tmp/Apps_and_or_Policies.ldif
```

Replace *dc=example,dc=com* in the commands above with your base DN value.

4. To also export policy and/or application data from subrealms, repeat the previous step, but altered to include the path to the subrealm in the included branches.

   For example, to export data from a realm named `mySubRealm`, add `ou=services,o=mySubRealm,` into each of the included branches, as shown below:

```
$ ./export-ldif \
--hostname openam.example.com \
--port 4444 \
--bindDN uid=admin \
--bindPassword forgerock \
--includeBranch
 o=sunamhiddenrealmdelegationservicepermissions,ou=services,o=mySubRealm,ou=services,dc=example,dc=com
 \
--includeBranch ou=AgentService,ou=services,o=mySubRealm,ou=services,dc=example,dc=com \
--includeBranch ou=sunEntitlementService,ou=services,o=mySubRealm,ou=services,dc=example,dc=com \
--includeBranch ou=sunEntitlementIndexes,ou=services,o=mySubRealm,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMCOTConfigService,ou=services,o=mySubRealm,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMIDFFMetadataService,ou=services,o=mySubRealm,ou=services,dc=example,dc=com \
--includeBranch ou=sunFMSAML2MetadataService,ou=services,o=mySubRealm,ou=services,dc=example,dc=com \
--includeBranch
 ou=sunFMWSFederationMetadataService,ou=services,o=mySubRealm,ou=services,dc=example,dc=com \
--backendId appData \
--appendToLdif \
--ldifFile /tmp/Apps_and_Policies.ldif
```

Repeat this step for each subrealm that contains application and/or policy data you want to transfer to an external store.

5. Edit the LDIF file to include the required top-level `ou=services` entry. For example, after the entry for `dn: dc=example,dc=com` add the following:

```
dn: ou=services,dc=example,dc=com
objectclass: top
objectclass: organizationalunit
objectclass: sunServiceComponent
ou: services
```

Note that the example above requires a blank line both before and after, and you must replace *dc=example,dc=com* with your base DN value.

The start of the resulting LDIF will resemble the following:

```
dn: dc=example,dc=com
objectClass: top
objectClass: untypedObject
dc: openam
aci: (targetattr="*")(version 3.0;acl "Allow CRUDQ operations";allow (search, read, write, add,
 delete)(userdn = "ldap:///uid=am-config,ou=admins,dc=example,dc=com");)
aci: (targetcontrol="2.16.840.1.113730.3.4.3")(version 3.0;acl "Allow persistent search"; allow
 (search, read)(userdn = "ldap:///uid=am-config,ou=admins,dc=example,dc=com");)
aci: (targetcontrol="1.2.840.113556.1.4.473")(version 3.0;acl "Allow server-side sorting"; allow
 (read)(userdn = "ldap:///uid=am-config,ou=admins,dc=example,dc=com");)
entryUUID: 5beee9ea-1c31-3129-a8f9-c79da3102f26

dn: ou=services,dc=example,dc=com
objectclass: top
objectclass: organizationalunit
objectclass: sunServiceComponent
ou: services

dn: ou=admins,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
ou: admins
...
...
```

6. Repeat the previous step to create the parent entries for any subrealms that you exported.

   For example, before the `ou=admins,dc=example,dc=com` line, add the following for a subrealm named `mySubRealm`:

   ```
   dn: o=mySubRealm,ou=services,dc=example,dc=com
   objectClass: sunRealmService
   objectClass: top
   o: mySubRealm

   dn: ou=services,o=mySubRealm,ou=services,dc=example,dc=com
   objectClass: organizationalUnit
   objectClass: sunServiceComponent
   objectClass: top
   ou: services
   ```

   Note that the example above requires a blank line both before and after, and you must replace *dc=example,dc=com* with your base DN value.

   The start of the resulting LDIF will resemble the following:

   ```
   dn: dc=example,dc=com
   objectClass: top
   objectClass: untypedObject
   dc: openam
   aci: (targetattr="*")(version 3.0;acl "Allow CRUDQ operations";allow (search, read, write, add,
    delete)(userdn = "ldap:///uid=am-config,ou=admins,dc=example,dc=com");)
   aci: (targetcontrol="2.16.840.1.113730.3.4.3")(version 3.0;acl "Allow persistent search"; allow
    (search, read)(userdn = "ldap:///uid=am-config,ou=admins,dc=example,dc=com");)
   aci: (targetcontrol="1.2.840.113556.1.4.473")(version 3.0;acl "Allow server-side sorting"; allow
    (read)(userdn = "ldap:///uid=am-config,ou=admins,dc=example,dc=com");)
   entryUUID: 5beee9ea-1c31-3129-a8f9-c79da3102f26
   ```

```
dn: ou=services,dc=example,dc=com
objectclass: top
objectclass: organizationalunit
objectclass: sunServiceComponent
ou: services

dn: o=mySubRealm,ou=services,dc=example,dc=com
objectClass: sunRealmService
objectClass: top
o: mySubRealm

dn: ou=services,o=mySubRealm,ou=services,dc=example,dc=com
objectClass: organizationalUnit
objectClass: sunServiceComponent
objectClass: top
ou: services

dn: ou=admins,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
ou: admins
...
...
```

7. Use the **import-ldif** command to add the updated LDIF data to the new instance. For example:

```
$ ./import-ldif \
--hostname external.example.com \
--port 5444 \
--bindDN "cn=Directory Manager" \
--bindPassword forgerock \
--backendId cfgStore \
--ldifFile /tmp/Apps_and_Policies.ldif
```

8. Configure AM to use the new store for policy and/or application data, by following the steps in "Setting Up External Policy and Application Stores." in the *Setup and Maintenance Guide*.

For more information and the specific steps for importing and exporting data by using LDIF, see *Importing and Exporting Data* in the *DS Maintenance Guide*.

Backing Up and Restoring a DS Instance

This section gives an overview of migrating data to new DS instances by using the DS **backup** and **restore** commands.

The top-level steps are as follows:

1. Use the **backup** command to make a backup copy of the existing configuration store.
2. Install a DS instance to store the policy and/or application data, by following the steps in "Installing and Configuring Directory Services for Policy and/or Application Data".

   Note that the DS instance MUST have the same base DN, and backend name, as the instance from which the backup was created.

3. Use the **restore** command to restore your policy and/or application data to the new store.

4. Configure AM to use the new store for policy and/or application data, by following the steps in "Setting Up External Policy and Application Stores." in the *Setup and Maintenance Guide*.

For more information and the specific steps for performing backup and restore operations in DS, see *Backing Up and Restoring Data* in the *DS Maintenance Guide*, and *To Initialize a Replica From Backup Files* in the *DS Server Configuration Guide*.

Chapter 3
Installing and Starting Servers

This chapter covers installation and startup.

The chapter includes procedures for installing AM on a single server, installing AM on multiple servers, and installing AM's administrative tools. It also covers how to start AM, including overriding default startup options.

The following table contains a list of activities you might perform while installing and starting AM:

Installation Options

Installation Action	Documentation Reference
Install quickly for evaluation using default settings	"To Configure With Defaults" Alternatively, follow the full example in Quick Start Guide.
Install AM server, choosing settings	"To Custom Configure an Instance"
Start an AM server	"Starting Servers"
Erase the configuration and start over	"To Delete a Configuration Before Redeploying"
Add an AM server to a site	"To Add a Server to a Site"
Implement the Core Token Service	"*Implementing the Core Token Service*"
Troubleshoot an AM installation	"To Troubleshoot an Installation"
Install **ssoadm** for CLI configuration	"Installing and Using the Tools"
Perform a command-line install	"Installing Silently"
Removing AM	"*Removing Installations*"

This chapter does not cover installation for enforcing policies on resource servers. To manage access to resources on other servers, you can use IG or AM web and Java agents.

ForgeRock Identity Gateway is a high-performance reverse proxy server with specialized session management and credential replay functionality. It can function as a standards-based policy enforcement point.

AM web and Java agents provide policy enforcement on supported web servers and Java containers, and are tightly integrated with AM. See the Web Policy Agents documentation, or the Java Policy Agents documentation. for instructions on installing AM web or Java agents in supported web servers and Java application containers.

Installing a Single Server

This section covers tasks required to install a single AM server.

To Configure With Defaults

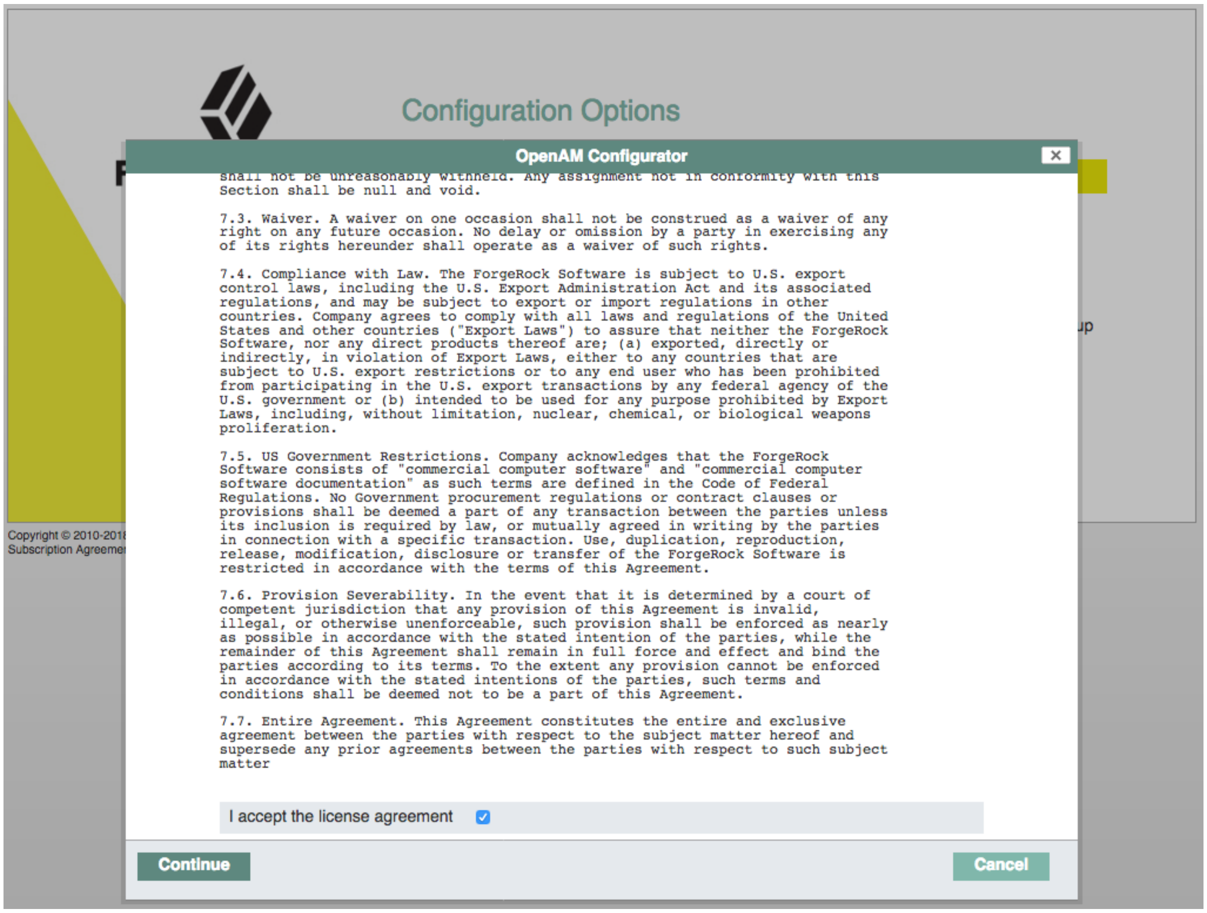
The default configuration option configures the embedded DS server using default ports. If the ports are already in use, AM uses free ports as both configuration store and identity store.

The default configuration sets the cookie domain based on the full URL that was used to access the configurator.

Configuration settings are saved to the home directory of the user running the web application container in a directory named after the deployment URI. In other words if AM is deployed under `/openam`, then the configuration is saved under `$HOME/openam/`.

1. In the initial configuration screen, click Create Default Configuration under Default Configuration.

2. Review the software license agreement. If you agree to the license, click "I accept the license agreement", and then click Continue.



shall not be unreasonably withheld. Any assignment not in conformity with this Section shall be null and void.

7.3. Waiver. A waiver on one occasion shall not be construed as a waiver of any right on any future occasion. No delay or omission by a party in exercising any of its rights hereunder shall operate as a waiver of such rights.

7.4. Compliance with Law. The ForgeRock Software is subject to U.S. export control laws, including the U.S. Export Administration Act and its associated regulations, and may be subject to export or import regulations in other countries. Company agrees to comply with all laws and regulations of the United States and other countries ("Export Laws") to assure that neither the ForgeRock Software, nor any direct products thereof are; (a) exported, directly or indirectly, in violation of Export Laws, either to any countries that are subject to U.S. export restrictions or to any end user who has been prohibited from participating in the U.S. export transactions by any federal agency of the U.S. government or (b) intended to be used for any purpose prohibited by Export Laws, including, without limitation, nuclear, chemical, or biological weapons proliferation.

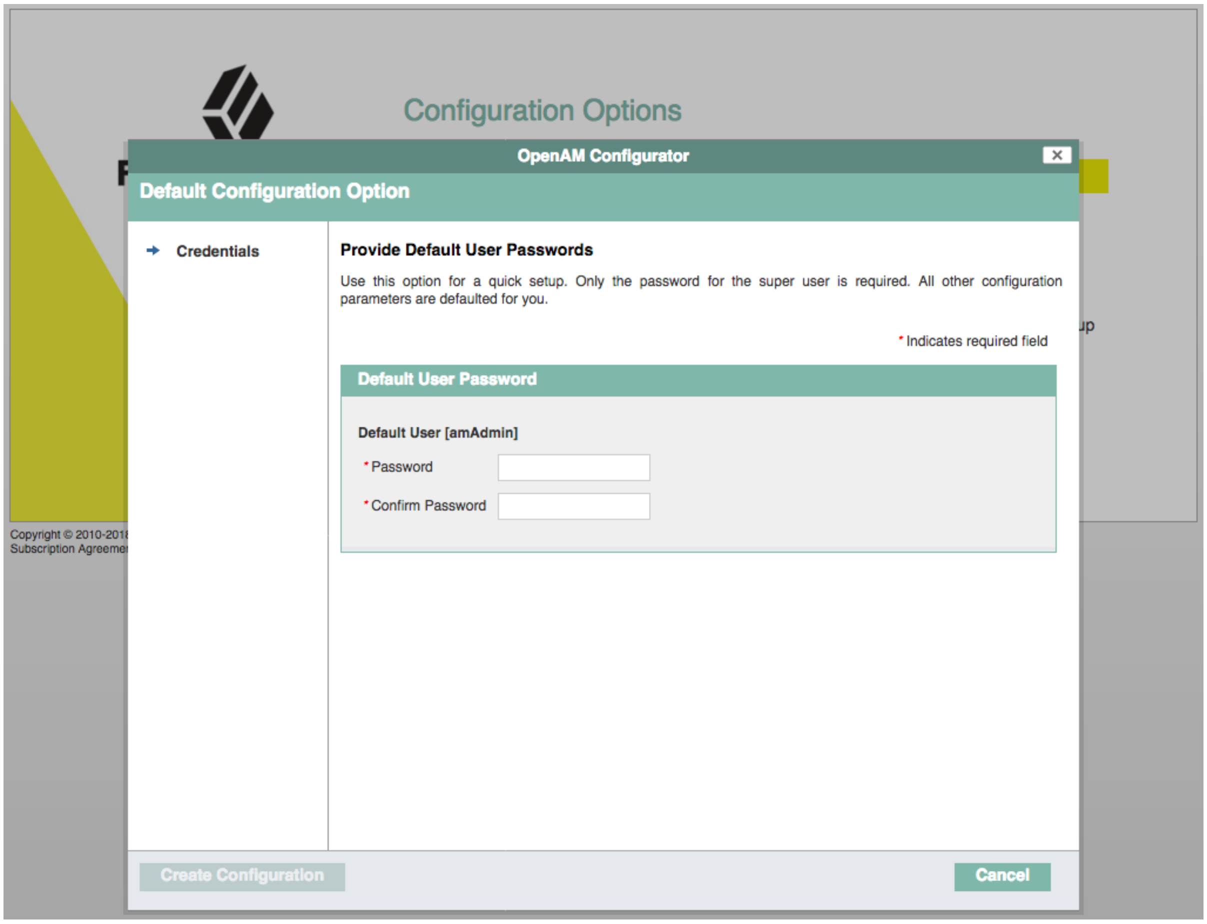
7.5. US Government Restrictions. Company acknowledges that the ForgeRock Software consists of "commercial computer software" and "commercial computer software documentation" as such terms are defined in the Code of Federal Regulations. No Government procurement regulations or contract clauses or provisions shall be deemed a part of any transaction between the parties unless its inclusion is required by law, or mutually agreed in writing by the parties in connection with a specific transaction. Use, duplication, reproduction, release, modification, disclosure or transfer of the ForgeRock Software is restricted in accordance with the terms of this Agreement.

7.6. Provision Severability. In the event that it is determined by a court of competent jurisdiction that any provision of this Agreement is invalid, illegal, or otherwise unenforceable, such provision shall be enforced as nearly as possible in accordance with the stated intention of the parties, while the remainder of this Agreement shall remain in full force and effect and bind the parties according to its terms. To the extent any provision cannot be enforced in accordance with the stated intentions of the parties, such terms and conditions shall be deemed not to be a part of this Agreement.

7.7. Entire Agreement. This Agreement constitutes the entire and exclusive agreement between the parties with respect to the subject matter hereof and supersede any prior agreements between the parties with respect to such subject matter

I accept the license agreement ✓

3. Provide the password for the default AM administrator, `amadmin`.


Configuration Options

OpenAM Configurator

Default Configuration Option

Credentials

Provide Default User Passwords

Use this option for a quick setup. Only the password for the super user is required. All other configuration parameters are defaulted for you.

* Indicates required field

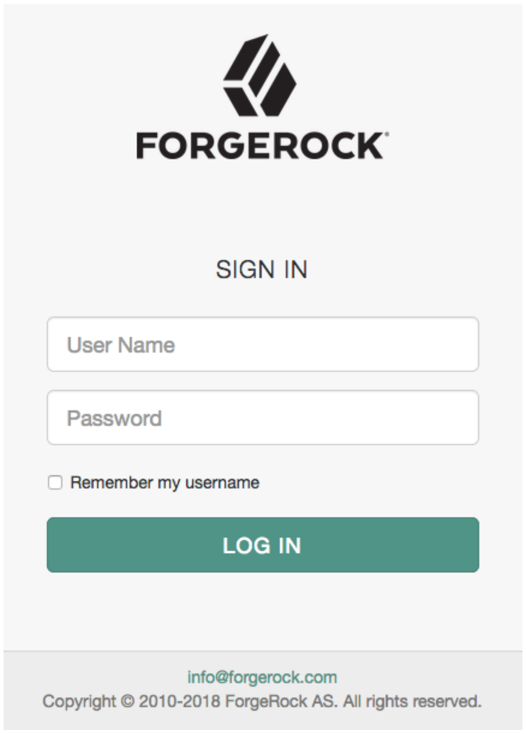
Default User Password

Default User [amAdmin]

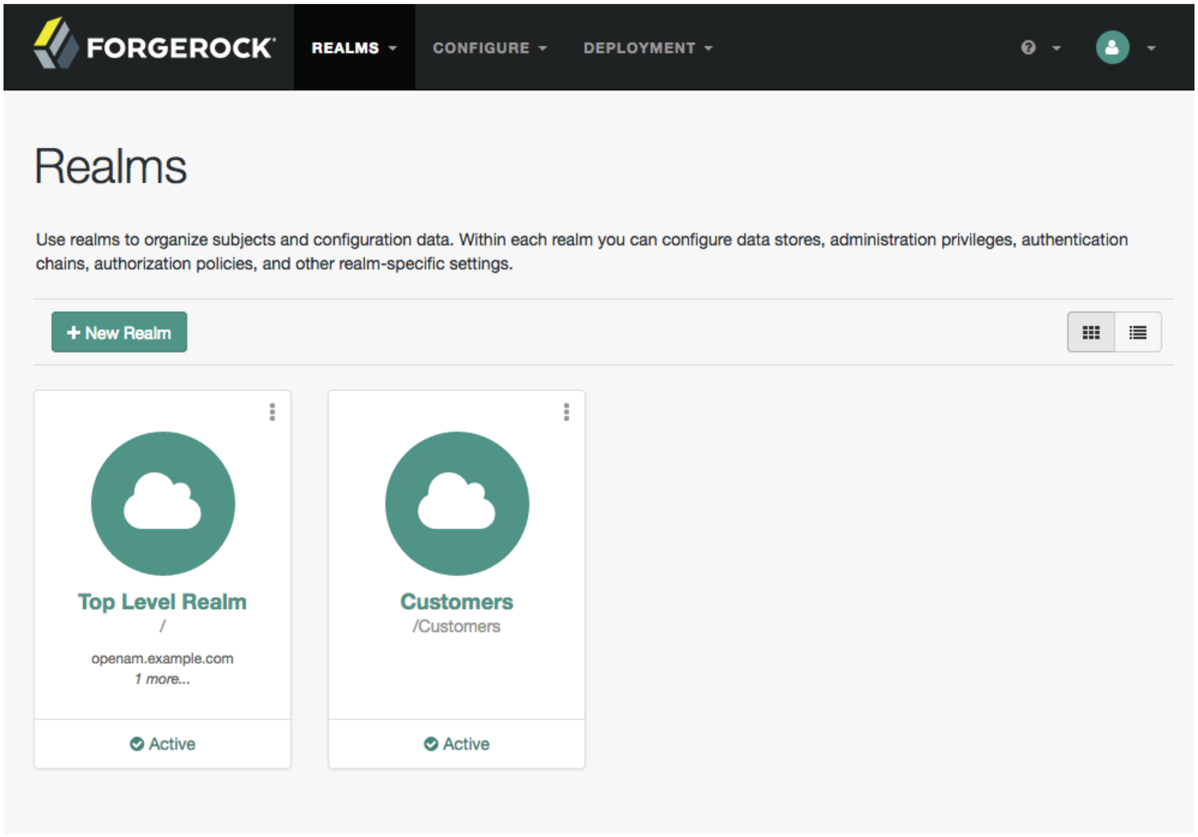
* Password

* Confirm Password

4. When the configuration completes, click Proceed to Login, and then login as the AM administrator, `amAdmin`.



After successful login, AM redirects you to AM Realms.



Realms

Use realms to organize subjects and configuration data. Within each realm you can configure data stores, administration privileges, authentication chains, authorization policies, and other realm-specific settings.

To Delete a Configuration Before Redeploying

If you need to delete your configuration and start the process from the beginning, follow these steps.

1. Stop the AM web application to clear the configuration held in memory.

   The following example shows down Apache Tomcat (Tomcat) for example.

```
$ /path/to/tomcat/bin/shutdown.sh
Password:
Using CATALINA_BASE:   /path/to/tomcat
Using CATALINA_HOME:   /path/to/tomcat
Using CATALINA_TMPDIR: /path/to/tomcat/temp
Using JRE_HOME:        /path/to/jdk/jre
Using CLASSPATH:
       /path/to/tomcat/bin/bootstrap.jar:/path/to/tomcat/bin/tomcat-juli.jar
```

2. Delete AM configuration files, by default under the `$HOME` of the user running the web application container.

```
$ rm -rf $HOME/openam $HOME/.openamcfg
```

When using the internal AM configuration store, this step deletes the embedded directory server and all of its contents. This is why you stop the application server before removing the configuration.

If you use an external configuration store, delete the entries under the configured AM suffix (by default dc=openam,dc=forgerock,dc=org).

3. Delete any cached files from the container. For example, on Tomcat, files are cached in a folder named after the deployment path, such as `/path/to/tomcat/work/Catalina/localhost/deployment path`. Use a command such as the following to delete these cached files:

```
$ rm -rf /path/to/tomcat/work/Catalina/localhost/openam
```

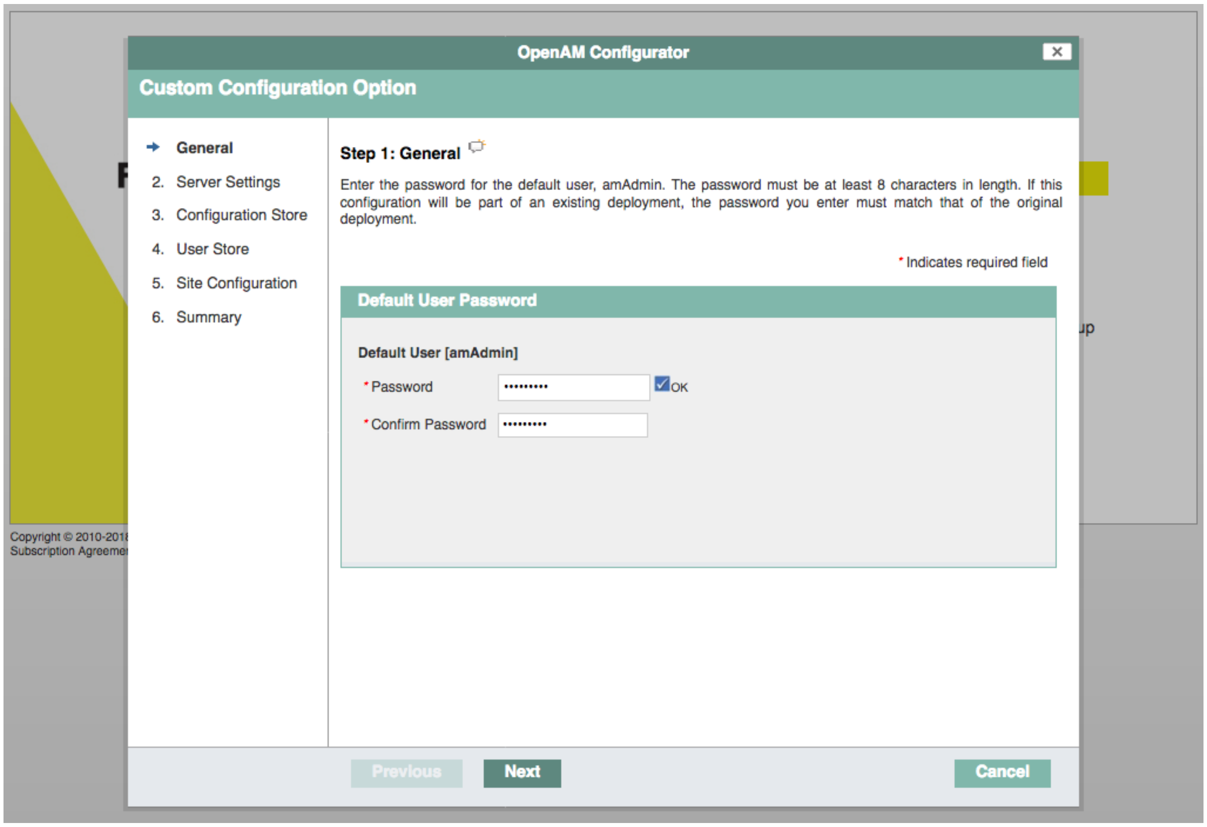
4. Restart the AM web application.

   The following example starts the Tomcat container.

```
$ /path/to/tomcat/bin/startup.sh
Password:
Using CATALINA_BASE:   /path/to/tomcat
Using CATALINA_HOME:   /path/to/tomcat
Using CATALINA_TMPDIR: /path/to/tomcat/temp
Using JRE_HOME:        /path/to/jdk/jre
Using CLASSPATH:
       /path/to/tomcat/bin/bootstrap.jar:/path/to/tomcat/bin/tomcat-juli.jar
```

To Custom Configure an Instance

1. In the initial configuration screen, click Create New Configuration under Custom Configuration.
2. Read the license agreement. If you agree to the license, click "I agree to the license agreement", and then click Continue.
3. On the Default User Password page, provide a password with at least eight characters for the AM Administrator, `amadmin`.



Step 1: General

Enter the password for the default user, amAdmin. The password must be at least 8 characters in length. If this configuration will be part of an existing deployment, the password you enter must match that of the original deployment.

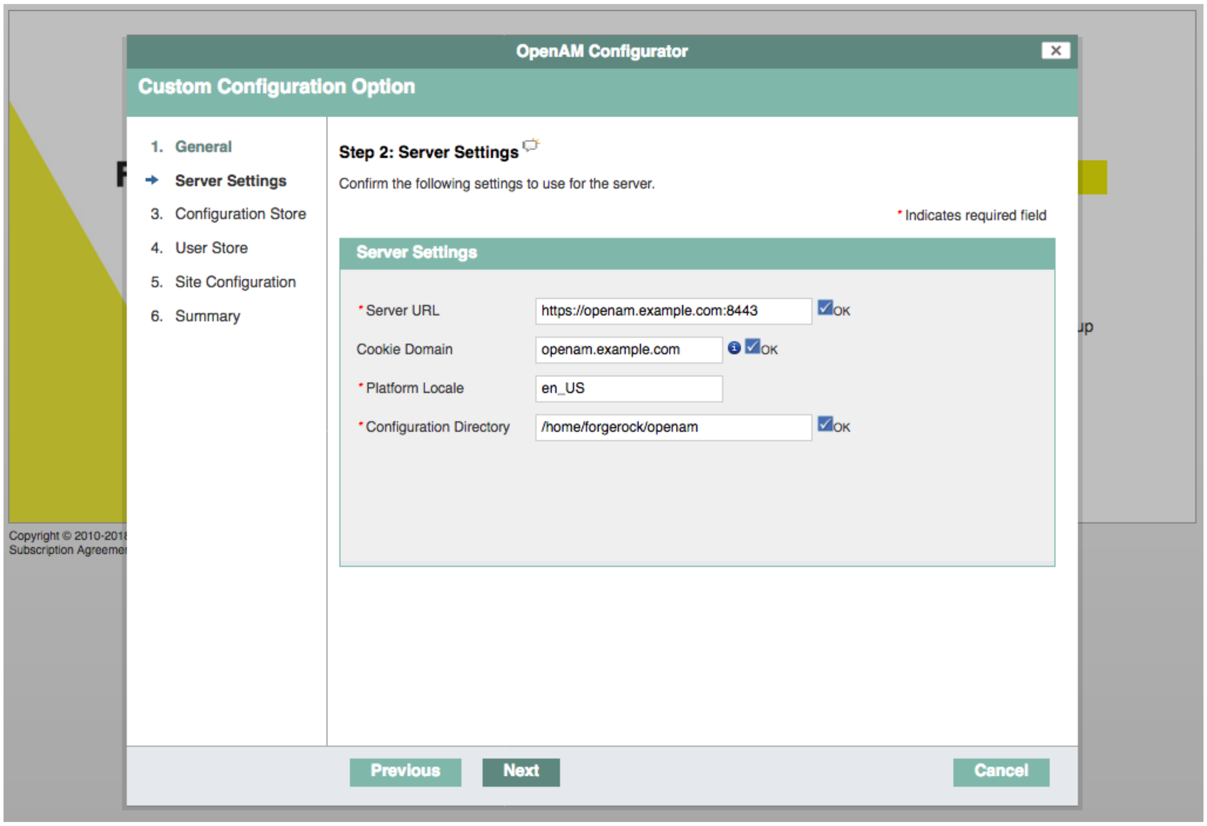
* Indicates required field

Default User Password

Default User [amAdmin]

* Password	••••••••• ✓ OK
* Confirm Password	•••••••••

4. Verify that the server settings are valid for your configuration.



Step 2: Server Settings

Confirm the following settings to use for the server.

* Indicates required field

Server Settings

* Server URL	https://openam.example.com:8443	✓ OK
Cookie Domain	openam.example.com	✓ OK
* Platform Locale	en_US	
* Configuration Directory	/home/forgerock/openam	✓ OK

Server URL

Provide a valid URL to the base of your AM web container, including a FQDN.

In a test environment, you can simulate the FQDN by adding it to your `/etc/hosts` as an alias. The following excerpt shows lines from the `/etc/hosts` file on a Linux system where AM is installed.

```
127.0.0.1 localhost.localdomain localhost
::1 localhost6.localdomain6 localhost6
127.0.1.1 openam openam.example.com
```

Cookie Domain

Domain that created cookies will be valid for, for example `example.com`.

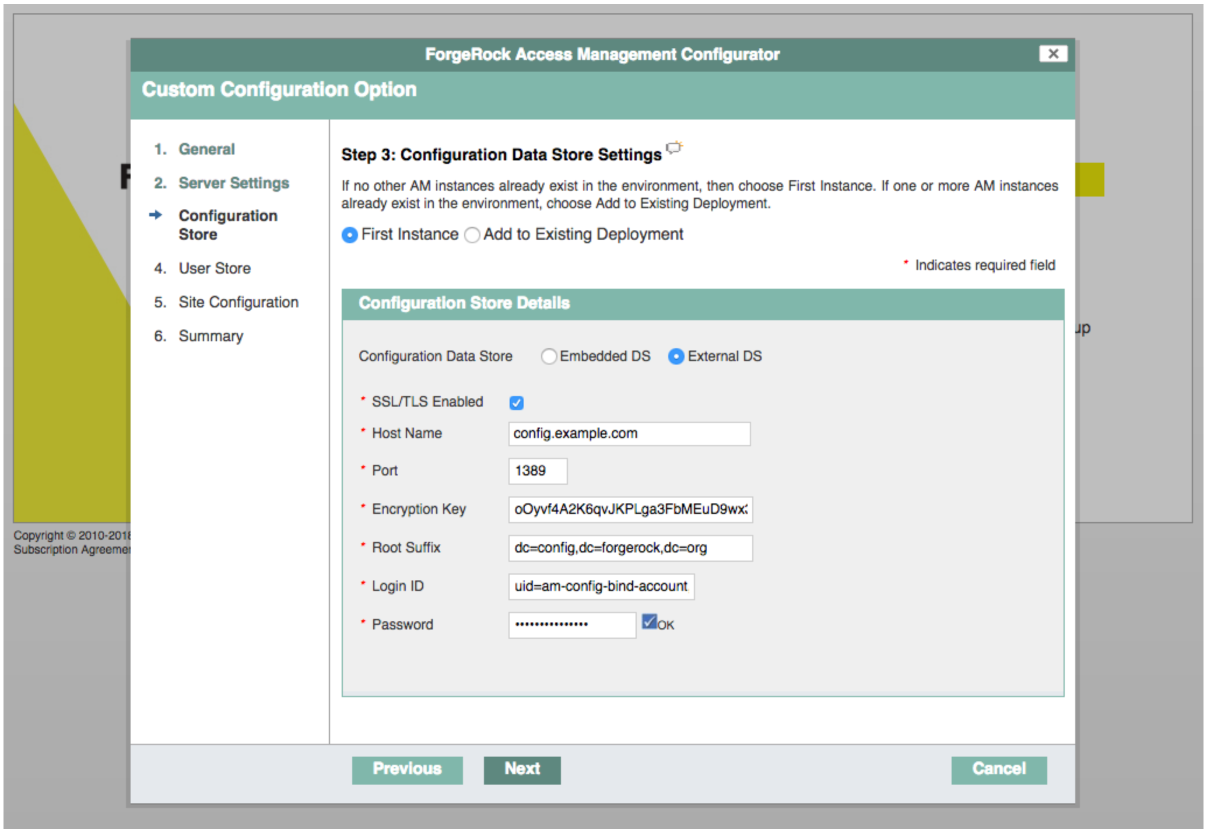
Platform Locale

Supported locales include en_US (English), de (German), es (Spanish), fr (French), ja (Japanese), ko (Korean), zh_CN (Simplified Chinese), and zh_TW (Traditional Chinese).

Configuration Directory

Location on server for AM configuration files. AM must be able to write to this directory.

5. In the Configuration Store screen, you can accept the defaults to allow AM to store configuration data in an embedded directory. The embedded directory can be configured separately to replicate data for high availability if necessary.



Step 3: Configuration Data Store Settings

If no other AM instances already exist in the environment, then choose First Instance. If one or more AM instances already exist in the environment, choose Add to Existing Deployment.

[✓ First Instance] Add to Existing Deployment

* Indicates required field

Configuration Store Details

Configuration Data Store: Embedded DS [✓ External DS]

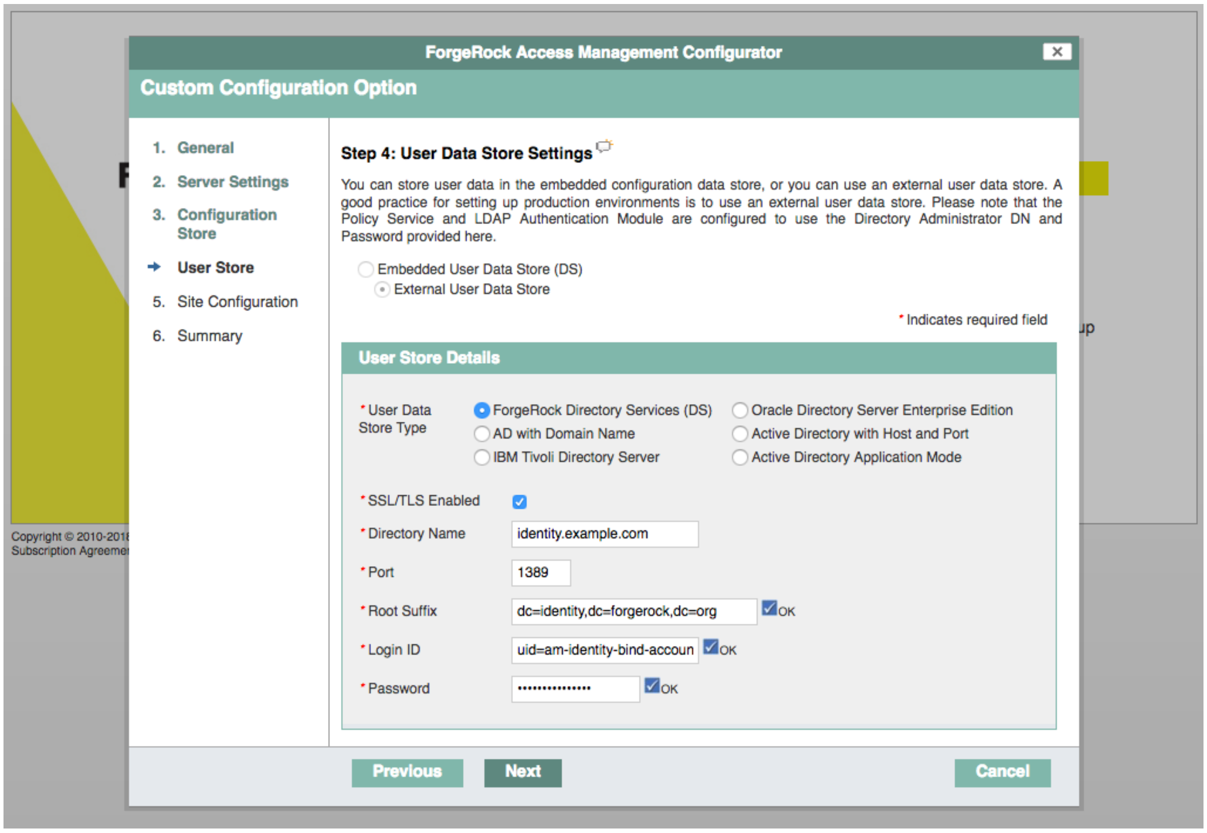
* SSL/TLS Enabled	✓
* Host Name	config.example.com
* Port	1389
* Encryption Key	oOyvf4A2K6qvJKPLga3FbMEuD9wx
* Root Suffix	dc=config,dc=forgerock,dc=org
* Login ID	uid=am-config-bind-account
* Password	•••••••••••••••• ✓OK

You can also add this AM installation to an existing deployment, providing the URL of the site. See "To Add a Server to a Site" for details.

Alternatively, if you already manage an existing DS deployment, you can store AM configuration data in your existing directory server. You must, however, create the suffix to store configuration data on the directory server before you configure AM. AM does not create the suffix when you use an external configuration store.

6. In the User Store screen, you configure where AM looks for user identities.

   AM must have write access to the directory service you choose, as it adds to the directory schema needed to allow AM to manage access for users in the user store.



ForgeRock Access Management Configurator

Custom Configuration Option

1. General
2. Server Settings
3. Configuration Store
→ User Store
5. Site Configuration
6. Summary

Step 4: User Data Store Settings

You can store user data in the embedded configuration data store, or you can use an external user data store. A good practice for setting up production environments is to use an external user data store. Please note that the Policy Service and LDAP Authentication Module are configured to use the Directory Administrator DN and Password provided here.

Embedded User Data Store (DS) [✓ External User Data Store]

* Indicates required field

User Store Details

* User Data Store Type	[✓ ForgeRock Directory Services (DS)] Oracle Directory Server Enterprise Edition AD with Domain Name Active Directory with Host and Port IBM Tivoli Directory Server Active Directory Application Mode
* SSL/TLS Enabled	✓
* Directory Name	identity.example.com
* Port	1389
* Root Suffix	dc=identity,dc=forgerock,dc=org ✓OK
* Login ID	uid=am-identity-bind-accoun ✓OK
* Password	••••••••••••••• ✓OK

User Data Store Type

If you have already provisioned a directory service with users in a supported user data store, then select that type of directory from the options available.

SSL/TLS Enabled

To use a secure connection, check this box, then make sure the port you define corresponds to the port the directory server listens to for StartTLS or SSL connections. When using this option you also need to make sure the trust store used by the JVM running AM has the necessary certificates installed.

Directory Name

FQDN for the host housing the directory service.

Port

LDAP directory port. The default for LDAP and LDAP with StartTLS to protect the connection is port 389. The default for LDAP over SSL is port 636. Your directory service might use a different port.

Root Suffix

Base distinguished name (DN) where user data is stored.

Login ID

Directory administrator user DN. The administrator must be able to update the schema and user data.

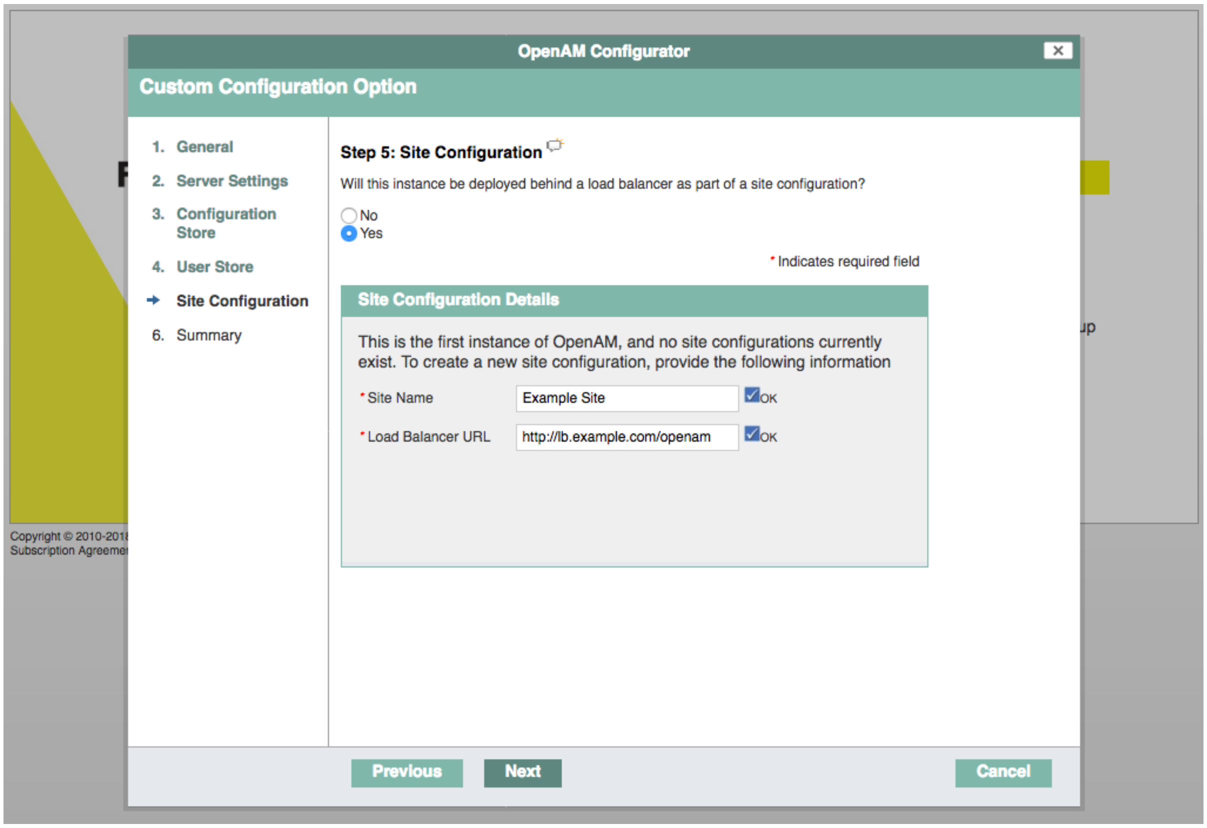
Password

Password for the directory administrator user.

7. In the Site Configuration screen, you can set up AM as part of a site where the load is balanced across multiple AM servers.

   When you deploy multiple servers, AM automatically enables session high availability.[1] AM stores session data in a directory service that is shared by multiple AM servers. The shared storage means that if an AM server fails, other AM servers in the deployment have access to the user's session data and can serve requests about that user. As a result, the user does not have to log in again.

[1] You can configure AM to store sessions in the Core Token Service (CTS) token store or on the client. Because client-based sessions reside in HTTP cookies, they do not need to be retrieved from a persistent data store in the event of a server failure—they can be retrieved from the cookies. AM does not store client-based sessions in the CTS token store. For more information about sessions, see "About Sessions" in the *Authentication and Single Sign-On Guide*.



Step 5: Site Configuration

Will this instance be deployed behind a load balancer as part of a site configuration?

No [✓ Yes]

* Indicates required field

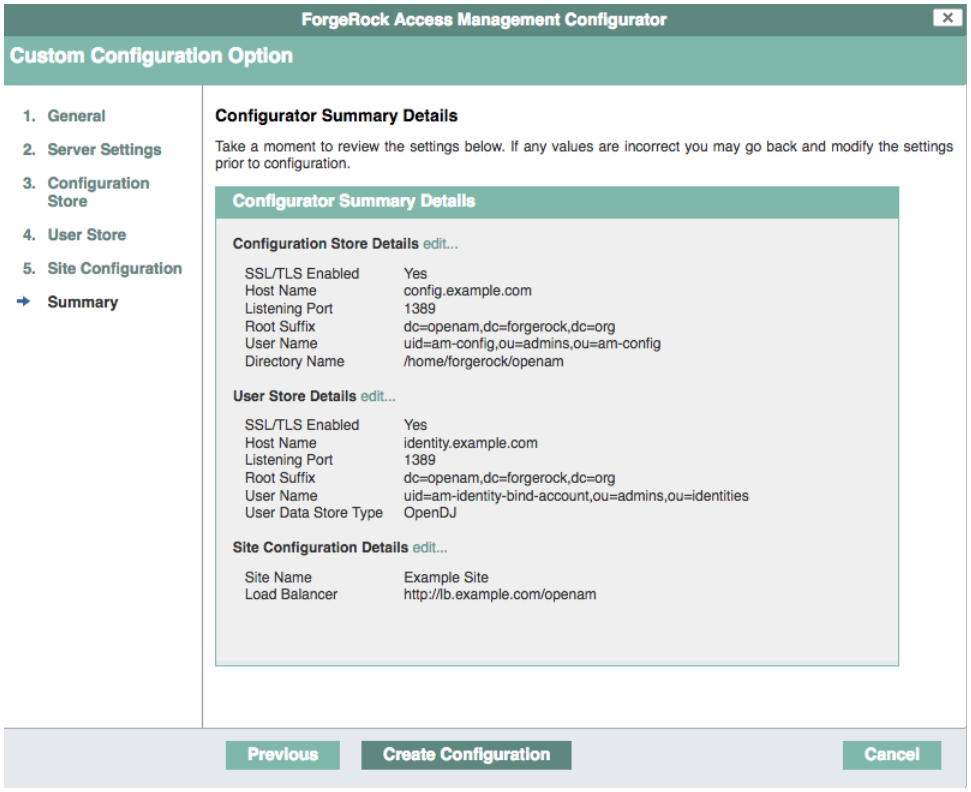
Site Configuration Details

This is the first instance of OpenAM, and no site configurations currently exist. To create a new site configuration, provide the following information

* Site Name	Example Site	✓ OK
* Load Balancer URL	http://lb.example.com/openam	✓ OK

It is possible to set up a site after initial installation and configuration. See "Deployment Configuration" in the *Reference* for more information.

8. Check the summary screen, and if necessary, click Previous to return to earlier screens to fix any configuration errors as needed.



Configurator Summary Details

Take a moment to review the settings below. If any values are incorrect you may go back and modify the settings prior to configuration.

Configurator Summary Details

Configuration Store Details edit...

SSL/TLS Enabled	Yes
Host Name	config.example.com
Listening Port	1389
Root Suffix	dc=openam,dc=forgerock,dc=org
User Name	uid=am-config,ou=admins,ou=am-config
Directory Name	/home/forgerock/openam

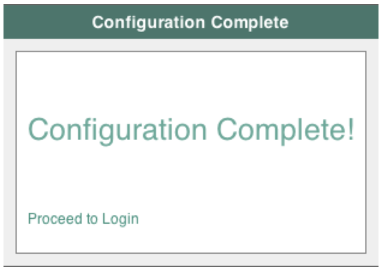
User Store Details edit...

SSL/TLS Enabled	Yes
Host Name	identity.example.com
Listening Port	1389
Root Suffix	dc=openam,dc=forgerock,dc=org
User Name	uid=am-identity-bind-account,ou=admins,ou=identities
User Data Store Type	OpenDJ

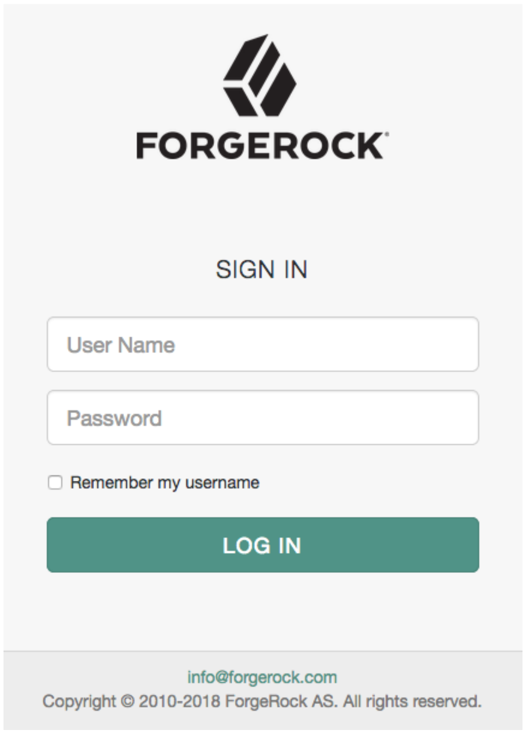
Site Configuration Details edit...

Site Name	Example Site
Load Balancer	http://lb.example.com/openam

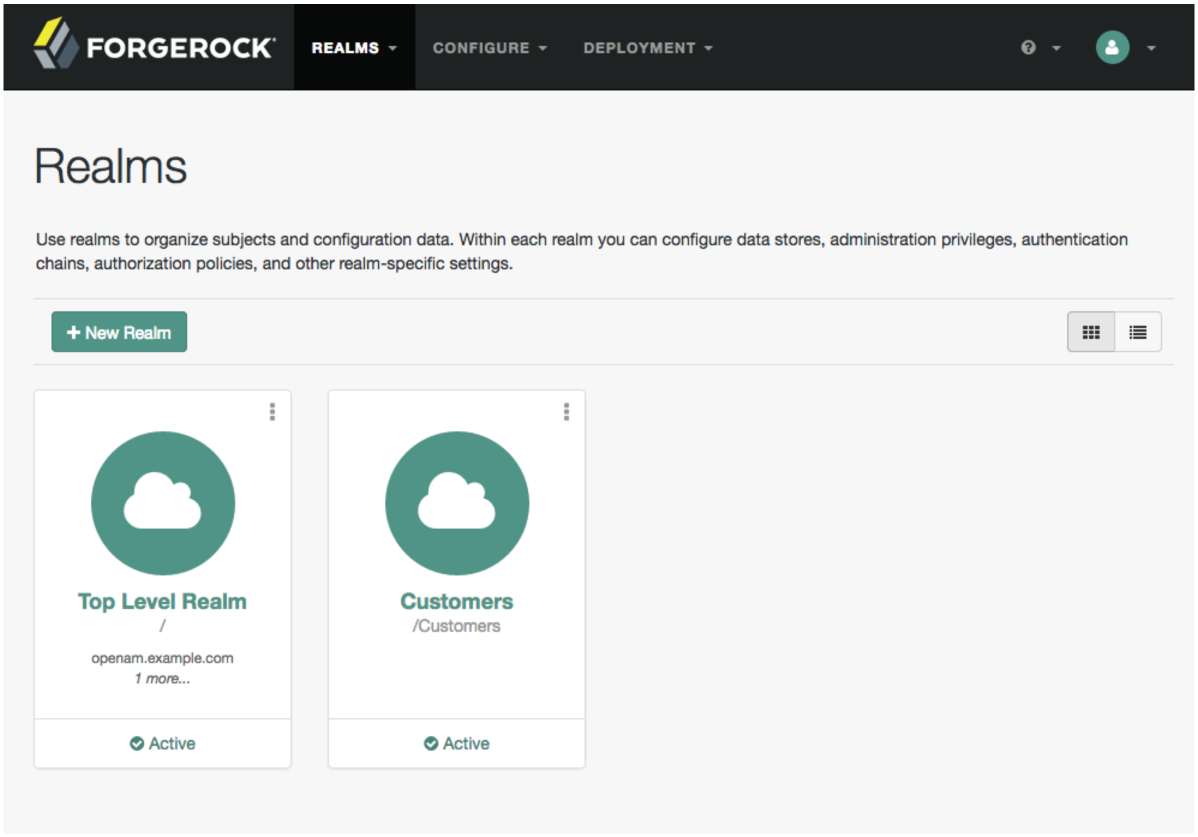
After you click Create Configuration in the summary screen, configuration proceeds, logging progress that you can read in your browser and later, in the installation log. The process ends, and AM shows the Proceed to Login prompt.



9. When the configuration completes, click Proceed to Login, and then login as the AM administrator, `amadmin`.



After login, AM redirects you to the AM Realms page.



Realms

Use realms to organize subjects and configuration data. Within each realm you can configure data stores, administration privileges, authentication chains, authorization policies, and other realm-specific settings.

You can also access the AM console by browsing to the console URL, for example `https://openam.example.com:8443/openam/console`.

10. Restrict permissions to the configuration directory (by default, `$HOME/openam`, where $HOME corresponds to the user who runs the web container). Prevent other users from accessing files in the configuration directory.

11. The AM install wizard uses two Apache libraries that should be removed after installation, for security reasons.

    When your installation 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`

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

Installing Multiple Servers

This chapter covers what to do when installing multiple AM servers.

Things to Consider When Installing Multiple Servers

When installing multiple servers, consider the following points:

- You generally install multiple servers to provide service availability. If one server is down for any reason, another server can respond instead. This means that you need some type of component, such as a load balancer or a proxying server, between incoming traffic and AM to route around servers that are down.

  AM uses a *site* for this purpose. In an AM site, multiple AM servers are configured in the same way, and accessed through a load balancer layer.[2] The load balancer can be implemented in hardware or software, but it is separate and independent from AM software. When installed properly, a site configuration improves service availability, as the load balancer routes around AM servers that are down, sending traffic to other servers in the site.

- The cookie domain is set to the server's full URL that was used to access the configurator, such as `example.net`, `server.west.example.com`, or `helloexample.local`.

- You can use a load balancer layer to protect AM services as well. The load balancer can restrict access to AM services, throttle traffic, offload HTTPS encryption, and so forth.

  As an alternative, or in addition, you can use a separate reverse proxy.

- When you are protecting AM with a load balancer or proxy service, configure your container so that AM can trust the load balancer or proxy service.

- The container for each server in the site must trust any certificate authorities (CA) used to sign certificates used by other servers in the site in order to communicate using SSL.

- AM authentication can depend on information about the user to authenticate, such as the IP address where the request originated. When AM is accessed through a load balancer or proxy layer, pass this information along using request headers. Also, configure AM to consume and to forward the headers as necessary. See "Handling HTTP Request Headers" for details.

Configuring Sites

The most expedient way to configure a server in a site is to set the site up during the initial AM configuration. In the GUI configurator, this is done in the Site Configuration screen.

[2] Technically, it is possible to configure a site with only one AM server.

To Add a Server to a Site

High availability requires redundant servers in case of failure. With AM, you configure an AM site with multiple servers in a pool behind a load balancing service that exposes a single URL as an entry point to the site.

Follow these steps to configure a server to an existing site:

1. Navigate to the deployment URL of the new instance. You should see the AM configurator page.
2. In the initial configuration screen, under Custom Configuration, click Create New Configuration.
3. In the first screen, enter the same password entered for the AM Administrator, `amAdmin`, when you configured the first server in the site.
4. Configure server settings as required.

   The cookie domain should be identical to that of the first server in the site.

   > **Note**
   >
   > The installer may show that the Configuration Directory is not empty; it is a warning in case you are trying to use a directory that contains data not pertaining to AM.

5. In the configuration store screen, select Add to Existing Deployment, and enter the URL of the first AM server in the site.

   The directory used to store configuration data should use the same directory service used for this purpose by other AM servers in the site. If you use the embedded DS server, for example, you can set up the configurator for data replication with embedded directory servers used by other servers in the site.

   Settings for the user store are then shared with the existing server, so the corresponding wizard screen is skipped.

6. In the site configuration screen, select `Yes` and enter the same site configuration details as for the first server in the site.

   Settings for agent information are also shared with the existing server, so the corresponding wizard screen is skipped.

7. In the summary screen, verify the settings you chose, and then click Create Configuration.
8. When the configuration process finishes, **stop** the newly-installed AM instance or the container where it runs, and do not try to access it.
9. Compare the `/path/to/openam/boot.json` bootstrap file with that of a running instance. You must ensure that the newly installed instance's bootstrap file is appropriate for your environment.

Depending on the configuration of the AM keystore in the site, the installation process may not create the bootstrap file. If this is your case,copy the bootstrap file from another instance and continue with the procedure.

Unless your environment has a requirement to configure the AM keystore in a different location on each instance, it is likely that the bootstrap file should be the same across the site.

If you are overriding the start up settings:

- Ensure you have copied the customized bootstrap file from another instance in the site.
- Ensure you are overwriting the existing bootstrap file with your modified file prior to every AM restart.

10. Make the existing AM keystore infrastructure available to the new instance:

    a. Back up the new instance's default keystore and password files. By default, they are located in the `/path/to/openam/openam` directory.

    b. Ensure that the existing keystores in the site are available in the same location to the new instance. This may mean copying the keystores and their password files, mounting a volume, or others.

    c. Ensure that the keystore files configured in the `/path/to/openam/boot.json` file are available to the instance.

       If this keystore and its files are not accessible, the AM instance will not start.

       For more information about the `boot.json` file, see "Startup Settings in the Bootstrap File".

11. Make the existing secret store infrastructure in the site available to the new instance:

    a. Log in to the AM console of an existing instance in the site and navigate to Configure > Secret Stores.

    b. Review the list of secret stores configured globally and make sure to provide the relevant stores to the new instance. For example:

       - For keystore-type secret stores, copy the keystores to the same path on the new instance.
       - For filesystem-type secret stores, copy the contents of the directories to the same path or make the filesystem available on the same mount point on the new instance.
       - For HSM-type stores, ensure the new instance can access it.
       - For secrets configured as environment variables accessible by the container where AM runs, ensure they are also accessible by the container of the new instance.

    c. Navigate to Realms > *Realm Name* > Secret Stores.

d. Review the list of secret stores configured per realm and make sure to provide the relevant stores to the new instance.

12. Restart the new instance.

    The instance is now configured for the site.

13. Ensure that the cookie domain configuration is appropriate for your site. For more information, see "Changing the Cookie Domain" in the *Setup and Maintenance Guide*.

It is also possible to configure a site separately. If you did not set up a site during initial configuration, perform the following steps to configure a site:

To Configure a Site With the First Server

The following steps show how to set up the site for the first AM server.

1. Log in to the AM console as administrator, by default `amadmin`, and then navigate to Deployment > Sites.

2. Click Add a Site to start configuring the new site.

3. On the New Site page enter the site name, and set the Primary URL to the load balancer URL that is the entry point for the site, such as `https://lb.example.com/openam`.

   The site URL is the URL to the load balancer in front of the AM servers in the site. For example, if your load balancer listens for HTTPS on host `lb.example.com` and port `443` with AM under `/openam`, then your site URL is `https://lb.example.com/openam`.

   Client applications and web or Java agents access the servers in the site through the site URL.

4. Click Save to keep the site configuration.

5. Configure the cookie domain of your site as required. For more information, see "Changing the Cookie Domain" in the *Setup and Maintenance Guide*.

6. Navigate to Deployment > Servers > *Server Name* > General.

7. Set the Parent Site drop-down to the name of the site you just created, and save your changes.

   At this point, the server is part of the new site you have configured.

   For all additional servers in the AM site, add them to the site at configuration time as described in "To Add a Server to a Site".

Configuring Load Balancing for a Site

Load balancer configuration requirements differ depending on where you configure AM to store sessions and authentication sessions, as shown in the following table:

Load Balancing Requirements by Session Storage Location

Storage Location	Load Balancing Requirement	Comments
CTS token store	None. Session stickiness recommended for performance	Although the CTS token store is the authoritative source for CTS-based sessions and authentication sessions, AM servers cache the session or authentication session when: • Authenticating a user • Satisfying a session request An AM site configured to use CTS-based sessions achieves the best performance when the server that originally authenticated a user continually manages that user's session, since it does not need to retrieve it from the CTS token store. In the same way, an AM realm configured to use CTS-based authentication sessions achieves the best performance when the same server manages every request for the same authentication flow.
Client	None. Session stickiness recommended for performance	Although the user's session or authentication session resides in a JWT stored on the client which is passed to AM server along with the request, client-based sessions should be signed and/or encrypted for security reasons. Because decrypting and verifying the session on each request may be an expensive operation depending on the algorithms configured, AM caches the decrypt sequence in memory to improve performance. Therefore, an AM site configured to use client-based sessions achieves best performance when the same server manages every request for the same session or authentication session.
In AM's memory [a]	Session stickiness required	Deployments where AM stores authentication sessions in memory require sticky load balancing to route all requests pertaining to a particular authentication flow to the same AM server. If a request reaches a different AM server, the authentication flow will start anew.

[a]Authentication sessions only.

For more information about AM session and authentication sessions, see "About Sessions" in the *Authentication and Single Sign-On Guide*.

Session storage location can be heterogeneous within the same AM deployment to suit the requirements of each of your realms. If your deployment uses a substantial number of CTS-based sessions, follow the recommendations for deployments configured for CTS-based sessions.

Configuring Site Sticky Load Balancing

Configure AM for sticky load balancing as follows:

To Configure Site Load Balancing

1. Navigate to Deployment > Sites.

2. Ensure you have a site created and that your servers are part of it. For more information, see "To Configure a Site With the First Server".

3. Ensure that the `amlbcookie` cookie has a unique value for each AM server:

   a. Navigate to Deployment > Servers > *Server Name* > Advanced and review the value of the `com.iplanet.am.lbcookie.value` property. By default, the cookie value is set to the AM server ID.

      Keep the value of the `amlbcookie` cookie set to the AM server ID to improve server performance when using Web Agents.

      If you have replaced the value of the this property and you need to match the AM server URLs with their corresponding server IDs, query the `global-config/servers` endpoint. For example:

      ```
      $ curl \
      --header 'Accept: application/json' \
      --header "iPlanetDirectoryPro: AQIC5...NDU1*" \
      --header  "Accept-API-Version: resource=1.0, protocol=2.1" \
      'https://openam.example.com:8443/openam/json/global-config/servers?_queryFilter=true'
      {
          "result": [
              {
                  "_id": "01",
                  "_rev": "-1541617246",
                  "siteName": null,
                  "url": "https://openam.example.com:8443/openam"
              }
          ],
          "resultCount": 1,
          "totalPagedResults": -1,
          "totalPagedResultsPolicy": "NONE"
      }
      ```

      In the example above, the server ID for server `https://openam.example.com:8443/openam` is `01`.

      Changes take effect only after you restart the AM server.

   b. Restart the AM server. You can then check the cookie value by logging in to the AM console, and examining the `amlbcookie` cookie in your browser.

4. Repeat the previous steps for each of the AM servers that should be part of the site.

5. Configure your load balancer to perform sticky load balancing based on the `amlbcookie` value.

   In other words, the load balancer layer must keep track of which `amlbcookie` cookie value corresponds to which AM server.

When the load balancer receives a request, it inspects the value of the `amlbcookie` cookie, and then forwards the request to the corresponding AM server.

Load Balancer Termination

When traffic to and from the load balancer is protected with HTTPS, the approach described in "To Configure Site Load Balancing" requires that you terminate the connection on the load balancer. You then either re-encrypt the traffic from the load balancer to AM, or make connections from the load balancer to AM over HTTP.

Request Forwarding Caveats

Sticky load balancing based on the value of the `amlbcookie` cookie does not guarantee request forwarding to the corresponding AM server in all cases. For example, ForgeRock Common REST API calls do not typically use cookies. Therefore, load balancers are not able to route these calls to the AM server on which a user's session is cached.

The AM server that does not hold the user's session in cache must locate the user's session by retrieving it from the Core Token Service's token store.

Handling HTTP Request Headers

HTTP requests can include information needed for access management, such as the client IP address used for adaptive risk-based authentication.

Configure your load balancer or proxy to pass the information to AM by using request headers. For example, the load balancer or proxy can send the client IP address by using the `X-Forwarded-For` HTTP request header.

Also configure AM to consume and to forward the headers as necessary. For example, to configure AM to look for the client IP address in the `X-Forwarded-For` request header, set the advanced configuration property `com.sun.identity.authentication.client.ipAddressHeader` to `X-Forwarded-For` under Deployment > Servers > *Server Name* > Advanced.

In a site configuration where one AM server can forward requests to another AM server, you can retain the header by adding it to the advanced configuration property `openam.retained.http.request.headers`. If `X-Forwarded-For` is the only additional header to retain, set `openam.retained.http.request.headers` to `X-DSAMEVersion,X-Forwarded-For`, for example.

Configure these properties under Deployment > Servers > *Server Name* > Advanced.

Handling Multiple Cookie Domains When Using Wildfly

If you are using Wildfly as the AM web container with multiple cookie domains, you must set the advanced server property, `com.sun.identity.authentication.setCookieToAllDomains`, to `false`.

Set this property in the AM console under Configure > Server Defaults > Advanced.

Installing and Using the Tools

The tools are found in `.zip` files where you unpacked the archive of the entire package, such as `~/Downloads/openam`.

The `.zip` files containing AM tools are:

`SSOAdminTools-5.1.2.24.zip`

Administration tools: **ampassword**, **ssoadm**, and **amverifyarchive**

See "Setting Up Administration Tools".

`SSOConfiguratorTools-5.1.2.24.zip`

Configuration and upgrade tools, alternatives to using the GUI configuration wizard

See "Setting up Configuration Tools" and "Installing Silently".

Setting Up Administration Tools

The **ssoadm** administration tool requires access to AM configuration files, and therefore must be installed on the same host as AM core services.

To Set Up Administration Tools

1. Verify that AM is installed and running before proceeding.
2. Verify that the `JAVA_HOME` environment variable is set properly:

```
$ echo $JAVA_HOME
/path/to/jdk
```

3. Create a file system directory to unpack the tools:

```
$ mkdir -p /path/to/openam-tools/admin
```

4. Unpack the tools:

```
$ cd /path/to/openam-tools/admin
$ unzip ~/Downloads/openam/SSOAdminTools-5.1.2.24.zip
```

5. Add `--acceptLicense` to the **java** command at the end of the **setup** or **setup.bat** script if you want to auto-accept the license agreement and suppress the license acceptance screen to the user:

```
$JAVA_HOME/bin/java -D"load.config=yes" \
                    -D"help.print=$help_print" \
                    -D"path.AMConfig=$path_AMConfig" \
                    -D"path.debug=$path_debug" \
                    -D"path.log=$path_log" \
                    -cp "$CLASSPATH" com.sun.identity.tools.bundles.Main \
                    --acceptLicense
```

6. (Optional) If you use IBM Java, add `-D"amCryptoDescriptor.provider=IBMJCE"` and `-D"amKeyGenDescriptor.provider=IBMJCE"` options to the **setup** or **setup.bat** script before you install the tools.

   The options should be set for the **java** command at the end of the script:

```
$ tail setup
CLASSPATH="$CLASSPATH:resources"

$JAVA_HOME/bin/java -D"load.config=yes" \
                    -D"help.print=$help_print" \
                    -D"path.AMConfig=$path_AMConfig" \
                    -D"path.debug=$path_debug" \
                    -D"path.log=$path_log" \
                    -D"amCryptoDescriptor.provider=IBMJCE" \
                    -D"amKeyGenDescriptor.provider=IBMJCE" \
                    -cp "$CLASSPATH" \
                    com.sun.identity.tools.bundles.Main
```

7. Run the **setup** utility (**setup.bat** on Windows), providing paths to the directories where AM configuration files are located, and where debug and log information will be located:

```
$ ./setup
Path to config files of OpenAM server [/home/user/openam]:
Debug Directory [/path/to/openam-tools/admin/debug]:
Log Directory [/path/to/openam-tools/admin/log]:
The scripts are properly setup under directory:
 /path/to/openam-tools/admin/openam
Debug directory is /path/to/openam-tools/admin/debug.
Log directory is /path/to/openam-tools/admin/log.
The version of this tools.zip is: version and date
The version of your server instance is: OpenAM version and date
```

   After setup, the tools are located under a directory named after the instance of AM:

```
$ ls openam/bin/
ampassword  amverifyarchive  ssoadm
```

   On Windows, these files are `.bat` scripts.

8. (Optional) If your web container uses a self-signed certificate as described in "To Set Up With HTTPS and Self-Signed Certificates", then the **ssoadm** command will not trust the certificate when connecting to AM over HTTPS, or when AM connects to the configuration store over LDAPS.

   To allow the **ssoadm** command to trust the certificate, add the `-D"javax.net.ssl.trustStore=/path/to/tomcat/conf/keystore.jks"` option to the **ssoadm** or **ssoadm.bat** script before using the script.

The option should be set before the call to `com.sun.identity.cli.CommandManager` at the end of the script:

```
$ tail -2 /path/to/openam-tools/admin/openam/bin/ssoadm
    -D"javax.net.ssl.trustStore=/path/to/tomcat/conf/keystore.jks" \
    com.sun.identity.cli.CommandManager "$@"
```

> **Note**
>
> In non-production environments, you can configure the **ssoadm** command to trust all server certificates. For more information, see Q. How do I configure ssoadm to trust all certificates? in the *ForgeRock Knowledge Base*.

9. (Optional) If you use IBM Java, add `-D"amCryptoDescriptor.provider=IBMJCE"` and `-D"amKeyGenDescriptor.provider=IBMJCE"` options to the **ssoadm** or **ssoadm.bat** script before using the script.

   The options should be set before the call to `com.sun.identity.cli.CommandManager` at the end of the script:

   ```
   $ tail -3 /path/to/openam-tools/admin/openam/bin/ssoadm
       -D"amCryptoDescriptor.provider=IBMJCE" \
       -D"amKeyGenDescriptor.provider=IBMJCE" \
       com.sun.identity.cli.CommandManager "$@"
   ```

10. Check that the **ssoadm** command works properly:

    a. Create a text file, for example `$HOME/.pwd.txt`, containing the AM administrative user's password string in cleartext on a single line.

    b. Make the text file read-only:

    ```
    $ chmod 400 $HOME/.pwd.txt
    ```

    c. Run the **ssoadm** command to list the configured servers:

    ```
    $ cd /path/to/openam-tools/admin/openam/bin/
    $ ./ssoadm list-servers -u amadmin -f $HOME/.pwd.txt
    https://openam.example.com:8443/openam
    ```

11. If you have deployed AM in a site configuration, edit the **ssoadm** (**ssoadm.bat** on Windows) script to map the site URL to the AM server URL.

    To do this, set the `com.iplanet.am.naming.map.site.to.server` system property as a **java** command option in the script. The option takes the following form:

    ```
    -D"com.iplanet.am.naming.map.site.to.server=lb-url=openam-url[,
     other-lb-url=openam-url ...]"
    ```

    The property maps each *lb-url* key to an *openam-url* value, where *lb-url* is the URL to a site load balancer, and *openam-url* is the URL to the AM server against which you set up the **ssoadm** command.

Important

The **ssoadm** command is dependent on the AM server against which you set it up, so always map site load balancer URLs to that server's *openam-url*.

For example, if your site is behind `https://lb.example.com:443/openam`, and the AM server against which you set up the **ssoadm** command is at `https://openam.example.com:8443/openam`, then add the following property to the **java** command (all on one line without spaces):

```
-D"com.iplanet.am.naming.map.site.to.server=
 https://lb.example.com:443/openam=https://openam.example.com:8443/openam"
```

Repeat this step for each AM server in your site configuration. You can install all your instances of **ssoadm** on the same host, but in each case the command should manage only one AM server.

Setting up Configuration Tools

This section covers setting up the configuration and upgrade tools, which are alternatives to using the GUI configuration wizard.

To Set up Configuration Tools

1. Verify that the `JAVA_HOME` environment variable is properly set:

```
$ echo $JAVA_HOME
/path/to/jdk
```

2. Create a file system directory to unpack the tools:

```
$ mkdir -p /path/to/openam-tools/config
```

3. Unpack the tools from where you unzipped AM:

```
$ cd /path/to/openam-tools/config
$ unzip ~/Downloads/openam/SSOConfiguratorTools-5.1.2.24.zip
Archive:  ~/Downloads/openam/SSOConfiguratorTools-5.1.2.24.zip
   creating: legal-notices/
  inflating: legal-notices/LICENSE.DOM-software.html
  inflating: legal-notices/NOTICE.resolver.txt
  inflating: legal-notices/LICENSE.DOM-documentation.html
        ... (more output) ...
 extracting: lib/xml-apis-2.11.0.jar
 extracting: openam-configurator-tool-14.1.2.24.jar
 extracting: lib/servlet-api-2.5.jar
```

Installing Silently

Use the AM configurator tool, `openam-configurator-tool-14.1.2.24.jar`, to silently install AM. The AM server must be deployed and running, but not yet configured, when you use the tool.

To Silently Install

Perform the following steps:

1. Verify that the `JAVA_HOME` environment variable is properly set:

   ```
   $ echo $JAVA_HOME
   /path/to/jdk
   ```

2. The configurator tool relies on a property file to specify the configuration for the AM server. For property file options, see configurator.jar(1).

   Copy the sample configuration property file provided with AM, and then modify properties as needed:

   ```
   $ cd /path/to/openam-tools/config
   $ cp sampleconfiguration config.properties
   $ grep -v "^#" config.properties | grep -v "^$"
   SERVER_URL=http://openam.example.com:8080
   DEPLOYMENT_URI=/openam
   BASE_DIR=/home/openam/openam
   locale=en_US
   PLATFORM_LOCALE=en_US
   AM_ENC_KEY=
   ADMIN_PWD=password
   COOKIE_DOMAIN=openam.example.com
   ACCEPT_LICENSES=true
   DATA_STORE=embedded
   DIRECTORY_SSL=SIMPLE
   DIRECTORY_SERVER=openam.example.com
   DIRECTORY_PORT=50389
   DIRECTORY_ADMIN_PORT=4444
   DIRECTORY_JMX_PORT=1689
   ROOT_SUFFIX=dc=openam,dc=forgerock,dc=org
   DS_DIRMGRDN=cn=Directory Manager
   DS_DIRMGRPASSWD=password
   ```

   When setting options in the property file, note the following:

   - If you include the `ACCEPT_LICENSES=true` property, AM auto-accepts the software licensing agreement and suppresses display of the license acceptance screen during silent installation.
   - When installing AM to support HTTPS, make sure the `SERVER_URL` property specifies a URL with HTTPS, and set the `DIRECTORY_SSL` property to `SIMPLE`.

3. Run the AM configurator tool, `openam-configurator-tool-14.1.2.24.jar`:

   ```
   java -jar openam-configurator-tool-14.1.2.24.jar --file config.properties
   ```

   If required, you can specify additional run-time options on the command line:

- With the `--acceptLicense` option, the installer auto-accepts the software licensing agreement and suppresses the display of the license acceptance screen, resulting in the same behavior as specifying `ACCEPT_LICENSES=true` in the configuration property file.
- The `-Djavax.net.ssl.trustStore=PATH_TO_JKS_TRUSTSTORE` option is required when installing AM to support HTTPS. Specify the AM web container's trust store for `PATH_TO_JKS_TRUSTSTORE`.

Output similar to the following appears:

```
$ java -jar openam-configurator-tool-14.1.2.24.jar --file config.properties
Checking license acceptance...License terms accepted.
Checking configuration directory /home/openam/openam....Success.
Installing OpenAM configuration store...Success RSA/ECB/OAEPWithSHA1AndMGF1...
Extracting OpenDJ, please wait...Complete
Running OpenDJ setupSetup command: --cli --adminConnectorPort 4444
 --baseDN dc=openam,dc=forgerock,dc=org --rootUserDN cn=Directory Manager
 --ldapPort 50389 --skipPortCheck --rootUserPassword xxxxxxx --jmxPort 1689
 --no-prompt --doNotStart --hostname openam.example.com ...
...Success
Installing OpenAM configuration store in /home/openam/openam/... ...Success.
Creating OpenAM suffixImport+task+ ... ...Success
Tag swapping schema files....Success.
Loading Schema opendj_config_schema.ldif...Success.

...

...Success.
Reinitializing system properties....Done
Registering service dashboardService.xml...Success.

...

Configuring system....Done
Configuring server instance....Done
Creating demo user....Done
Creating Web Service Security Agents....Done
Setting up monitoring authentication file.
Configuration complete!
```

Starting Servers

AM is a web application installed in a web container, such as Apache Tomcat. Starting the web container starts the AM application.

At the beginning of its startup process, AM performs an operation called *bootstrapping*, during which AM obtains startup settings from a bootstrap file in its configuration directory, then uses those settings to initiate its operation. AM creates the bootstrap file, `boot.json`, during installation.

The installation or upgrade process creates the file after configuring the instance, provided it can find the AM keystore and its password files in either of the following locations:

- Configure > Server Defaults > Security > Key Store

- Deployment > Servers > *Server Name* > Security > Key Store

ForgeRock recommends changing the AM default keystore configuration at Server Default level, so that the environment is homogeneous.

After every successful startup, AM rewrites the bootstrap file using the current information for the AM keystore.

If you change the configuration of the AM keystore, for example, the path to its files, AM will save the changes to the bootstrap file the next time it starts successfully. This is why, if you want to override AM's startup settings, you need to replace the bootstrap file manually before AM starts.

Overriding Startup Settings

Users who deploy AM with DevOps tooling—for example, Docker and Kubernetes—might want to launch multiple AM servers from a single image, providing startup settings dynamically when AM starts up instead of reading the settings from the bootstrap file created during AM installation.

You can replace the bootstrap file and provide your own static and dynamic startup settings. The following sections describe how to override the bootstrap file created during AM installation:

- "Replacing the Bootstrap File" covers how to specify a custom bootstrap file, and describes all the startup settings in the bootstrap file.
- "Overriding Startup Settings by Using Environment Variables" covers how to dynamically override startup settings in the bootstrap file with environment variables.
- "Overriding Startup Settings by Using Java Properties" covers how to dynamically override startup settings in the bootstrap file with Java properties.

Replacing the Bootstrap File

AM's bootstrap file is located at the path `/path/to/openam/boot.json`, where `/path/to/openam` is the AM configuration directory. The AM configuration directory is specified during during AM installation, as follows:

- In the Configuration Directory field on the Server Settings page when using GUI installation. See "To Custom Configure an Instance" for details.
- In the `BASE_DIR` property in the installation configuration file when using command-line installation. See configurator.jar(1) for more information.

To override AM's startup configuration, modify the bootstrap file, `boot.json`, and then overwrite the existing bootstrap file with your modified file *prior to every AM restart*. You must overwrite the file each time you start AM because after startup, AM overwrites the bootstrap file with the initial startup settings created during AM installation, removing any modifications you might have made to startup settings in the bootstrap file.

Make changes to supporting files and passwords before changing bootstrap file properties—AM will fail to start up when bootstrap file properties do not correspond to actual configuration. For example, if you change the value of the `keyStorePasswordFile` property to a file that does not exist, AM will not be able to start up.

The following is an example AM bootstrap file:

```
{
  "instance" : "https://openam.example.com:8443/openam",
  "dsameUser" : "cn=dsameuser,ou=DSAME Users,dc=openam,dc=forgerock,dc=org",
  "keystores" : {
    "default" : {
      "keyStorePasswordFile" : "/home/openam/openam/.storepass",
      "keyPasswordFile" : "/home/openam/openam/.keypass",
      "keyStoreType" : "JCEKS",
      "keyStoreFile" : "/home/openam/openam/keystore.jceks"
    }
  },
  "configStoreList" : [ {
    "baseDN" : "dc=openam,dc=forgerock,dc=org",
    "dirManagerDN" : "cn=Directory Manager",
    "ldapHost" : "opendj.example.com",
    "ldapPort" : 1389,
    "ldapProtocol" : "ldap"
  } ]
}
```

The AM bootstrap file has the following properties:

Startup Settings in the Bootstrap File

Property	Description and Derivation
`instance`	AM server URL. Defaults to the Server URL field on the Server Settings page (GUI configurator) or the `SERVER_URL` configuration property (command-line configurator). This property's value is the URL for directly accessing an AM server instance, *not* an AM site using a load balancer URL. Do not modify this bootstrap file property. If you need to change the AM server URL, reinstall AM.
`dsameUser`	Special AM user. The first part of the user's DN is always created initially as `cn=dsameuser, ou=DSAME Users`. The second part of the DN defaults to the Root Suffix field on the Configuration Data Store Settings page (GUI configurator) or the `ROOT_SUFFIX` configuration property (command-line configurator).
`keystores.default`	The AM keystore. Currently, no other keystores are referenced in the bootstrap file.
`keystores.default.keyStorePasswordFile`	Path to the file that contains the password required to open the AM keystore. Always created initially as `/path/to/openam/openam/.storepass`.

Property	Description and Derivation
	When creating a new `.storepass` file, ensure that there are no hidden trailing characters after the password. For example, use the **echo -n** command to add the password to the new file.
`keystores.default.keyPasswordFile`	Path to the file that contains the password used to encrypt individual keystore entries. Always created initially as `/path/to/openam/openam/.keypass`. When creating a new `.keypass` file, ensure that there are no hidden trailing characters after the password. For example, use the **echo -n** command to add the password to the new file.
`keystores.default.keyStoreType`	AM key store type. Currently, the only valid value is `JCEKS`.
`keystores.default.keyStoreFile`	Path to the AM keystore. Always created initially as `/path/to/openam/openam/keystore.jceks`. The AM keystore is required for startup because it contains the password of the directory manager user of the AM configuration store.
`configStoreList[*]`	Array of one or more objects that describe AM configuration stores. The initial object in the array is mandatory and defines the primary configuration store. Additional objects are optional and define failover configuration stores.
`configStoreList[*].baseDN`	Root suffix of the AM configuration store. Defaults to the Root Suffix field on the Configuration Data Store Settings page (GUI configurator) or the `ROOT_SUFFIX` configuration property (command-line configurator).
`configStoreList[*].dirManagerDN`	DN of the configuration store directory manager user. Defaults to `cn=Directory Manager` (GUI configurator) or the `DS_DIRMGRDN` configuration property (command-line configurator).
`configStoreList[*].ldapHost`	Fully-qualified domain name (FQDN) of the configuration store's host. Defaults to the Host Name field on the Configuration Data Store Settings page (GUI configurator) or the `DIRECTORY_SERVER` configuration property (command-line configurator).
`configStoreList[*].ldapPort`	LDAP or LDAPS port number on which to access the configuration store. Defaults to the Port field on the Configuration Data Store Settings page (GUI configurator) or the `DIRECTORY_PORT` configuration property (command-line configurator).
`configStoreList[*].ldapProtocol`	Protocol with which to access the directory service running the configuration store. The value can be `ldap` or `ldaps`. Defaults to the SSL/TLS Enabled field on the Configuration Data Store Settings page (GUI configurator) or the `DIRECTORY_SSL` configuration property (command-line configurator).

Overriding Startup Settings by Using Environment Variables

You can dynamically override startup settings in the bootstrap file by defining environment variables in the shell that starts AM and referencing the variables in a modified version of the bootstrap file.

Specify JSON properties that reference environment variables in a modified bootstrap file that uses the notation `${env.MY_ENVIRONMENT_VARIABLE}`.

For example, you could dynamically change the AM instance URL as follows:

To Override Startup Settings by Using Environment Variables

1. Set an environment variable named `MY_INSTANCE` in the shell that starts AM.
2. Create a modified version of the bootstrap file with the following line:

   ```
   "instance" : "${env.MY_INSTANCE}",
   ```

3. Overwrite the initial bootstrap file with the modified bootstrap file.
4. Start AM.

Overriding Startup Settings by Using Java Properties

You can dynamically override startup settings in the bootstrap file by referencing Java system properties in a modified version of the bootstrap file. You can reference both built-in Java system properties and properties specified with the `-D` option in the web container that runs AM.

Specify JSON properties that reference Java properties in a modified bootstrap file that uses the notation `${MY_JAVA_PROPERTY}`.

For example, you could dynamically change the AM keystore's path to the user's home directory as follows:

To Override Startup Settings by Using Java Properties

1. Create a modified version of the bootstrap file, specifying the default AM keystore as follows:

   ```
   "keystores" : {
       "default" : {
         "keyStorePasswordFile" : "${usr.home}/.storepass",
         "keyPasswordFile" : "${usr.home}/.keypass",
         "keyStoreType" : "JCEKS",
         "keyStoreFile" : "${usr.home}/keystore.jceks"
       }
     },
   ```

2. Overwrite the initial bootstrap file with the modified bootstrap file.
3. Start AM.

Chapter 4
Implementing the Core Token Service

AM's Core Token Service (CTS) provides generalized, persistent, and highly available storage for sessions and tokens used by AM. AM uses CTS as the authoritative source for CTS-based sessions and caches these sessions in its memory heap to improve performance.[1]

CTS supports *session high availability*, which lets AM manage a session as long as one of the AM servers in a clustered deployment is available. After a user has successfully authenticated, AM creates a CTS-based session. Any AM instance that is configured to use the same CTS can retrieve the session and allow access to it. The user does not need to log in again unless the entire deployment goes down.[2]

CTS provides storage for the following:

- AM CTS-based sessions and authentications sessions
- OAuth 2.0
- UMA 2.0
- Session blacklist (if enabled for client-based sessions)
- Authentication session whitelist (if enabled for client-based sessions)
- SAML v2.0 (if enabled for Security Token Service token validation and cancellation)
- OAuth 2.0 CTS-based tokens and client-based token blacklist
- Push notification during authentication
- Cluster-wide notification

CTS Deployment Architectures

You can deploy CTS token stores in a number of deployment architectures depending on your system requirements. In particular, four types of deployment exist for CTS:

- "CTS Token Store Deployment Using Embedded Configuration Store"
- "CTS Active-Passive Deployment"
- "CTS Affinity Deployment"
- "CTS Site Deployment"

[1]Prior to AM 5, the authoritative source for sessions not stored in the client was the memory heap of AM's web container.
[2]Prior to AM 5, session high availability, formerly referred to as session failover, was optional. Starting with AM 5, session high availability is the default behavior in AM and cannot be disabled.

CTS Token Store Deployment Using Embedded Configuration Store

By default, AM writes CTS entries in the AM configuration store: either an embedded or external configuration store. If you configured AM to use an embedded configuration store, limit your use of this default deployment to very small-scale, single-server test deployments—in multi-server deployments with load balancing, the active-active topology used by multiple embedded configuration stores can lead to write collisions.

Important

Do not use this deployment in production environments.

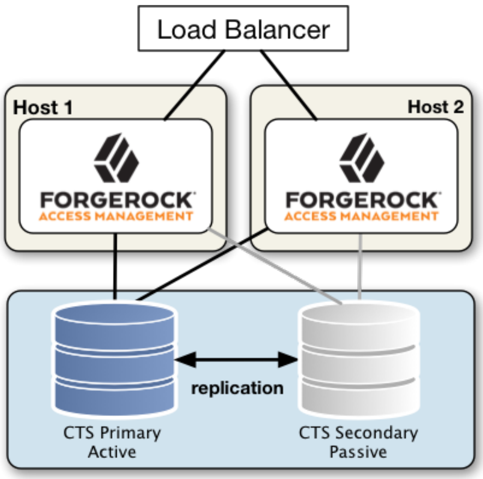
CTS Active-Passive Deployment

An active-passive CTS deployment consists of an active DS instance that receives all requests from the AM servers, and one or more passive DS instances on standby. If the active instance becomes unreachable, one of the passive instances takes over and continues to process transactions. Active-passive clusters provide flexible deployment options with AM and are best suited for small to medium-sized deployments.

The following example shows an active-passive DS cluster configured as CTS token stores. The active DS instances use multimaster replication across the CTS cluster to ensure data is up-to-date.

Prioritize the connections to the primary directory server instance on the AM console and designate additional DS instances for failover using the Connection String(s) property in the CTS configuration. This property allows you to configure multiple DS servers for your CTS token stores without a load balancer.

A CTS Active-Passive Deployment



Important

Do not deploy CTS token stores behind a load balancer using a random or round-robin algorithm.

When AM writes to a directory server in the external CTS store, directory server replication pushes the write to other directory servers in the same replication group. Under load, operations in an AM server can happen more quickly than the network can push replication updates. Therefore, balancing the LDAP traffic from AM to the CTS store using a random or round-robin algorithm leads to errors where a read operation arrives at a replica before the expected write operation can cross the network.

If you need to deploy CTS behind a load balancer, make sure that it is configured for session stickiness.

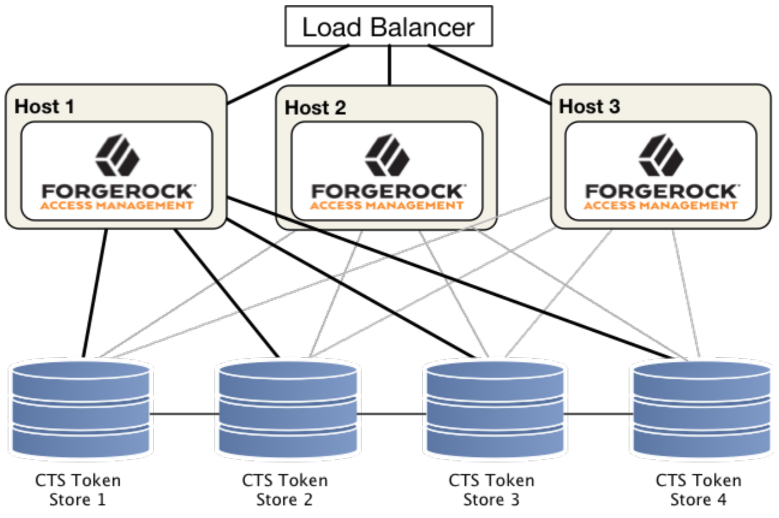
For more information on the Connection String(s) property, see "External Store Configuration" in the *Reference*.

CTS Affinity Deployment

In an *affinity* deployment, AM balances LDAP requests across one or more directory servers. AM always routes LDAP requests for a specific CTS token to the same directory server. This prevents attempts to read a token that has been written to another directory server, but not replicated yet. Affinity deployments are well suited for deployments with many AM servers.

Use AM's Connection String(s) property on the AM console to configure server affinity without a load balancer. For more information on the Connection String(s) property, see "External Store Configuration" in the *Reference*.

A CTS Affinity Deployment



For more information on CTS affinity deployments, see Best practice for using Core Token Service (CTS) Affinity based load balancing in AM (All versions) and OpenAM 13.5.1 in the *ForgeRock Knowledge Base*.

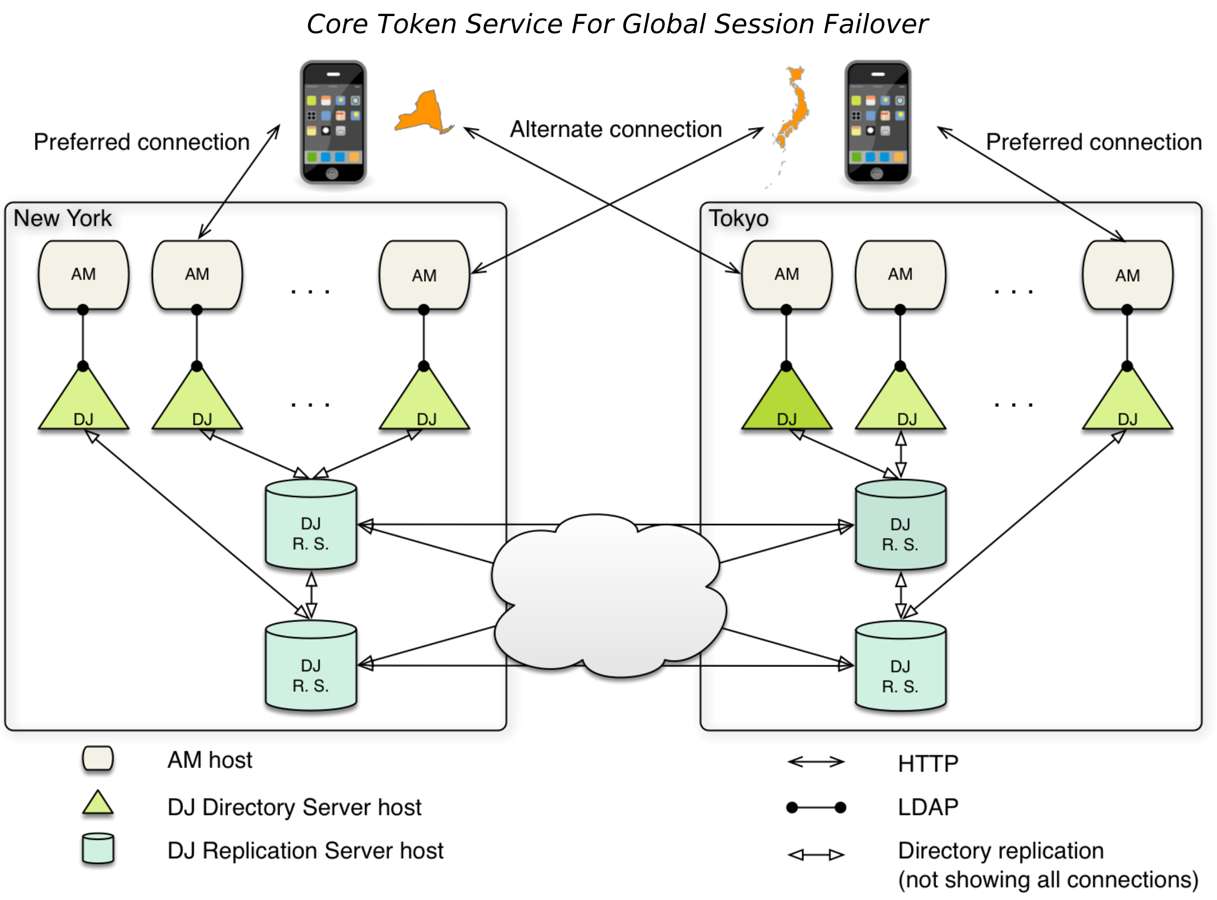
You can also employ a strategy using a DS proxy to distribute data across multiple DS *shards*.

For more information and an example of distributing CTS deployments using DS proxies, see Scaling Out Using Data Distribution in the *Directory Services 6.5 Deployment Guide*.

CTS Site Deployment

CTS supports uninterrupted session availability in deployments with multiple sites if all sites use the same global underlying CTS store replicated across all sites. If an entire site fails or becomes unavailable, AM servers in another site can detect the failure of the site's load balancer and attempt to use sessions from the global Core Token Service.

In the event of a failure, client applications can connect to an AM server in an active data center as shown in "Core Token Service For Global Session Failover":



For more information on CTS for global session high availability with DS server, see the DS documentation on *Managing Data Replication*.

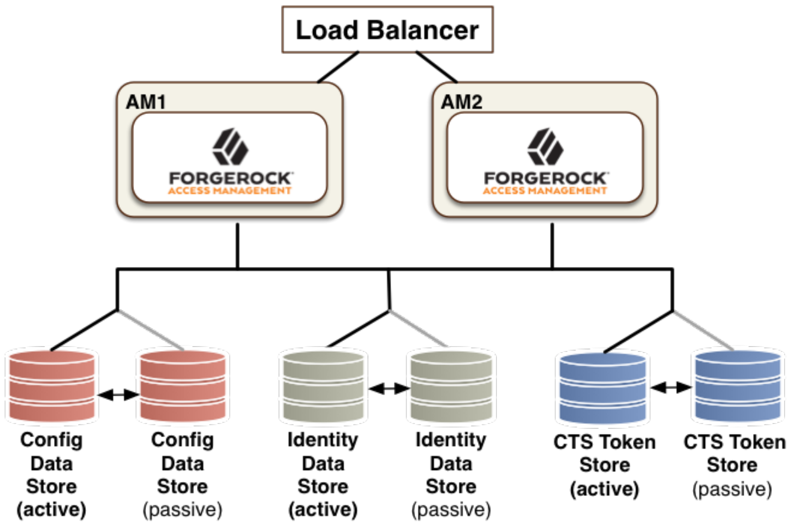
General Recommendations for CTS Configuration

CTS helps your deployment avoid single points of failure (SPOF). To reduce the impact of any given failure, consider the following recommendations:

- **Configure External CTS Stores for High Volumes**. If you require a higher-level performance threshold, you may want to move the CTS token storage to one or more dedicated systems, as CTS generally causes much more replication traffic than less volatile configuration data. The CTS token store is the primary source for session tokens and will experience both high read and write activity depending on session usage. Dedicated external CTS stores provide an extra level of control over the amount of global replication that is occurring.

- **Isolate the Different Stores**. CTS entries are large, around 5KB, but are short-lived, whereas configuration data is static and long-lived. User entries are more dynamic than configuration data but much less volatile than CTS data. Therefore, isolating the user, configuration, and CTS data from AM into separate stores allow for different tuning and storage settings per token store type.

Isolate the Datastores


Load Balancer
AM1
FORGEROCK ACCESS MANAGEMENT
AM2
FORGEROCK ACCESS MANAGEMENT
Config Data Store (active)
Config Data Store (passive)
Identity Data Store (active)
Identity Data Store (passive)
CTS Token Store (active)
CTS Token Store (passive)

- **Properly Tune Your DS Servers**. To improve performance, ensure that you have properly-sized directory servers for your external CTS stores. In addition, you can enable token compression as discussed in "Managing CTS Tokens". When enabled, token compression reduces load requirements on the network connection between token stores in exchange for processing time-compressing tokens.

- **Consider Dedicated Replication Servers**. Once configured, the DS server replicates CTS data transmitted from AM servers to connected DS servers. The amount of replication traffic can be significant, especially if replication proceeds over a WAN. You can limit this replication traffic by separating DS instances into directory and replication servers as seen in "Core Token Service For Global Session Failover". For more information on how this is done with DS, see the DS documentation on *Standalone Replication Servers*.

Configuring CTS in AM

This section explains how to set up the CTS repository in an AM instance using the AM console.

Preparing CTS Stores

The Default Configuration option installs AM with an embedded DS server that stores both configuration and CTS data.

The default option is suitable for AM evaluation purposes and small-scale environments.

In general, CTS causes more volatile replication traffic due to the possibility of short-lived entries compared to regular configuration data. To handle the data volatility, you can configure AM to use the embedded directory server as a dedicated configuration store, while using a separate external DS server instance as a CTS store. This type of deployment is useful if you have multiple AM instances sharing an external CTS store, for example over a WAN.

Installing and Configuring Directory Services for CTS Data

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

To Install and Configure Directory Services for CTS Data

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

Using a setup profile will create the backend, schema, bind user, and indexes required for use with CTS data.

1. To install DS using a setup profile, follow the steps in "To Use DS for AM Core Token Service (CTS) Data" in the *Directory Services 6.5 Installation Guide*.

2. Proceed to configuring the CTS store in AM. See "Configuring CTS in the AM Administration Console".

   The bind account to use when configuring the CTS store in AM is `uid=openam_cts,ou=admins,ou=famrecords,ou=openam-session,ou=tokens`.

Configuring CTS in the AM Administration Console

The section assumes that you have successfully prepared the external DS for CTS, and deployed two AM instances in a site. If you have not completed these steps, see "Preparing CTS Stores" and "To Configure Site Load Balancing". It is also assumed that both AM instances communicate with the single CTS instance, which has an FQDN of `cts.example.com` and is running on port 1389.

To Configure CTS

Important

If AM cannot access the CTS token store, you will be unable to log in to the AM console.

Back up your deployment before making any changes to your CTS token store configuration.

Perform the following steps to configure an external CTS token store:

1. Open the AM console and navigate to Configure > Server Defaults, and then click CTS.
2. On the CTS Token Store tab, configure the parameters as follows:

CTS Token Store Parameters

Parameter	**Value**	**Notes**
Store Mode	`External Token Store`	
Root Suffix	`ou=famrecords,ou=openam-session, ou=tokens`	
Max Connections	`17`	For production, this value needs to be tuned. Consider 2^n+1, where n=4, 5, 6, and so on. For example, try setting this to 17, 33, 65, and test performance under load.

3. On the External Store Configuration tab, configure the parameters as follows:

External Store Configuration Parameters

Parameter	**Value**	**Notes**
SSL/TLS Enabled	True	When connecting to a DS server in production mode, enable secure connections. When you enable SSL/TLS, make sure the AM server can trust the DS server certificate and that the certificate matches the CTS store FQDN.
Connection String(s)	`opendj.example.com:1389`	
Login ID	`uid=openam_cts,ou=admins, ou=famrecords,ou=openam-session, ou=tokens`	
Password	*strngB!ndPw@rd!*	
Heartbeat	`10`	For production, this value needs to be tuned.

4. Click Save Changes.
5. Restart AM or the web container where it runs for the changes to take effect.

Testing Session High Availability

To test session high availability, use two browsers: Chrome and Firefox. You can use any two browser types, or run the browsers in incognito mode. You can also view tokens using an LDAP browser.

To Test Session High Availability

1. In Chrome, log in to the second AM instance with the `amadmin` user, select the realm, and then click on `sessions`.
2. In Firefox, log in to the first AM instance with a test user.
3. In Chrome, verify that the test user exists in the first AM instance's session list and not in the second instance.
4. Shut down the first AM instance.
5. In Firefox, rewrite the URL to point to the second AM instance. If successful, the browser should not prompt for login.
6. Confirm the session is still available. In Chrome, list the sessions on the second instance, the test user's session should be present.
7. Restart the first AM instance to complete the testing.

CTS Backups and Directory Services Replication Purge Delay

Replication is the process of copying updates between directory servers to help all servers converge to identical copies of directory, token, session, SAML v2.0, and OAuth 2.0 data. DS uses advanced data replication methods to ensure that directory services remain available in the event of a server crash or network interruption.

The historical information needed to resolve the latest changes is periodically purged to prevent it from becoming an unmanageable size. The age at which the information is purged is known as the `replication-purge-delay`.

With CTS, the default `replication-purge-delay` for DS is 3 days. Unless you have configured a separate DS server for CTS data, you may have to balance the needs for backups, the requirements for replication, disk space, and different useful lifetimes for CTS tokens and other DS data. Adjustments may be required. One way to set a new period for `replication-purge-delay` of *n* hours is with the following command:

```
$ ./dsconfig \
 set-replication-server-prop \
 --port 4444 \
 --hostname opendj-cts.example.org \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --provider-name "Multimaster Synchronization" \
 --set replication-purge-delay:n \
 --no-prompt \
 --trustStorePath /path/to/truststore
```

At this point, you need to understand whether CTS data backups are important in your deployment. Session, SAML v2.0, and OAuth 2.0 token data is often short-lived. In some deployments, the worst-case scenario is that users have to log in again.

If CTS data backups are important in your deployment, note that DS backups that are older than the `replication-purge-delay` are useless and must be discarded. You can use the DS **backup** to schedule backups. For example, the following command uses `crontab` format to configure daily backups for a hypothetical Base DN of `ctsData` at x minutes after every hour:

```
$ ./backup \
 --port 4444 \
 --bindDN "cn="Directory Manager" \
 --bindPassword password \
 --backendID ctsData \
 --backupDirectory /path/to/opendj/backup \
 --recurringTask "x * * * *" \
 --completionNotify backupadmin@example.com \
 --errorNotify backupadmin@example.com
```

If you adjust the time periods associated with `replication-purge-delay` and backups, you need to backup more frequently so that the change log records required to restore date are not lost.

Managing CTS Tokens

You can configure AM to encrypt or compress CTS tokens as they are stored in the token store. The following properties, disabled by default, are associated with token encryption and compression:

`com.sun.identity.session.repository.enableEncryption`

Supports encryption of CTS tokens. Default: `false`.

`com.sun.identity.session.repository.enableCompression`

Enables GZip-based compression of CTS tokens. Default: `false`.

`com.sun.identity.session.repository.enableAttributeCompression`

Supports compression over and above the GZip-based compression of CTS tokens. Default: `false`.

Important

Compression can undermine the security of encryption. You should evaluate this threat depending on your use cases before enabling compression and encryption together.

To Configure AM to Encrypt and / or Compress CTS Tokens for Storage

Warning

When encryption or compression properties are changed, all previous tokens in the LDAP store will be unreadable; thus, invalidating any user's sessions. As a result, the user will be required to log in again.

1. Navigate to Configure > Server Defaults > Advanced.
2. Find the property you want to enable in the Property Name column.
3. Replace the `false` value with `true` in the Property Value column.
4. Save your changes.
5. Enable the same property on every AM instance within the site. Failure to do so may cause unexpected issues storing and reading tokens across the environment.
6. Restart the AM servers for the changes to take effect.

Tip

Configuring the CTS to encrypt and store tokens incurs a performance penalty for AM. If you need to encrypt the stored tokens in your environment, consider configuring the CTS token store DS instance to encrypt the data instead. For more information about encrypting a DS instance, see the *ForgeRock Directory Services Administration Guide*.

Configuring the CTS Reaper

Tokens in the CTS store have a time to live after which they must be pruned. AM can manage expired tokens by either using its own reaper (the default) or by delegating the task to DS. Both AM and DS are equally adept at managing expired tokens. However, choosing DS to manage token expiration in the place of AM's reaper frees resources in the AM servers than can then be used for policy or authorization requests.

Review the following list for more information about the different configurations:

AM CTS Reaper (Default)

When an AM server modifies a token in the CTS token store, it also takes responsibility to manage it when it expires. To determine which tokens have expired, each AM server maintains a local cache of which tokens to delete and when. This reduces the number of relatively slow queries to the CTS store.

Use of the local reaper cache means fewer searches to the CTS store to determine expired tokens to delete, which improves overall cluster performance. A search of the CTS store for expired tokens is still performed as a fail safe, to ensure expired tokens are not missed when a server in the cluster goes down.

The AM CTS reaper is enabled by default. However, if you have configured the DS expiration and deletion feature, and you need to enable the AM CTS reaper again, see "To Enable the AM Reaper".

DS Entry Expiration and Deletion Feature

When the AM CTS reaper is disabled, AM relies on DS capabilities to delete expired tokens.

By default, DS does not reap its contents, so you need to configure the entry expiration and deletion feature manually. This feature uses an ordering index to find those tokens that have reached their time to live and expires (deletes) them.

Before you decide to configure DS to manage CTS token expiration, consider the following points:

- The CTS token store must be in DS version 6.0 or later.
- Deletion of entries is not replicated across DS servers. Therefore, you must ensure all the CTS store replicas are configured in the same way. For more information, see Automating Entry Expiration and Deletion in the *ForgeRock Directory Services Administration Guide*.
- Completely disabling the AM reaper impacts session-related functionality, such as sending notifications about session expiration or timeouts to agents.

  Tip

  A common implementation to manage CTS tokens is to configure AM to reap session tokens only, while using DS capabilities to manage the non-session tokens. This configuration ensures your environment still can make use of all session functionality, while benefiting from DS's capabilities as well.

  You can configure the AM reaper to manage different subsets of tokens other than session tokens, too, if your environment requires it.

  The following is a list of session functionality that is not available when the AM reaper is completely disabled:

  - Implementations of `org.forgerock.openam.cts.reaper.TokenDeletionStrategy`, which are responsible for deleting expiring tokens. Any custom logic for specific token types that is not related to simply deleting tokens no longer work.
  - Implementations of `org.forgerock.openam.cts.continuous.watching.ContinuousListener` and `org.forgerock.openam.cts.continuous.watching.ContinuousWatcher` no longer receive deletion notifications in case of session timeout (active logout is not impacted).

- Code built on top of `com.iplanet.dpro.session.service.SessionEventListener` no longer receives `IDLE_TIMEOUT` or `MAX_TIMEOUT` events. This affects:
  - Session timeout monitoring
  - Session timeout auditing
  - Session timeout logging
  - Session timeout notifications for agents (PLL, WebSockets)
  - Timeout handlers configured via `openam-session-timeout-handler-list`
  - Session web hooks registered during authentication
- Code built on top of `com.iplanet.dpro.session.watchers.listeners.SessionDeletionListener` no longer receives notifications when the session idle or maximum lifetime is exceeded (common usage includes caches).
- Listeners built on top of `com.sun.identity.plugin.session.SessionListener` no longer receive notifications on active logout, nor when the maximum session time is reached (idle timeout is ignored).

If your environment does not require the session capabilities listed above, see "To Enable DS to Manage the Tokens in the CTS Store (Impacted Session Capabilities Not Required) ".

If your environment requires the session capabilities above, see "To Enable DS to Manage the Tokens in the CTS Store (Impacted Session Capabilities Required)".

To Enable DS to Manage the Tokens in the CTS Store (Impacted Session Capabilities Not Required)

This procedure assumes you have ensured that your environment does not require the session capabilities impacted by stopping the AM CTS reaper. Perform the steps in this procedure to configure DS to manage expiration and deletion of all CTS tokens in your environment:

1. Perform one of the following options depending on whether you are installing a new Directory Services instance for CTS, or are modifying an existing instance:

   - If you are creating a *new instance* of Directory Services for use with CTS, perform the following step:

     - Use a setup profile, and set the `tokenExpirationPolicy` property to `ds`.

       For more information, see "DS for AM CTS With Token Expiration" in the *Directory Services Installation Guide*.

- If you are modifying an *existing instance* of Directory Services for use with CTS, perform the following step:
    - Configure the DS entry expiration and deletion feature for the `coreTokenExpirationDate` attribute. Note that the attribute is already indexed, but requires the TTL-related properties to be enabled. For example:

      ```
      $ /path/to/dj/bin/dsconfig set-backend-index-prop \
       --hostname cts.example.com \
       --port 4444 \
       --bindDN "cn=Directory Manager" \
       --bindPassword 'str0ngEx4mplePa55word' \
       --backend-name ctsStore \
       --index-name coreTokenExpirationDate \
       --set ttl-enabled:true \
       --set ttl-age:10s \
       --trustAll
      ```

      Deletion of entries is not replicated across DS servers. Therefore, you must ensure all the CTS store replicas are configured in the same way. For more information, see Automating Entry Expiration and Deletion in the *ForgeRock Directory Services Administration Guide*.

2. Disable the AM CTS reaper by navigating to Configure > Server Defaults > Advanced and setting the `org.forgerock.services.cts.store.reaper.enabled` property to `false` on all the AM servers on your deployment.

   The changes are effective immediately.

   For more information about these advanced server properties, see "Advanced Properties" in the *Reference*.

To Enable DS to Manage the Tokens in the CTS Store (Impacted Session Capabilities Required)

This procedure assumes that your environment requires the session capabilities that would be impacted by stopping the AM CTS reaper. Perform the steps in this procedure to configure AM to reap `SESSION` tokens only, while using DS capabilities to manage the rest:

1. Perform one of the following options depending on whether you are installing a new Directory Services instance for CTS, or are modifying an existing instance:

    - If you are creating a *new instance* of Directory Services for use with CTS, perform the following step:
        - Use a setup profile, and set the `tokenExpirationPolicy` property to `am-sessions-only`.

          For more information, see "DS for AM CTS With Token Expiration (Session Capabilities Required)" in the *Directory Services Installation Guide*.

- If you are modifying an *existing instance* of Directory Services for use with CTS, perform the following steps:

  a. (Optional) If you have existing tokens in the CTS store, you must copy the value of the `coreTokenExpirationDate` attribute to the `coreTokenTtlDate` for all the existing tokens **except** for `SESSION` tokens.

     AM will manage the `SESSION` tokens separately.

     If you do not perform this step, neither the AM reaper nor the DS expiration feature will delete these tokens and they will remain untouched in the CTS store until manually removed.

  b. Create an ordering index for the `coreTokenTtlDate` attribute. For example:

     ```
     $ /path/to/dj/dsconfig create-backend-index \
      --hostname cts.example.com \
      --port 4444 \
      --bindDN "cn=Directory Manager" --bindPassword 'str0ngEx4mplePa55word' \
      --backend-name ctsStore \
      --index-name coreTokenTtlDate \
      --set index-type:ordering \
      --trustAll \
      --no-prompt
     ```

     Deletion of entries is not replicated across DS servers. Therefore, you must ensure all the CTS store replicas are configured in the same way.

  c. Rebuild the new index using the **rebuild-index** command. For example:

     ```
     $ /path/to/dj/rebuild-index \
      --hostname cts.example.com \
      --port 4444 \
      --bindDN "cn=Directory Manager" \
      --bindPassword 'str0ngEx4mplePa55word' \
      --baseDN "dc=cts,dc=example,dc=com" \
      --index coreTokenTtlDate \
      --trustAll
     ```

     Deletion of entries is not replicated across DS servers. Therefore, you must ensure all the CTS store replicas are configured in the same way.

  d. Configure DS's entry expiration and deletion feature in the CTS store for the `coreTokenTtlDate` index created in the previous steps. For example:

```
$ /path/to/dj/bin/dsconfig set-backend-index-prop \
 --hostname cts.example.com \
 --port 4444 \
 --bindDN "cn=Directory Manager" \
 --bindPassword 'str0ngEx4mplePa55word' \
 --index-name coreTokenTtlDate \
 --backend-name ctsStore \
 --set ttl-enabled:true \
 --set ttl-age:10s \
 --trustAll
```

Deletion of entries is not replicated across DS servers. Therefore, you must ensure all the CTS store replicas are configured in the same way.

For more information, see Automating Entry Expiration and Deletion in the *ForgeRock Directory Services Administration Guide*.

2. In the AM administration console, navigate to Configure > Server Defaults > Advanced and configure the following advanced server properties in all of the AM servers sharing the CTS store cluster:

   - Set `org.forgerock.services.cts.store.ttlsupport.enabled` to `true`.
   - Set `org.forgerock.services.cts.store.ttlsupport.exclusionlist` to `SESSION`.
   - Set `org.forgerock.services.cts.store.reaper.enabled` to `true`.

   The changes are effective immediately.

   For more information about these advanced server properties, see "Advanced Properties" in the *Reference*.

To Enable the AM Reaper

This procedure assumes you enabled the DS expiration and deletion feature and you want to disable it and enable the AM reaper again:

1. (Optional) If you enabled the DS expiration and deletion feature, following the steps in "To Enable DS to Manage the Tokens in the CTS Store (Impacted Session Capabilities Not Required) ", perform the following steps to enable the AM CTS reaper for all tokens:

   a. Disable the DS entry expiration and deletion feature for the `coreTokenExpirationDate` index. Note that you should not delete the index. For example:

```
$ /path/to/dj/bin/dsconfig set-backend-index-prop \
--hostname cts.example.com \
--port 4444 \
--bindDN "cn=Directory Manager" \
--bindPassword 'str0ngEx4mplePa55word' \
--backend-name ctsStore \
--index-name coreTokenExpirationDate \
 --set ttl-enabled:false
```

Ensure that all the CTS store replicas are configured in the same way.

b. Enable the AM CTS reaper by by navigating to Configure > Server Defaults > Advanced and setting the `org.forgerock.services.cts.store.reaper.enabled` property to `true` on all the AM servers on your deployment.

2. (Optional) If you enabled the DS expiration and deletion feature following the steps in "To Enable DS to Manage the Tokens in the CTS Store (Impacted Session Capabilities Required)", perform the following steps to enable the AM CTS reaper for all tokens:

   a. Disable the DS entry expiration and deletion feature for the `coreTokenTtlDate` index. For example:

```
$ /path/to/dj/bin/dsconfig set-backend-index-prop \
--hostname cts.example.com \
--port 4444 \
--bindDN "cn=Directory Manager" \
--bindPassword 'str0ngEx4mplePa55word' \
--index-name coreTokenTtlDate \
--backend-name ctsStore \
--set ttl-enabled:false
```

   Ensure that all the CTS store replicas are configured in the same way.

   b. Delete the index created for the `coreTokenTtlDate` attribute. For example:

```
$ /path/to/dj/dsconfig delete-backend-index \
--hostname cts.example.com \
--port 4444 \
--bindDN "cn=Directory Manager"
--bindPassword 'str0ngEx4mplePa55word' \
--backend-name ctsStore \
--index-name coreTokenTtlDate \
--trustAll \
--no-prompt
```

   Ensure that all the CTS store replicas are configured in the same way.

   c. Navigate to Configure > Server Defaults > Advanced and configure the following advanced server properties in all of the AM servers sharing the CTS store cluster:

   - Set `org.forgerock.services.cts.store.ttlsupport.enabled` to `false`.
   - Remove the `org.forgerock.services.cts.store.ttlsupport.exclusionlist` property from the configuration.
   - Set `org.forgerock.services.cts.store.reaper.enabled` to `true`.

   The changes are effective immediately.

   For more information about these advanced server properties, see "Advanced Properties" in the *Reference*.

CTS Tuning Considerations

There are several CTS tuning considerations that you can make for the efficient processing of your CTS token store: reaper cache size, queue size and timeout, and virtual attributes.

Reaper Cache Size

The size of the AM reaper cache is controlled by the `org.forgerock.services.cts.reaper.cache.size` advanced property. The default size is `500000` tokens.

If an AM server is under sustained heavy load, the reaper cache may reach capacity, causing degraded performance due to the additional slower searches of the CTS store. If the reaper cache is full, messages are logged in the `Session` debug log, such as the following:

```
The CTS token reaper cache is full. This will result in degraded performance.
You should increase the cache size by setting the advanced server property
'org.forgerock.services.cts.reaper.cache.size' to a number higher than 500000.
```

If this debug message appears frequently in the debug logs, increase the value of the `org.forgerock.services.cts.reaper.cache.size` property. To alter the value, in the AM console, navigate to Configure > Server Defaults > Advanced, and add the property and increased value to the list.

Increasing the size of the reaper cache causes higher memory usage on the AM server. If a cache of the default size of 500000 entries is nearly full, the server memory used could be up to approximately 100 megabytes.

Note

Tune the AM reaper cache size only if the `org.forgerock.services.cts.store.reaper.enabled` advanced server property is set to `true`.

Queue Size and Timeout

One tuning consideration is to manage the CTS queue size and timeout for efficient throughput. AM makes CTS requests from the following components:

- AM CTS-based sessions and authentication sessions
- OAuth 2.0
- UMA
- Session blacklist (if enabled for client-based sessions)
- Authentication session whitelist (if enabled for client-based sessions)
- SAML v2.0 (if enabled for Security Token Service token validation and cancellation)
- OAuth 2.0 CTS-based tokens and client-based token blacklist
- Push notification during authentication
- Cluster-wide notification

Every create, update, and delete requests to CTS are placed into an asynchronous buffer before being handled by an asynchronous processor. This ensures that callers performing write operations can continue without waiting for CTS to complete processing.

Once the queue is full, all new operations are "blocked" before being placed in the queue. Once the queue is frees up, the caller can continue as normal.

CTS is designed to automatically throttle throughput when the buffer fills up with requests. Therefore, if you require a balance between performance versus system memory, AM provides two properties that can be used to tune CTS, queue size and queue timeout.

`org.forgerock.services.cts.async.queue.size`

> Default size: 5000. Determines the amount of request operations that can be buffered before the queue size becomes full, after which the caller will be required to wait for the buffered requests to complete processing. All CRUDQ operations are converted to tasks, which are placed on the queue, ensuring that operations happen in the correct sequence.

`org.forgerock.services.cts.async.queue.timeout`

> Default timeout is 120 seconds. Determines the length of time a caller will wait when the buffer is full. If the timeout expires, the caller receives an error. The timeout property is used in any system configuration where the LDAP server throughput is considerably slower than the AM server, which can result in blocked requests as the backlog increases.

To set the queue size and timeout properties in the AM console, navigate to Configure > Server Defaults > Advanced, enter the key name and value, and then click Add.

For additional information on tuning CTS, see "Tuning CTS Store LDAP Connections" in the *Setup and Maintenance Guide*.

Page Size

You can alter the number of records that are returned in one page when searching the CTS store for expired records.

AM uses the advanced property `org.forgerock.services.cts.reaper.search.pageSize` to specify the number of records to return in each page.

> **Note**
>
> The AM property overrides the DS property for controlling page sizes, `ds-rlim-size-limit`.
>
> For more information on the DS property, see Limiting Search Resources in the *DS Maintenance Guide*.

If there are likely to be a large number of CTS tokens that are due to expire at approximately the same time, you could increase the search page size from the default of `1000`, in order to affect more expired tokens at a time.

For more information on the properties available for tuning CTS search page size, see "Advanced Properties" in the *Reference*.

Virtual Attributes

DS supports a number of virtual attributes, which dynamically generate entry values that are not persisted in the token store. By default, the following DS virtual attributes are enabled at install:

- collective-attribute-subentries
- entity-tag
- entry-dn
- entry-uuid
- governing-structure-rule
- has-subordinates
- is-member-of
- num-subordinates
- password-expiration-time
- password-policy-subentry
- structural-object-class
- subschema-subentry

To improve CTS throughput, you can disable your virtual attributes using the **dsconfig** command, *except* for the Entity Tag virtual attribute, which must remain enabled else it will lead to an inoperable server.

To disable a virtual attribute, use the **dsconfig** command on DS to disable, for example, the Collective Attribute Subentries virtual attribute:

```
./dsconfig set-virtual-attribute-prop \
 --name=collective-attribute-subentries-virtual-attribute \
 --set enabled: false \
 --hostname localhost \
 --port 4444 \
 --bindDN "cn=Directory manager" \
 --bindPassword password \
 --no-prompt
```

Important

The DS `entity-tag-virtual-attribute` is required and should never be disabled.

OAuth 2.0 CTS Storage Scheme

AM 6.5 introduced a new scheme for storing OAuth 2.0 tokens in the CTS store, called the *grant-set* scheme.

The `grant-set` scheme groups multiple authorizations for a given OAuth 2.0 client and resource owner pair and stores them in a single CTS `OAUTH2_GRANT_SET` entry. This implementation reduces the size and quantity of entries stored, as well as the number of calls required to perform OAuth 2.0 operations.

The `one-to-one` scheme stores the state of multiple authorizations for a given OAuth 2.0 client and resource owner pair across multiple entries, and is more resource intensive. You should upgrade to the `grant-set` scheme once all the servers on your environment have been upgraded to AM 6.5 or later.

The `grant-set` scheme is backwards-compatible with existing entries stored in the CTS store. Therefore, any access or refresh token issued before configuring the `grant-set` scheme is still valid. Existing tokens will be retained in their original form until the refresh token expires or it is actively revoked.

Users will not notice any change in the tokens they receive, and there is no change to the OAuth 2.0 endpoints.

To enable the `grant-set` scheme, navigate to Configure > Global Services > OAuth2 Provider > Global Attributes and set the CTS Storage Scheme drop-down to Grant-Set Storage Scheme. Then, save your changes.

New OAuth 2.0 tokens stored in the CTS after the change will use the new scheme automatically.

Chapter 5
Securing Installations

This chapter identifies best practices for securing your AM installation.

Avoiding Obvious Defaults

AM includes default settings to make it easier for you to evaluate the software. Avoid these default settings in production deployments:

- When connecting to LDAP, bind with a specific administrative account rather than a root DN account, if possible.
- Change the default `iPlanetDirectoryPro` cookie name both in AM (`com.iplanet.am.cookie.name`) and in your agent profiles (`com.sun.identity.agents.config.cookie.name`).
- When installing AM, do not use `/openam` or `/opensso` as the deployment URI.
- Create an administrator in the Top Level Realm with a different ID than the default `amadmin`.
- Create specific administrator users to track better who makes configuration changes.
- Remove the demo user account. For example, if you configure the embedded DS server as a configuration and CTS store, the default demo user account gets created during the installation process. You should remove the user using the AM console under Realms > Top Level Realm > Identities. Select the user you want to remove, and then select Delete.
- Deactivate the Anonymous User.

  The anonymous user is enabled by default. To harden security, deactivate the anonymous user, unless anonymous access is specifically required in your deployment. See How do I deactivate the default anonymous user in AM.
- Disable module based authentication for all AM realms. Module based authentication lets users authenticate using the `module=module-name` login parameter. To disable module based authentication for a realm, select the realm in the AM console, then select Authentication > Settings > Security and clear the Module Based Authentication check box.

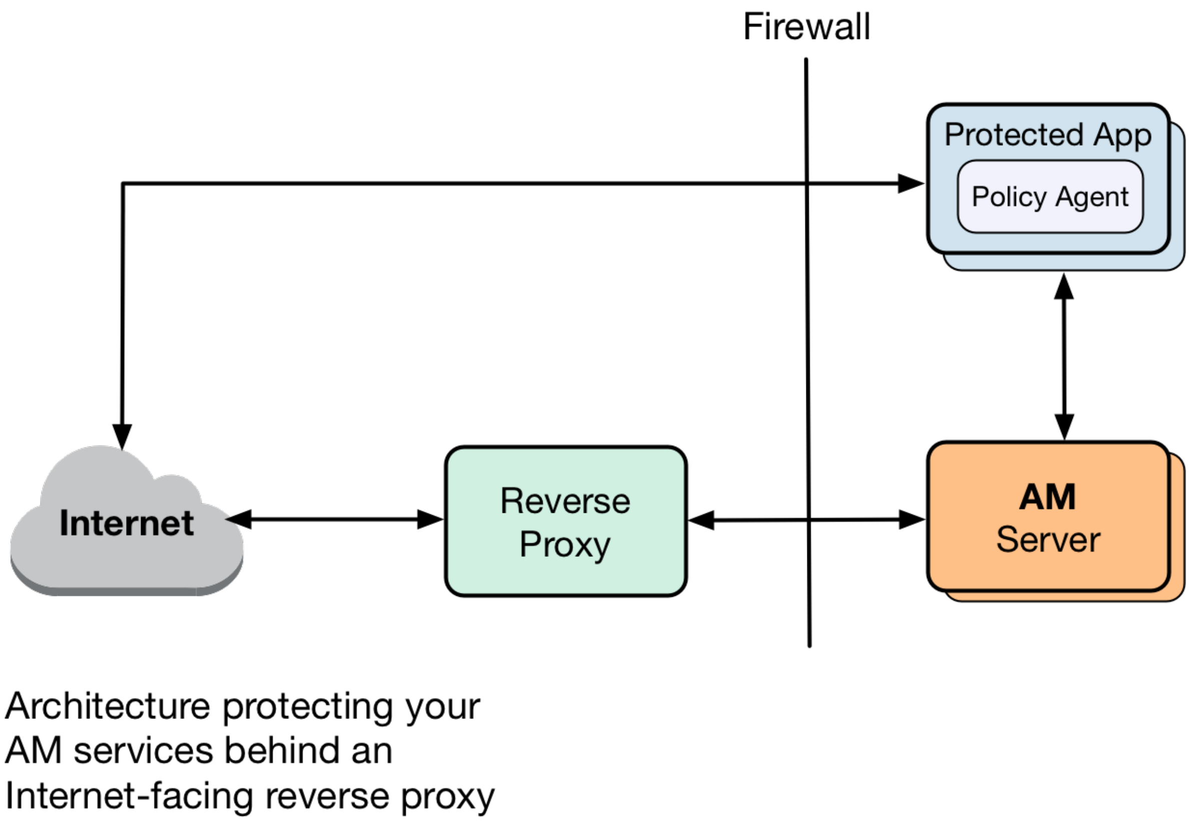
Protecting Network Access

Anytime users interact with a web service, there are risks. With AM, you can reduce those risks by limiting what is exposed through the firewall using the following strategy:

- Use a reverse proxy in front of AM to allow access only to the necessary URLs. A reverse proxy exposes only those endpoints needed for an application. For example, if you need to expose the OAuth2/OpenID Connect endpoints and REST interface, then you should implement a reverse proxy.

  The following figure shows the recommended architecture with a reverse proxy.

Exposing Only a Reverse Proxy to the Internet


Firewall
Protected App
Policy Agent
Internet
Reverse Proxy
AM Server
Architecture protecting your AM services behind an Internet-facing reverse proxy

- If possible in your deployment, control access to the AM console by network address, such that administrators can only connect from well-known systems and networks.
- Restrict access to URIs that you do not use, and prevent internal endpoints, such as `/sessionservice` from being reachable over the Internet.

  For a full list of endpoints, see "*Service Endpoints*" in the *Reference*.

Restricting Endpoint Caching

To help protect against snooping of data on public computers, you should configure AM instances to disallow browsers to cache user-submitted information.

You can prevent a browser from caching data by including the following headers in responses from endpoints:

```
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: 0
```

It is also recommended to set `Cache-Control: private` on all endpoints that contain `/json/` in their path, but not the following endpoints:

- `/serverinfo/*`
- `/serverinfo/version`
- `/serverinfo/cookieDomains`

You can apply these headers to the specified endpoints by configuring your reverse proxy or gateway, or you can use the AM `SetHeaderFilter` feature in your web containers' `web.xml` file, as follows:

To Restrict Endpoint Caching by using SetHeaderFilter

1. Open the deployment descriptor file `web.xml` in a text editor. The location of the file depends on your web application container, for example in Tomcat it might be located at: `/path/to/tomcat/webapps/openam/WEB-INF/web.xml`.
2. In the deployment descriptor file, add the following filter and filter mapping, which applies `Cache-Control: private` to the correct endpoints:

```
<filter>
    <filter-name>CachePrivate</filter-name>
    <filter-class>org.forgerock.openam.headers.SetHeadersFilter</filter-class>
    <init-param>
        <param-name>Cache-Control</param-name>
        <param-value>private</param-value>
    </init-param>
    <init-param>
        <param-name>excludes</param-name>
        <param-value>/serverinfo/*,/serverinfo/version,/serverinfo/cookieDomains</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>CachePrivate</filter-name>
    <url-pattern>/json/*</url-pattern>
</filter-mapping>
```

3. Restart your web container for the changes to take effect.

Securing Administration

Create realms for your organization(s) and separate administrative users from end users. For instructions, see "*Setting Up Realms*" in the *Setup and Maintenance Guide*.

- To direct relevant users to the correct realms in AM, you can then either:
    - Use the `realm=realm-name` query string parameter.
    - Create fully qualified domain name DNS aliases for the realms.
- When customizing `config/auth/default*/Login.jsp`, make sure that you do not introduce any security vulnerabilities, such as cross-site scripting due to unvalidated input.
- Create an agent profile for each web or Java agent. See "*Setting Up Agent Profiles*" in the *Setup and Maintenance Guide* for instructions.

Securing Communications

Keep communications secure by using encryption, properly configured cookies, and request and response signatures:

- Protect network traffic by using HTTPS and LDAPS where possible.
- When using HTTPS, use secure cookies, which are transmitted only over secured connections.

  To configure AM server to use secure cookies, in the AM console, navigate to Configure > Server Defaults > Security. On the Cookie tab, select Secure Cookie, and then click Save Changes.

HttpOnly cookies are meant to be transmitted only over HTTP and HTTPS, and not through non-HTTP methods, such as JavaScript functions.

You can configure the AM server to use `HttpOnly` cookies by navigating to Configure > Server Defaults > Advanced, and setting the `com.sun.identity.cookie.httponly` property's value to `true`. Save your changes.

- Specify an appropriate TLS protocol for communication between AM and one or more of the following:
  - An external AM configuration store
  - An external CTS token store
  - An external user store
  - An Active Directory repository used for AM authentication with the Active Directory module
  - An LDAP repository used for AM authentication with the LDAP module
  - A repository used for AM certificate authentication with the certificate module

  To specify the TLS protocol, use the `-Dorg.forgerock.openam.ldap.secure.protocol.version` JVM setting. For more information, see "Security Settings".
- Where possible, use subdomain cookies, and control subdomains in a specific DNS master.
- Use cookie hijacking protection with restricted tokens, where each web or Java agent uses different SSO tokens for the same user. See "Protecting Against Cookie Hijacking" in the *Authentication and Single Sign-On Guide* for instructions.
- Replace default keys and aliases. See "About the Default Keystores and Secret Stores" in the *Setup and Maintenance Guide*.
- When using SAML v2.0, if the other entities in your circle of trust can handle encryption, then use encryption in addition to signing requests and responses.

About Certificates

Digital signatures are constructed and verified as follows:

- The signer computes a hash of the data to sign, and encrypts the hash using a private key to get the signature.
- The signer then attaches the signature to the data, and sends the message with the recipient.
- To validate the digital signature on the message, the recipient decrypts the signature using the public key certificate that corresponds to the private key of the signer.

- The recipient computes the hash of the data, then checks that the decrypted signature (the decrypted hash) matches the computed hash.

Parties signing requests, responses, or assertions must share the public key certificates for signing keys. The certificates can either be shared in advance and imported into the trusted partners' trust stores, then referenced in the configuration by their trust store aliases, or shared in each signed message.

You should not have to concern yourself with certificates when working with AM. AM's core services and Java agents depend on the certificates installed for use with the web application container in which they run. AM web agents depend on the certificates installed for use with the web server. Each certificate has been signed by a well-known certificate authority (CA), whose certificate is already installed in the Java CA certificates trust store (`$JAVA_HOME/jre/lib/security/cacerts`, default password `changeit`) and in browsers, and so is recognized by other software used without you having to configure anything.

However, you may want to configure AM advanced features such as SAML v2.0, OpenID Connect 1.0, and others, which require certificates and key aliases to be maintained in a keystore whose location is configured in AM.

Using Self-Signed Certificates

You can use either CA or self-signed certificates with AM, although you should have in mind that you will need to configure your applications to trust your self-signed certificates. For more information about installing AM in a secure container with a self-signed certificate, see "To Set Up With HTTPS and Self-Signed Certificates". For more information about sharing self-signed certificates among applications, see "To Share Self-Signed Certificates".

To Set Up With HTTPS and Self-Signed Certificates

The container in which you install AM requires a certificate in order to set up secure connections. Perform the following steps to set up Apache Tomcat 8.0 (Tomcat) with an HTTPS connector, using the Java **keytool** command to create a self-signed key pair:

1. Stop Tomcat.

2. Create a certificate and store it in a new keystore:

```
$ cd /path/to/tomcat/conf/
$ keytool -genkey -alias openam.example.com -storetype JCEKS
-keyalg RSA -keystore keystore.jceks
Enter keystore password:
 What is your first and last name?
 [Unknown]:  openam.example.com
What is the name of your organizational unit?
 [Unknown]:  Eng
What is the name of your organization?
 [Unknown]:  ForgeRock.com
What is the name of your City or Locality?
 [Unknown]:  Grenoble
What is the name of your State or Province?
 [Unknown]:  Isere
What is the two-letter country code for this unit?
 [Unknown]:  FR
Is CN=openam.example.com, OU=Eng, O=ForgeRock.com, L=Grenoble, ST=Isere,
 C=FR correct?
 [no]:  yes
Enter key password for <openam.example.com>
 (RETURN if same as keystore password):
```

3. Uncomment the SSL connector configuration in Tomcat's `conf/server.xml`, and specify your keystore file, type, and password:

```
<!-- Define a SSL HTTP/1.1 Connector on port 8443
This connector uses the JSSE configuration, when using APR, the
connector should be using the OpenSSL style configuration
described in the APR documentation -->
<!--
-->
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
 SSLEnabled="true" maxThreads="150" scheme="https" secure="true"
 keystoreFile="/path/to/tomcat/conf/keystore.jceks"
 keystorePass="changeit"
 keystoreType="JCEKS"
 clientAuth="false" sslProtocol="TLS" />
```

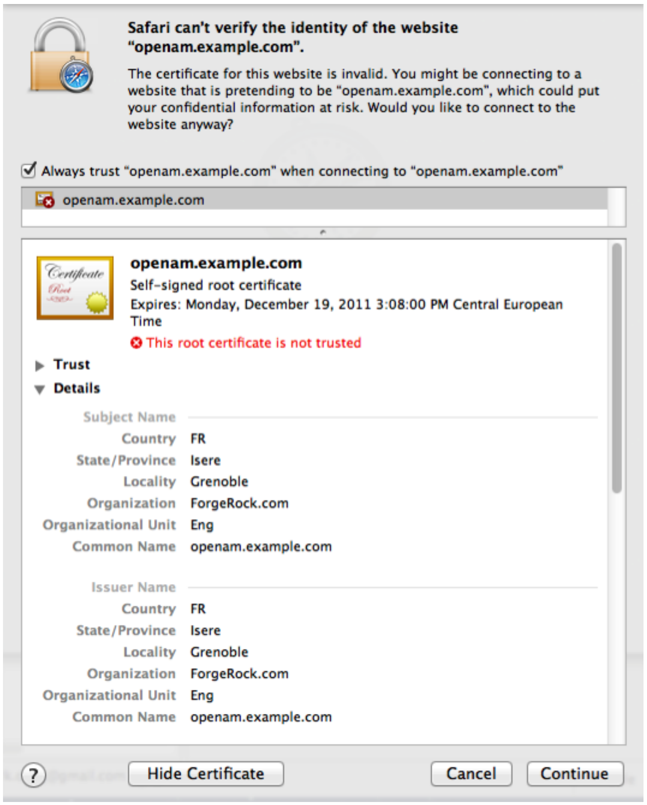
   You may need different settings depending on your configuration and version of Apache Tomcat. See the documentation for your version for more information.

4. Start Tomcat.

5. Verify that you can connect to Tomcat on port 8443 over HTTPS.

   Your browser does not trust the certificate, because the certificate is self-signed and not signed by any of the CAs stored in your browser.

Unknown Certificate



Safari can't verify the identity of the website "openam.example.com".

The certificate for this website is invalid. You might be connecting to a website that is pretending to be "openam.example.com", which could put your confidential information at risk. Would you like to connect to the website anyway?

☑ Always trust "openam.example.com" when connecting to "openam.example.com"

openam.example.com

openam.example.com
Self-signed root certificate
Expires: Monday, December 19, 2011 3:08:00 PM Central European Time
This root certificate is not trusted

▶ Trust
▼ Details

Subject Name	
Country	FR
State/Province	Isere
Locality	Grenoble
Organization	ForgeRock.com
Organizational Unit	Eng
Common Name	openam.example.com
Issuer Name	
Country	FR
State/Province	Isere
Locality	Grenoble
Organization	ForgeRock.com
Organizational Unit	Eng
Common Name	openam.example.com

You recognize the subject and issuer of your certificate, and so can choose to trust the certificate, saving it into your browser's trust store.

6. Deploy and configure AM.

7. To share the self-signed certificate in your container with other applications or servers, see "To Share Self-Signed Certificates".

To Share Self-Signed Certificates

How you configure the containers where AM and your applications run to use self-signed certificates depends on your web application server or web server software. The following basic principles apply:

- First, your container requires its own certificate for setting up secure connections.

- Second, the clients connecting must be able to trust the container's certificate. Generally, this means that clients recognize the container's certificate because they have a copy of the public certificate stored somewhere the client trusts.
- Third, if you use certificate authentication in AM, AM must also be able to find a copy of the client's public certificate to trust the client, most likely by finding a match with the certificate stored in the client profile from the identity repository. How you include client certificates in their identity repository entries depends on your identity repository more than it depends on AM.

Some client applications let you trust certificates blindly. This can be helpful when working in your lab or test environment with self-signed certificates. For example, you might want to use HTTPS with the AM RESTful API without having the client recognize the self-signed server certificate:

```
$ curl \
--header "Accept-API-Version: resource=1.0" \
'https://openam.example.com:8443/openam/identity/authenticate?username=bjensen&password=hifalutin'
{
    curl: (60) Peer certificate cannot be authenticated with known CA certificates
}
```

More details here: http://curl.haxx.se/docs/sslcerts.html

curl performs SSL certificate verification by default, using a "bundle" of Certificate Authority (CA) public keys (CA certs). If the default bundle file isnt adequate, you can specify an alternate file using the --cacert option. If this HTTPS server uses a certificate signed by a CA represented in the bundle, the certificate verification probably failed due to a problem with the certificate (it might be expired, or the name might not match the domain name in the URL). If you'd like to turn off curl's verification of the certificate, use the -k (or --insecure) option.

```
$ curl \
--insecure \
'https://openam.example.com:8443/openam/identity/authenticate?username=bjensen&password=hifalutin'
{
    token.id=AQIC5wM2LY4SfczMax8jegpSiaigB96NOWylLilsd0PUMjY.*AAJTSQACMDE.*
}
```

When you use a self-signed certificate for your container, clients connecting must be able to trust the container certificate. Your browser makes this an easy, but manual process. For other client applications, you must import the certificate into the trust store used by the client. By default, Java applications can use the `$JAVA_HOME/jre/lib/security/cacerts` store. The default password is `changeit`.[1] The steps that follow demonstrate how to import a self-signed certificate into the Java `cacerts` store:

1. Export the certificate from the keystore:

[1]Alternatively, you can specify the trust store for a Java application, such as `-Djavax.net.ssl.trustStore=/path/to/truststore.jks -Djavax.net.ssl.trustStorePassword=changeit`.

```
$ cd /path/to/tomcat/conf/
$ keytool \
-exportcert \
-alias openam.example.com \
-file openam.crt \
-storetype JCEKS
-keystore keystore.jceks
{
Enter keystore password:
Certificate stored in file <openam.crt>;
}
```

2. Import the certificate into the trust store:

```
$ keytool \
-importcert \
-alias openam.example.com \
-file openam.crt
-trustcacerts \
-keystore $JAVA_HOME/jre/lib/security/cacerts
{
Enter keystore password:
Owner: CN=openam.example.com, OU=Eng, O=ForgeRock.com, L=Grenoble, ST=Isere,
C=FR
Issuer: CN=openam.example.com, OU=Eng, O=ForgeRock.com, L=Grenoble, ST=Isere,
C=FR
Serial number: 4e789e40
Valid from: Tue Sep 20 16:08:00 CEST 2011 until: Mon Dec 19 15:08:00 CET 2011
Certificate fingerprints:
MD5:  31:08:11:3B:15:75:87:C2:12:08:E9:66:00:81:61:8D
SHA1: AA:90:2F:42:0A:F4:A9:A5:0C:90:A9:FC:69:FD:64:65:D9:78:BA:1D
Signature algorithm name: SHA1withRSA
Version: 3
Trust this certificate? [no]:
}
yes
Certificate was added to keystore
```

Request Security Considerations

AM can receive requests from multiple sources and for different purposes, such as authentication requests, RESTful requests to the endpoints, and POSTs that potentially may include a lot of data.

Containers usually have settings to mitigate against DoS attacks that POST large amounts of form data to your applications. Refer to your container documentation for more information about their settings, and how they can protect AM.

These settings, however, do not protect AM from receiving large amounts of POST data from other sources.

The following table summarizes the steps AM takes to protect against being overloaded, and how to adjust default values:

Task	Resources
Controlling the Maximum Size of Decompressed JWTs By default, AM rejects JWTs that expand to a size larger than 32 KiB (32768 bytes) when decrypted.	"Controlling the Maximum Size of Compressed JWTs".
Limiting the Size of the Request Body By default, AM rejects incoming requests whose body is larger than 1 MB (1048576 bytes) in size.	"Limiting the Size of the Request Body".

Controlling the Maximum Size of Compressed JWTs

A number of AM features accept JWTs to receive information. Some examples are:

- "OAuth 2.0 Remote Consent Service" in the *OAuth 2.0 Guide*, when it receives consent responses.
- The OAuth 2.0/OpenID Connect authorization service, when:
  - OpenID Connect clients send `request` in the *OAuth 2.0 Guide* parameters as a JWT instead of as HTTP parameters.
  - OpenID Connect clients register dynamically using software statements in the *OAuth 2.0 Guide*.
- The Authentication service, when configured to issue client-based sessions in the *Authentication and Single Sign-On Guide*.

These JWTs that AM receives can be signed and/or encrypted. Sometimes, larger JWTs are compressed to improve delivery speeds to AM.

Decompressing a JWT makes it expand in size. By default, AM rejects any JWT that expands to more than 32 KiB (32768 bytes), and throws an exception with a message similar to `JWT payload decompressed to larger than maximum allowed size`.

Ensure that the JWTs your clients send to AM are smaller than 32 KiB before compression, or increase the 32 KiB value to a reasonable limit. Take into account that AM performs decryption and decompression operations in its heap, and that you do not want to allow very large JWTs to, potentially, leave AM out of memory.

If you need to change the default value, perform the following steps:

1. Configure the `org.forgerock.json.jose.jwe.compression.max.decompressed.size.bytes` Java system property on the container where AM runs.

   For example, edit the `setenv.sh` file of the Apache Tomcat instance, and set the property with the new size in bytes:

   ```
   JAVA_OPTS="$JAVA_OPTS -Dorg.forgerock.json.jose.jwe.compression.max.decompressed.size.bytes=40960"
   ```

2. Restart the container for the changes to make effect.

Limiting the Size of the Request Body

HTTP requests are not limited by the specification. Rather, the method used limits the amount of data that a client can send. The GET and DELETE methods, for example, are limited by the size of the URL. The POST method is not. Instead, browsers and application servers limit the amount of data a request can send to your applications.

Ensure that the amount of data that reaches your applications and AM is not large enough to overwhelm them.

Application servers usually can mitigate against DoS attacks that POST large amounts of form data, but AM endpoints may receive large amounts of POST data in different ways, such as in JSON, JWT, or JWK formats.

By default, AM rejects incoming requests with a body larger than 1 MB (1048576 bytes) in size. It also returns an HTTP 413 error response, and logs a message similar to the following:

- `ERROR: Request Content-Length exceeds maximum allowed`, if the content's length was specified in the request.
- `ERROR: Counted request entity size exceeds maximum allowed`, if the content's length was not specified.

To change the default value, perform the following steps:

- Change the value of the `org.forgerock.openam.request.max.bytes.entity.size` advanced server property to the new size, in bytes.

  + *How Do I Configure Advanced Server Properties?*

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

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

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

  Save your changes.

  The property is hot-swappable. You do not need to restart AM for the changes to take effect.

Chapter 6
Removing Installations

This chapter shows you how to uninstall AM.

For instructions on removing AM agents, see the Web Policy Agents documentation, or the Java Policy Agents documentation.

To Remove an Instance

After you have deployed and configured AM, you may have as many as four locations where AM files are stored on your system.

Following the steps below removes the AM software and the internal configuration store. If you used an external configuration store, you can remove AM configuration data after removing all the software.

1. Shut down the web application container in which you deployed AM.

```
$ /etc/init.d/tomcat stop
Password:
Using CATALINA_BASE:   /path/to/tomcat
Using CATALINA_HOME:   /path/to/tomcat
Using CATALINA_TMPDIR: /path/to/tomcat/temp
Using JRE_HOME:        /path/to/jdk/jre
Using CLASSPATH:       /path/to/tomcat/bin/bootstrap.jar:
 /path/to/tomcat/bin/tomcat-juli.jar
```

2. Unconfigure AM by removing the configuration files found in the $HOME directory of the user running the web application container.

   A full uninstall of AM and configuration files consists of removing the following directories:

   - The configuration directory, by default `$HOME/openam`. If you did not use the default configuration location, check the value of the Base installation directory property under Deployment > Servers > *Server Name* > General > System.

   - The hidden directory that holds a file pointing to the configuration directory. For example, if you are using Apache Tomcat as the web container, this file could be `$HOME/.openamcfg/AMConfig_path_to_tomcat_webapps_openam_` OR `$HOME/.openssocfg/AMConfig_path_to_tomcat_webapps_openam_`.

```
$ rm -rf $HOME/openam $HOME/.openamcfg
```

   Or:

```
$ rm -rf $HOME/openam $HOME/.openssocfg
```

If you used an external configuration store, you must remove the configuration manually from your external directory server. The default base DN for the AM configuration is `dc=openam,dc=forgerock,dc=org`.

Note

At this point, you can restart the web container and configure AM anew if you only want to start over with a clean configuration rather than removing AM completely.

3. Undeploy the AM web application.

   For example, if you are using Apache Tomcat as the web container, remove the `.war` file and expanded web application from the container.

   ```
   $ cd /path/to/tomcat/webapps/
   $ rm -rf openam.war openam/
   ```

Chapter 7
Troubleshooting Installations

AM can capture information in debug log files that are useful when troubleshooting AM problems. "Debug Logging" in the *Setup and Maintenance Guide* describes how to enable debug logging after AM has been started.

It is also possible to capture debug logs while installing AM. This can be useful if you need to troubleshoot an installation problem.

To Troubleshoot an Installation

Follow these steps to capture debug logs while installing AM on Tomcat:

1. If Tomcat is already started, stop it.
2. Specify the `-Dcom.iplanet.services.debug.level=message` option in the `CATALINA_OPTS` environment variable:

   ```
   $ export CATALINA_OPTS=-Dcom.iplanet.services.debug.level=message
   ```

   There are several ways that you can specify the `CATALINA_OPTS` environment variable. You can set the variable:

   - In the `/path/to/tomcat/bin/setenv.sh` file
   - In the login shell of the user who runs Tomcat
3. Run the AM installation. Debug log files containing troubleshooting information appear in the `/path/to/openam/openam/debug` directory.
4. When you have completed AM installation and no longer need to capture debug logs, stop Tomcat, revert the debug logging options, and restart Tomcat.

Chapter 8
Reference

This reference section covers settings and other information relating to installing AM.

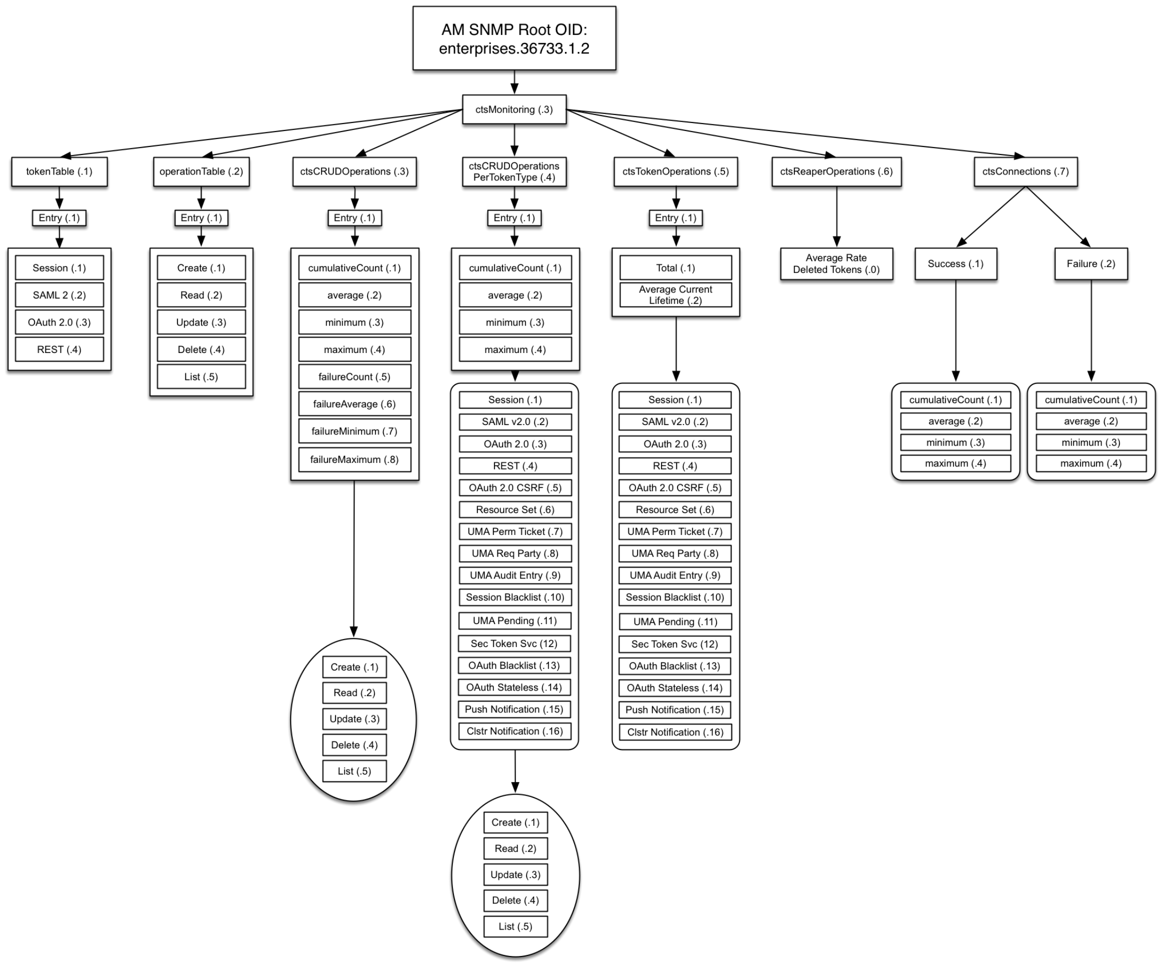
Core Token Service (CTS) Object Identifiers

The OIDs related to SNMP monitoring of CTS follow guidance described in RFC 1271.

The OIDs listed in this section include the prefix assigned to ForgeRock, `enterprises.36733`. They also include the entries associated with AM (1), SNMP (2), and CTS monitoring (3): `1.2.3`.

Therefore, the root OID for all CTS monitored components is `enterprises.36733.1.2.3`. All individual monitored CTS components are suffixes that are consistent with the image shown here.

Diagram of CTS OIDs



CTS Token Type OIDs

The table below shows how OIDs are split into different token types. Do not forget the prefix. For example, the complete OID for monitoring SAML v2.0 tokens is `enterprises.36733.1.2.3.1.1.2`

The options for the token table are shown in the following table. For example, the token table OID for SAML v2.0 is based on the entries associated with ForgeRock, `enterprises.36733`, AM `1`, SNMP `2`, CTS Monitoring `3`, token table `1`, entry `1`, and SAML v2.0 `2`, which is `enterprises.36733.1.2.3.1.1.2`.

CTS Monitoring OID Categories

OID, by Token Type	Description
`enterprises.36733.1.2.3.1.1.1`	Session
`enterprises.36733.1.2.3.1.1.2`	SAML v2.0
`enterprises.36733.1.2.3.1.1.3`	OAuth 2.0
`enterprises.36733.1.2.3.1.1.4`	REST
`enterprises.36733.1.2.3.1.1.5`	OAuth 2.0 CSRF Protection
`enterprises.36733.1.2.3.1.1.6`	UMA Resource
`enterprises.36733.1.2.3.1.1.7`	UMA Permission Ticket
`enterprises.36733.1.2.3.1.1.8`	UMA Requesting Party
`enterprises.36733.1.2.3.1.1.9`	UMA Audit Entry
`enterprises.36733.1.2.3.1.1.10`	Session Blacklist
`enterprises.36733.1.2.3.1.1.11`	UMA Pending Request
`enterprises.36733.1.2.3.1.1.12`	Security Token Service
`enterprises.36733.1.2.3.1.1.13`	OAuth 2.0 Blacklist
`enterprises.36733.1.2.3.1.1.14`	OAuth 2.0 Client-Based
`enterprises.36733.1.2.3.1.1.15`	Push Notification
`enterprises.36733.1.2.3.1.1.16`	Cluster-wide Notification

CTS Monitoring Operation Types

OIDs related to CTS monitoring operations are based on basic CRUD operations (plus list).

The options for the operation table are shown in the following table.

CTS Monitoring Operation Types

OID, by Operation	Description
`enterprises.36733.1.2.3.2.1.1`	Create
`enterprises.36733.1.2.3.2.1.2`	Read
`enterprises.36733.1.2.3.2.1.3`	Update
`enterprises.36733.1.2.3.2.1.4`	Delete
`enterprises.36733.1.2.3.2.1.5`	List

CTS Monitoring Entry Data Types

CTS monitoring entries use the following data types:

`Counter64`

A 64-bit, unsigned integer type.

`Counter64` is a standard data type returned by SNMP OIDs. For more information, see Structure of Management Information Version 2.

`Float2dp`

A floating point number with the value `d-2` in the `DISPLAY-HINT` clause. SNMP clients that handle the `DISPLAY-HINT` clause will correctly display the value as a floating point number with two decimal places. Other types of clients that do not handle the `DISPLAY-HINT` clause will incorrectly display the value as an integer that is one hundred times larger than the correct value.

`Float2dp` is a custom data type returned by some ForgeRock CTS OIDs.

CTS CRUD Operation Entries

The OIDs in this table relate to all CRUD (and list) operations.

The options for the CRUD operations table are shown in the following tables. Each value is associated with CRUD and list operations.

CTS CRUD Operation Entries

OID, by Operation Entry	Data Type	Description
`enterprises.36733.1.2.3.3.1.1`	`Counter64`	Cumulative count
`enterprises.36733.1.2.3.3.1.2`	`Float2dp`	Average (in period)
`enterprises.36733.1.2.3.3.1.3`	`Counter64`	Minimum (in period)
`enterprises.36733.1.2.3.3.1.4`	`Counter64`	Maximum (in period)
`enterprises.36733.1.2.3.3.1.5`	`Counter64`	Cumulative failure count
`enterprises.36733.1.2.3.3.1.6`	`Float2dp`	Average failures (in period)
`enterprises.36733.1.2.3.3.1.7`	`Counter64`	Minimum failures (in period)
`enterprises.36733.1.2.3.3.1.8`	`Counter64`	Maximum failures (in period)

Each of the options in this table can be divided into CRUD and list related operations. The suffix OID for such operations is as follows:

- 1: Create
- 2: Read
- 3: Update
- 4: Delete
- 5: List

For example, since the OID for cumulative count is `enterprises.36733.1.2.3.3.1.1`, the OID for the cumulative count of delete operations is `enterprises.36733.1.2.3.3.1.1.4`

CTS CRUD Operation Table Cumulative Operations

Cumulative Count Operations OID	Data Type	Description
`enterprises.36733.1.2.3.3.1.1.1`	`Counter64`	Cumulative count of CREATE operations
`enterprises.36733.1.2.3.3.1.1.2`	`Counter64`	Cumulative count of READ operations
`enterprises.36733.1.2.3.3.1.1.3`	`Counter64`	Cumulative count of UPDATE operations
`enterprises.36733.1.2.3.3.1.1.4`	`Counter64`	Cumulative count of DELETE operations
`enterprises.36733.1.2.3.3.1.1.5`	`Counter64`	Cumulative count of LIST operations

CTS CRUD Operation Table Average Operations (In Period)

Average Number Operations OID	Data Type	Description
`enterprises.36733.1.2.3.3.1.2.1`	`Float2dp`	Average number of CREATE operations (in period)
`enterprises.36733.1.2.3.3.1.2.2`	`Float2dp`	Average number of READ operations (in period)
`enterprises.36733.1.2.3.3.1.2.3`	`Float2dp`	Average number of UPDATE operations (in period)
`enterprises.36733.1.2.3.3.1.2.4`	`Float2dp`	Average number of DELETE operations (in period)
`enterprises.36733.1.2.3.3.1.2.5`	`Float2dp`	Average number of LIST operations (in period)

CTS CRUD Operation Table Minimum Operations (In Period)

Minimum Number Operations OID	Data Type	Description
`enterprises.36733.1.2.3.3.1.3.1`	`Counter64`	Minimum number of CREATE operations (in period)
`enterprises.36733.1.2.3.3.1.3.2`	`Counter64`	Minimum number of READ operations (in period)
`enterprises.36733.1.2.3.3.1.3.3`	`Counter64`	Minimum number of UPDATE operations (in period)
`enterprises.36733.1.2.3.3.1.3.4`	`Counter64`	Minimum number of DELETE operations (in period)
`enterprises.36733.1.2.3.3.1.3.5`	`Counter64`	Minimum number of LIST operations (in period)

CTS CRUD Operation Table Maximum Operations (In Period)

Maximum Number Operations OID	Data Type	Description
`enterprises.36733.1.2.3.3.1.4.1`	`Counter64`	Maximum number of CREATE operations (in period)
`enterprises.36733.1.2.3.3.1.4.2`	`Counter64`	Maximum number of READ operations (in period)
`enterprises.36733.1.2.3.3.1.4.3`	`Counter64`	Maximum number of UPDATE operations (in period)
`enterprises.36733.1.2.3.3.1.4.4`	`Counter64`	Maximum number of DELETE operations (in period)
`enterprises.36733.1.2.3.3.1.4.5`	`Counter64`	Maximum number of LIST operations (in period)

CTS CRUD Operation Table Cumulative Failure Operations

Cumulative Failure Operations OID	Data Type	Description
`enterprises.36733.1.2.3.3.1.5.1`	`Counter64`	Cumulative Failure of CREATE operations (in period)
`enterprises.36733.1.2.3.3.1.5.2`	`Counter64`	Cumulative Failure of READ operations (in period)
`enterprises.36733.1.2.3.3.1.5.3`	`Counter64`	Cumulative Failure of UPDATE operations (in period)
`enterprises.36733.1.2.3.3.1.5.4`	`Counter64`	Cumulative Failure of DELETE operations (in period)

Cumulative Failure Operations OID	Data Type	Description
`enterprises.36733.1.2.3.3.1.5.5`	`Counter64`	Cumulative Failure of LIST operations (in period)

CTS CRUD Operation Table Average Failure Operations in Period

Average Number, Failure Operations OID	Data Type	Description
`enterprises.36733.1.2.3.3.1.6.1`	`Float2dp`	Average number of CREATE operations failures (in period)
`enterprises.36733.1.2.3.3.1.6.2`	`Float2dp`	Average number of READ operations failures (in period)
`enterprises.36733.1.2.3.3.1.6.3`	`Float2dp`	Average number of UPDATE operations failures (in period)
`enterprises.36733.1.2.3.3.1.6.4`	`Float2dp`	Average number of DELETE operations failures (in period)
`enterprises.36733.1.2.3.3.1.6.5`	`Float2dp`	Average number of LIST operations failures (in period)

CTS CRUD Operation Table Minimum Operations Failures in Period

Minimum Number, Operations Failures OID	Data Type	Description
`enterprises.36733.1.2.3.3.1.7.1`	`Counter64`	Minimum number of CREATE operations failures (in period)
`enterprises.36733.1.2.3.3.1.7.2`	`Counter64`	Minimum number of READ operations failures (in period)
`enterprises.36733.1.2.3.3.1.7.3`	`Counter64`	Minimum number of UPDATE operations failures (in period)
`enterprises.36733.1.2.3.3.1.7.4`	`Counter64`	Minimum number of DELETE operations failures (in period)
`enterprises.36733.1.2.3.3.1.7.5`	`Counter64`	Minimum number of LIST operations failures (in period)

CTS CRUD Operation Table Maximum Operations Failures in Period

Maximum Number, Operations Failures OID	Data Type	Description
`enterprises.36733.1.2.3.3.1.8.1`	`Counter64`	Maximum number of CREATE operations failures (in period)
`enterprises.36733.1.2.3.3.1.8.2`	`Counter64`	Maximum number of READ operations failures (in period)

Maximum Number, Operations Failures OID	Data Type	Description
`enterprises.36733.1.2.3.3.1.8.3`	`Counter64`	Maximum number of UPDATE operations failures (in period)
`enterprises.36733.1.2.3.3.1.8.4`	`Counter64`	Maximum number of DELETE operations failures (in period)
`enterprises.36733.1.2.3.3.1.8.5`	`Counter64`	Maximum number of LIST operations failures (in period)

CTS CRUD Operations Per Token Type

OIDs that start with `enterprises.36733.1.2.3.4.1` are labels for CTS CRUD operations per token type.

Tokens of each type can be created, read, updated, deleted, and listed. Each of these types can be measured cumulatively. They can also be measured over a period of time (default=10 seconds), as an average, minimum, and maximum.

OID suffixes for CRUD operations are defined according to the following rules.

The first part of the OID is `enterprises.36733.1.2.3.4.1`.

The next OID suffix specifies a metric:

CTS CRUD Operation Metrics

OID Suffix	Data Type	Metric
1	`Counter64`	Cumulative count
2	`Float2dp`	Average (in period)
3	`Counter64`	Minimum (in period)
4	`Counter64`	Maximum (in period)

The next OID suffix specifies a token type:

CTS CRUD Operation Token Types

OID Suffix	Token Type
1	Session
2	SAML v2.0
3	OAuth 2
4	REST
5	OAuth 2.0 CSRF Protection
6	UMA Resource

OID Suffix	Token Type
7	UMA Permission Ticket
8	UMA Requesting Party
9	UMA Audit Entry
10	Session Blacklist
11	UMA Pending Request
12	Security Token Service
13	OAuth 2.0 Blacklist
14	OAuth 2.0 Client-Based
15	Push Notification
16	Cluster-wide Notification

The final OID suffix specifies an operation:

CTS CRUD Operations

OID Suffix	Operation
1	Create
2	Read
3	Update
4	Delete
5	List

The following examples illustrate OID construction for CTS CRUD operations per token type.

OID Examples for CTS CRUD Operations Per Token Type

OID	Data Type	Description
`enterprises.36733.1.2.3.4.1.1.1.3`	`Counter64`	Cumulative count of updated Session tokens
`enterprises.36733.1.2.3.4.1.4.3.4`	`Counter64`	Maximum deleted OAuth 2.0 tokens (in period)
`enterprises.36733.1.2.3.4.1.2.10.5`	`Float2dp`	Average listed Session Blacklist tokens (in period)

CTS Token Operation Status

The CTS token OIDs defined in this section specify the total number of tokens of each type and their average current lifetimes.

The options for token operations are shown in the following tables. Total and average current lifetimes are associated with each CTS token type.

CTS Total Tokens, by Type

Total Tokens, by Type	Data Type	Description
`enterprises.36733.1.2.3.5.1.1.1`	`Counter64`	Total number of Session tokens
`enterprises.36733.1.2.3.5.1.1.2`	`Counter64`	Total number of SAML v2.0 tokens
`enterprises.36733.1.2.3.5.1.1.3`	`Counter64`	Total number of OAuth 2.0 tokens
`enterprises.36733.1.2.3.5.1.1.4`	`Counter64`	Total number of REST tokens
`enterprises.36733.1.2.3.5.1.1.5`	`Counter64`	Total number of OAuth 2.0 CSRF Protection tokens
`enterprises.36733.1.2.3.5.1.1.6`	`Counter64`	Total number of UMA Resource tokens
`enterprises.36733.1.2.3.5.1.1.7`	`Counter64`	Total number of UMA Permission Ticket tokens
`enterprises.36733.1.2.3.5.1.1.8`	`Counter64`	Total number of UMA Requesting Party tokens
`enterprises.36733.1.2.3.5.1.1.9`	`Counter64`	Total number of UMA Audit Entry tokens
`enterprises.36733.1.2.3.5.1.1.10`	`Counter64`	Total number of Session Blacklist tokens
`enterprises.36733.1.2.3.5.1.1.11`	`Counter64`	Total number of UMA Pending Request tokens
`enterprises.36733.1.2.3.5.1.1.12`	`Counter64`	Total number of Security Token Service tokens
`enterprises.36733.1.2.3.5.1.1.13`	`Counter64`	Total number of OAuth 2.0 Blacklist tokens
`enterprises.36733.1.2.3.5.1.1.14`	`Counter64`	Total number of OAuth 2.0 client-based tokens
`enterprises.36733.1.2.3.5.1.1.15`	`Counter64`	Total number of Push Notification tokens
`enterprises.36733.1.2.3.5.1.1.16`	`Counter64`	Total number of Cluster-wide Notification tokens

CTS Token Average Lifetime, by Type

Average Token Lifetime, by Type	Data Type	Description
`enterprises.36733.1.2.3.5.1.2.1`	`Counter64`	Average lifetime of Session tokens in seconds
`enterprises.36733.1.2.3.5.1.2.2`	`Counter64`	Average lifetime of SAML v2.0 tokens in seconds
`enterprises.36733.1.2.3.5.1.2.3`	`Counter64`	Average lifetime of OAuth 2.0 tokens in seconds
`enterprises.36733.1.2.3.5.1.2.4`	`Counter64`	Average lifetime of REST tokens in seconds
`enterprises.36733.1.2.3.5.1.2.5`	`Counter64`	Average lifetime of OAuth 2.0 CSRF Protection tokens in seconds
`enterprises.36733.1.2.3.5.1.2.6`	`Counter64`	Average lifetime of UMA Resource tokens in seconds
`enterprises.36733.1.2.3.5.1.2.7`	`Counter64`	Average lifetime of UMA Permission Ticket tokens in seconds
`enterprises.36733.1.2.3.5.1.2.8`	`Counter64`	Average lifetime of UMA Requesting Party tokens in seconds
`enterprises.36733.1.2.3.5.1.2.9`	`Counter64`	Average lifetime of UMA Audit Entry tokens in seconds
`enterprises.36733.1.2.3.5.1.2.10`	`Counter64`	Average lifetime of Session Blacklist tokens in seconds
`enterprises.36733.1.2.3.5.1.2.11`	`Counter64`	Average lifetime of UMA Pending Request tokens in seconds
`enterprises.36733.1.2.3.5.1.2.12`	`Counter64`	Average lifetime of Security Token Service tokens in seconds
`enterprises.36733.1.2.3.5.1.2.13`	`Counter64`	Average lifetime of OAuth 2.0 Blacklist tokens in seconds
`enterprises.36733.1.2.3.5.1.2.14`	`Counter64`	Average lifetime of OAuth 2.0 client-based tokens in seconds
`enterprises.36733.1.2.3.5.1.2.15`	`Counter64`	Average lifetime of Push Notification tokens in seconds
`enterprises.36733.1.2.3.5.1.2.16`	`Counter64`	Average lifetime of Cluster-wide Notification tokens in seconds

CTS Reaper Run Information

The CTS reaper deletes unused or expired tokens. Unless AM is in a shutdown cycle, the CTS reaper is designed to run continuously. By default, the CTS reaper runs in fixed intervals, unless AM is in the process of shutting down.

A single OID, `enterprises.36733.1.2.3.6.0`, relates to the CTS reaper. This OID:

- Specifies the average rate of deleted tokens per CTS reaper run
- Has the `Float2dp`data type.

CTS Connection Factory OIDs

Every request for a CTS token is a request to the `CTSConnectionFactory`. Such requests can either succeed or fail. The following OIDs provide measures for both such connections. The CTSConnectionFactory OIDs are also measured using a rate window system, similar to all the other CTS OIDs, except the CTS Reaper.

As there are no indexes required to look up the value of CTSConnectionFactory OIDs, they end in 0. Success or failure of these OIDs are not specific to any operation or token type.

The following tables list the OIDs related to the CTSConnectionFactory.

CTSConnectionFactory, Successful Connections

Successes, CTSConnectionFactory	Data Type	Description
`enterprises.36733.1.2.3.7.1.1.0`	`Counter64`	Cumulative number of successful connections
`enterprises.36733.1.2.3.7.1.2.0`	`Float2dp`	Average number of successful connections (in period)
`enterprises.36733.1.2.3.7.1.3.0`	`Counter64`	Minimum number of successful connections (in period)
`enterprises.36733.1.2.3.7.1.4.0`	`Counter64`	Maximum number of successful connections (in period)

CTSConnectionFactory, Failed Connections

Failures, CTSConnectionFactory	Data Type	Description
`enterprises.36733.1.2.3.7.2.1.0`	`Counter64`	Cumulative number of failed connections
`enterprises.36733.1.2.3.7.2.2.0`	`Float2dp`	Average number of failed connections (in period)

Failures, CTSConnectionFactory	Data Type	Description
`enterprises.36733.1.2.3.7.2.3.0`	`Counter64`	Minimum number of failed connections (in period)
`enterprises.36733.1.2.3.7.2.4.0`	`Counter64`	Maximum number of failed connections (in period)

Command-line Tool Reference

Name

configurator.jar — install or upgrade AM using a configuration file

Synopsis

`configurator.jar` {options}

Description

This executable .jar file, openam-configurator-tool-14.1.2.24.jar, lets you perform silent installation, configuring a deployed AM server by applying settings from a configuration file.

Options

The following options are supported.

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

Configure a deployed AM web application archive using the specified configuration file. Installation and upgrade configuration files are described in the sections 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.

Installation Configuration File

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

Server Properties

These properties pertain to the AM server instance.

SERVER_URL

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

DEPLOYMENT_URI

URI where you want to deploy AM on the web container, such as `/openam`

BASE_DIR

Configuration directory where AM stores files and embedded configuration directory servers, such as `$HOME/openam`

locale

The user locale, such as `en_GB`

PLATFORM_LOCALE

The locale of the AM server, such as `en_US`

AM_ENC_KEY

The password encryption key, which must be the same on all servers in a multi-server installation, such as `O6QWwHPO4os+zEz3Nqn/2daAYWyiFE32`. If left blank, installing AM generates a random password encryption key that you can view in the AM console under Deployment > Servers > *Server Name* > Security.

ADMIN_PWD

Password of the AM administrator user `amadmin`, which must be at least 8 characters in length and must match that of other servers in a multiserver deployment

COOKIE_DOMAIN

Name of the trusted DNS domain AM returns to a browser when it grants a session ID to a user. By default, it is set to the full URL that was used to access the configurator, such as `example.com`.

ACCEPT_LICENSES

Optional boolean property that can be set to always auto-accept the software license agreement and suppress the display of 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-configurator-tool-14.1.2.24.jar file.

Configuration Store Properties

These properties pertain to the directory server where AM stores its configuration.

DATA_STORE

Type of the configuration data store. The value `embedded` means set up AM with an embedded, DS configuration store. The value `dirServer` means an external directory server, such as ForgeRock Directory Services, or Oracle Directory Server Enterprise Edition. If you set this to `dirServer`, and the configuration store contains the configuration of other AM servers, then the server is added to the existing multiserver installation.

DIRECTORY_SSL

To use LDAP without SSL, set this to `SIMPLE`. To use LDAP with SSL, set this to `SSL`.

DIRECTORY_SERVER

Fully qualified domain name of the configuration store directory server host, such as `opendj.example.com`

DIRECTORY_PORT

LDAP or LDAPS port number for the configuration store directory server, such as 389 or 636

DIRECTORY_ADMIN_PORT

Administration port number for the configuration store directory server, such as 4444

DIRECTORY_JMX_PORT

Java Management eXtension port number, such as `1689`, used with the DS embedded configuration store

ROOT_SUFFIX

Root suffix distinguished name (DN) for the configuration store, such as `o=openam`

DS_DIRMGRDN

Distinguished name of the directory manager of the configuration store, such as `cn=Directory Manager`

DS_DIRMGRPASSWD

Password for the directory manager of the configuration store

User Data Store Properties

These properties pertain to the directory server where AM stores user profiles. If you do not include these properties, or you leave these properties commented out, then AM uses the same directory server as it uses for the configuration store.

USERSTORE_TYPE

The type of directory server used. Valid values include the following.

- `LDAPv3ForOpenDS`: ForgeRock OpenDJ or Sun OpenDS
- `LDAPv3ForAD`: Active Directory with host and port settings
- `LDAPv3ForADDC`: Active Directory with a Domain Name setting

- `LDAPv3ForADAM`: Active Directory Application Mode
- `LDAPv3ForODSEE`: Sun Java System Directory Server
- `LDAPv3ForTivoli`: IBM Tivoli Directory Server

USERSTORE_SSL

To use LDAP without SSL, set this to `SIMPLE`. To use LDAP with SSL, set this to `SSL`.

USERSTORE_DOMAINNAME

If `USERSTORE_TYPE` is `LDAPv3ForADDC`, you set this to the Active Directory Domain Name, such as `ad.example.com`, and then set only the `USERSTORE_SSL`, `USERSTORE_MGRDN`, and `USERSTORE_PASSWD` additional parameters. This lets Active Directory use DNS to retrieve service locations. Otherwise, do not use.

USERSTORE_HOST

Fully qualified domain name of the user data store directory server, such as `opendj.example.com`

USERSTORE_PORT

Port number of the user data store. Default for LDAP is 389, and for LDAP over SSL is 636.

USERSTORE_SUFFIX

Root suffix distinguished name for the user data in the directory, such as `dc=example,dc=com`

USERSTORE_MGRDN

Distinguished name of the directory manager of the user data store, such as `cn=Directory Manager`

USERSTORE_PASSWD

Password for the directory manager of the user data store

Site Properties

These properties pertain when you configure multiple AM servers in a site deployment, where a load balancer spreads request across multiple servers. Use the `DS_EMB_REPL*` and `existingserverid` properties only for the second and subsequent servers in a site configuration.

LB_SITE_NAME

The name of the AM site

LB_PRIMARY_URL

The load balancer URL for the site, such as `http://lb.example.com:80/openam`.

DS_EMB_REPL_FLAG

Enable use of the embedded configuration store by setting this parameter to `embReplFlag`, only if the `DATA_STORE` parameter is set to `embedded`. Use the other `DS_EMB_REPL*` parameters in this section to set up configuration store data replication.

DS_EMB_REPL_REPLPORT1

Replication port number for the new AM server you are installing, such as 58989

DS_EMB_REPL_HOST2

Host name of an existing AM server housing the configuration store directory server with which to replicate, such as `openam1.example.com`

DS_EMB_REPL_ADMINPORT2

Administration port number for the configuration store directory server used by the existing AM server, such as 4444

DS_EMB_REPL_REPLPORT2

Replication port number for the configuration store directory server used by the existing AM server, such as 50899

existingserverid

Full URL of the existing AM server, such as `http://server1.example.com:8080/openam`

Upgrade Configuration File

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

Upgrade Properties

SERVER_URL

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

DEPLOYMENT_URI

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

ACCEPT_LICENSES

Optional boolean property that can be set to always auto-accept the software license agreement and suppress displaying the license acceptance screen to the user. A value of `true` auto-accepts the license; any other value will be assumed to equal `false`, resulting in the presentation of the license. Default value is `false`. This property takes precedence over the `--acceptLicense` option, which can also be passed in to the application with the openam-configurator-tool-14.1.2.24.jar file.

Examples

The following example shows a configuration file to install a server with an external user data store.

```
# Server properties, AM_ENC_KEY="" means generate random key
SERVER_URL=http://openam.example.com:8080
DEPLOYMENT_URI=/openam
BASE_DIR=$HOME/openam
locale=en_US
PLATFORM_LOCALE=en_US
AM_ENC_KEY=
ADMIN_PWD=change3me
COOKIE_DOMAIN=openam.example.com
ACCEPT_LICENSES=true

# Embedded configuration data store
DATA_STORE=embedded
DIRECTORY_SSL=SIMPLE
DIRECTORY_SERVER=openam.example.com
DIRECTORY_PORT=50389
DIRECTORY_ADMIN_PORT=4444
DIRECTORY_JMX_PORT=1689
ROOT_SUFFIX=o=openam
DS_DIRMGRDN=cn=Directory Manager
DS_DIRMGRPASSWD=chang3me

# External OpenDJ based user data store
USERSTORE_TYPE=LDAPv3ForOpenDS
USERSTORE_SSL=SIMPLE
#USERSTORE_DOMAINNAME=ad.example.com
USERSTORE_HOST=opendj.example.com
USERSTORE_PORT=389
USERSTORE_SUFFIX=dc=example,dc=com
USERSTORE_MGRDN=cn=Directory Manager
USERSTORE_PASSWD=secret12

# Uncomment to specify the site for the first server in a site configuration
#LB_SITE_NAME=lb
#LB_PRIMARY_URL=http://lb.example.com:80/openam
```

The following example shows a configuration file to install the second server in a site configuration.

```
# Server properties, AM_ENC_KEY from first server
SERVER_URL=http://server2.example.com:8080
DEPLOYMENT_URI=/openam
BASE_DIR=$HOME/openam
locale=en_US
PLATFORM_LOCALE=en_US
AM_ENC_KEY=O6QWwHPO4os+zEz3Nqn/2daAYWyiFE32
ADMIN_PWD=change3me
AMLDAPUSERPASSWD=secret12
COOKIE_DOMAIN=openam.example.com
ACCEPT_LICENSES=true
```

```

# Embedded configuration data store
DATA_STORE=embedded
DIRECTORY_SSL=SIMPLE
DIRECTORY_SERVER=server2.example.com
DIRECTORY_PORT=50389
DIRECTORY_ADMIN_PORT=4444
DIRECTORY_JMX_PORT=1689
ROOT_SUFFIX=o=openam
DS_DIRMGRDN=cn=Directory Manager
DS_DIRMGRPASSWD=chang3me

# External OpenDJ based user data store
USERSTORE_TYPE=LDAPv3ForOpenDS
USERSTORE_SSL=SIMPLE
#USERSTORE_DOMAINNAME=ad.example.com
USERSTORE_HOST=opendj.example.com
USERSTORE_PORT=389
USERSTORE_SUFFIX=dc=example,dc=com
USERSTORE_MGRDN=cn=Directory Manager
USERSTORE_PASSWD=secret12

# Site properties
LB_SITE_NAME=lb
LB_PRIMARY_URL=http://lb.example.com:80/openam
DS_EMB_REPL_FLAG=embReplFlag
DS_EMB_REPL_REPLPORT1=58989
DS_EMB_REPL_HOST2=server1.example.com
DS_EMB_REPL_ADMINPORT2=4444
DS_EMB_REPL_REPLPORT2=50889
existingserverid=http://server1.example.com:8080/openam
```

The following example shows a configuration file to upgrade an AM server.

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

The following example uses a configuration file with the `--acceptLicense` option on the command line.

```
$ java \
 -jar openam-configurator-tool-14.1.2.24.jar \
 -f config.file \
 --acceptLicense
```

Appendix A. Getting Support

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

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

ForgeRock publishes comprehensive documentation online:

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

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

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

Appendix B. Supported LDIF Files

AM installation deploys several LDIF files that can be used to create the schemas required by AM in an external configuration data store. LDIF files are available for Microsoft Active Directory, Microsoft Active Directory Lightweight Directory Services, Oracle Directory Server Enterprise Edition, ForgeRock Directory Services, Oracle Unified Directory, and IBM Tivoli Directory Server.

The following tables provide descriptions for each LDIF file:

Microsoft Active Directory LDIF Files

LDIF File	Description
ad_config_schema.ldif	LDIF for the configuration schema.
ad_dashboard.ldif	LDIF to support the dashboard service.
ad_deviceprint.ldif	LDIF to support the device print service.
ad_kba.ldif	LDIF to support the User Self-Service's knowledge-based questions and answers service.
ad_oathdevices.ldif	LDIF to support registered devices for the OATH authentication service.
ad_pushdevices.ldif	LDIF to support registered devices for the PUSH notification service.
ad_user_schema.ldif	LDIF for the user schema.
ad_webauthndevices.ldif	LDIF to support registered devices for the Web Authentication (WebAuthn) authentication service.

Microsoft Active Directory Lightweight Directory Services LDIF Files

LDIF File	Description
adam_dashboard.ldif	LDIF to support the dashboard service.
adam_deviceprint.ldif	LDIF to support the device print service.
adam_kba.ldif	LDIF to support the User Self-Service's knowledge-based questions and answers.
adam_oathdevices.ldif	LDIF to support registered devices for the OATH authentication service.
adam_pushdevices.ldif	LDIF to support registered devices for the PUSH notification service.
adam_user_schema.ldif	LDIF for the user schema.
adam_webauthndevices.ldif	LDIF to support registered devices for the Web Authentication (WebAuthn) authentication service.

Oracle Directory Server Enterprise Edition LDIF Files

LDIF File	Description
amsdk_plugin	Folder containg the AM SDK LDIF files: amsdk_init_template.ldif and amsdk_sunone_schema2.ldif.
odsee_config_index.ldif	LDIF for the ODSEE configuration indexes.
odsee_config_schema.ldif	LDIF for the ODSEE configuration schema.
odsee_dashboard.ldif	LDIF to support the dashboard service.
odsee_deviceprint.ldif	LDIF to support the device print service.
odsee_kba.ldif	LDIF to support the User Self-Service's knowledge-based questions and answers.
odsee_oathdevices.ldif	LDIF to support registered devices for the OATH authentication service.
odsee_pushdevices.ldif	LDIF to support registered devices for the PUSH notification service.
odsee_user_index.ldif	LDIF for the user respository indexes.
odsee_user_schema.ldif	LDIF for the user repository schema.
odsee_userinit.ldif	LDIF for the setting up user session initialization.
odsee_webauthndevices.ldif	LDIF to support registered devices for the Web Authentication (WebAuthn) authentication service.

DS LDIF Files

LDIF File	Description
oath_2fa.ldif	LDIF for the OATH two-factor authentication service.
opendj_aci_lift_user_password_restriction.ldif	LDIF to add an ACI entry to the root suffix to allow users to modify the user password attribute.
opendj_aci_remove_blanket_deny_all.ldif	LDIF to lift any user password restrictions for upgrade.
opendj_add_kba_attempts.ldif	LDIF to upgrade a user data store from a version earlier than AM 6 to support account lockout when the user fails to answer their security questions a number of times.
opendj_config_schema.ldif	LDIF for the DS configuration schema.
opendj_dashboard.ldif	LDIF to support the dashboard service.
opendj_deviceprint.ldif	LDIF to support the device print service.
opendj_embinit.ldif	LDIF for the DS user management and SMS/configuration datastore schema for embedded DS deployments.
opendj_kba.ldif	LDIF to support the User Self-Service's knowledge-based questions and answers.
opendj_oathdevices.ldif	LDIF to support registered devices for the OATH authentication service.
opendj_pushdevices.ldif	LDIF to support registered devices for the PUSH notification service.
opendj_remove_config_schema.ldif	LDIF to remove the configuration schema.
opendj_remove_user_schema.ldif	LDIF to remove the user schema.
opendj_retry_limit_node_count.ldif	LDIF to upgrade the identity store to support persisting failed login attempts to the user's profile when using the "Retry Limit Decision Node" in the *Authentication and Single Sign-On Guide*. There are no equivalent files for other supported directory servers. The required objects are available in the `*user_schema.ldif` files.
opendj_uma_audit.ldif	LDIF to add auditing capabilities for the UMA service.
opendj_uma_labels_schema.ldif	LDIF to add a schema for the UMA service labels.
opendj_uma_pending_requests.ldif	LDIF to add pending requests for the UMA service.
opendj_uma_resource_set_labels.ldif	LDIF to support labels for UMA resources.
opendj_uma_resource_sets.ldif	LDIF to support UMA resources.
opendj_update_aci_kba_attempts.ldif	LDIF to upgrade a user data store from a version earlier than AM 6 to support account lockout when the user fails to answer their security questions a number of times.
opendj_user_index.ldif	LDIF for the user respository indexes.

LDIF File	Description
opendj_user_schema.ldif	LDIF for the user repository schema.
opendj_userinit.ldif	LDIF for the setting up user session initialization.
opendj_webauthndevices.ldif	LDIF to support registered devices for the Web Authentication (WebAuthn) authentication service.
push_2fa.ldif	LDIF for the push two-factor authentication service.

Tivoli LDIF Files

LDIF File	Description
tivoli_dashboard.ldif	LDIF to support the dashboard service.
tivoli_deviceprint.ldif	LDIF to support the device print service.
tivoli_kba.ldif	LDIF to support the User Self-Service's knowledge-based questions and answers.
tivoli_oathdevices.ldif	LDIF to support registered devices for the OATH authentication service.
tivoli_pushdevices.ldif	LDIF to support registered devices for the PUSH notification service.
tivoli_user_schema.ldif	LDIF for the user repository schema.
tivoli_webauthndevices.ldif	LDIF to support registered devices for the Web Authentication (WebAuthn) authentication service.

Glossary

Access control	Control to grant or to deny access to a resource.
Account lockout	The act of making an account temporarily or permanently inactive after successive authentication failures.
Actions	Defined as part of policies, these verbs indicate what authorized identities can do to resources.
Advice	In the context of a policy decision denying access, a hint to the policy enforcement point about remedial action to take that could result in a decision allowing access.
Agent administrator	User having privileges only to read and write agent profile configuration information, typically created to delegate agent profile creation to the user installing a web or Java agent.
Agent authenticator	Entity with read-only access to multiple agent profiles defined in the same realm; allows an agent to read web service profiles.
Application	In general terms, a service exposing protected resources. In the context of AM policies, the application is a template that constrains the policies that govern access to protected resources. An application can have zero or more policies.
Application type	Application types act as templates for creating policy applications. Application types define a preset list of actions and functional logic, such as policy lookup and resource comparator logic.

	Application types also define the internal normalization, indexing logic, and comparator logic for applications.
Attribute-based access control (ABAC)	Access control that is based on attributes of a user, such as how old a user is or whether the user is a paying customer.
Authentication	The act of confirming the identity of a principal.
Authentication chaining	A series of authentication modules configured together which a principal must negotiate as configured in order to authenticate successfully.
Authentication level	Positive integer associated with an authentication module, usually used to require success with more stringent authentication measures when requesting resources requiring special protection.
Authentication module	AM authentication unit that handles one way of obtaining and verifying credentials.
Authorization	The act of determining whether to grant or to deny a principal access to a resource.
Authorization Server	In OAuth 2.0, issues access tokens to the client after authenticating a resource owner and confirming that the owner authorizes the client to access the protected resource. AM can play this role in the OAuth 2.0 authorization framework.
Auto-federation	Arrangement to federate a principal's identity automatically based on a common attribute value shared across the principal's profiles at different providers.
Bulk federation	Batch job permanently federating user profiles between a service provider and an identity provider based on a list of matched user identifiers that exist on both providers.
Circle of trust	Group of providers, including at least one identity provider, who have agreed to trust each other to participate in a SAML v2.0 provider federation.
Client	In OAuth 2.0, requests protected web resources on behalf of the resource owner given the owner's authorization. AM can play this role in the OAuth 2.0 authorization framework.
Client-based OAuth 2.0 tokens	After a successful OAuth 2.0 grant flow, AM returns a token to the client. This differs from CTS-based OAuth 2.0 tokens, where AM returns a *reference* to token to the client.
Client-based sessions	AM sessions for which AM returns session state to the client after each request, and require it to be passed in with the subsequent

	request. For browser-based clients, AM sets a cookie in the browser that contains the session information. For browser-based clients, AM sets a cookie in the browser that contains the session state. When the browser transmits the cookie back to AM, AM decodes the session state from the cookie.
Conditions	Defined as part of policies, these determine the circumstances under which which a policy applies. Environmental conditions reflect circumstances like the client IP address, time of day, how the subject authenticated, or the authentication level achieved. Subject conditions reflect characteristics of the subject like whether the subject authenticated, the identity of the subject, or claims in the subject's JWT.
Configuration datastore	LDAP directory service holding AM configuration data.
Cross-domain single sign-on (CDSSO)	AM capability allowing single sign-on across different DNS domains.
CTS-based OAuth 2.0 tokens	After a successful OAuth 2.0 grant flow, AM returns a *reference* to the token to the client, rather than the token itself. This differs from client-based OAuth 2.0 tokens, where AM returns the entire token to the client.
CTS-based sessions	AM sessions that reside in the Core Token Service's token store. CTS-based sessions might also be cached in memory on one or more AM servers. AM tracks these sessions in order to handle events like logout and timeout, to permit session constraints, and to notify applications involved in SSO when a session ends.
Delegation	Granting users administrative privileges with AM.
Entitlement	Decision that defines which resource names can and cannot be accessed for a given identity in the context of a particular application, which actions are allowed and which are denied, and any related advice and attributes.
Extended metadata	Federation configuration information specific to AM.
Extensible Access Control Markup Language (XACML)	Standard, XML-based access control policy language, including a processing model for making authorization decisions based on policies.
Federation	Standardized means for aggregating identities, sharing authentication and authorization data information between trusted providers, and

	allowing principals to access services across different providers without authenticating repeatedly.
Fedlet	Service provider application capable of participating in a circle of trust and allowing federation without installing all of AM on the service provider side; AM lets you create Java Fedlets.
Hot swappable	Refers to configuration properties for which changes can take effect without restarting the container where AM runs.
Identity	Set of data that uniquely describes a person or a thing such as a device or an application.
Identity federation	Linking of a principal's identity across multiple providers.
Identity provider (IdP)	Entity that produces assertions about a principal (such as how and when a principal authenticated, or that the principal's profile has a specified attribute value).
Identity repository	Data store holding user profiles and group information; different identity repositories can be defined for different realms.
Java agent	Java web application installed in a web container that acts as a policy enforcement point, filtering requests to other applications in the container with policies based on application resource URLs.
Metadata	Federation configuration information for a provider.
Policy	Set of rules that define who is granted access to a protected resource when, how, and under what conditions.
Policy agent	Java, web, or custom agent that intercepts requests for resources, directs principals to AM for authentication, and enforces policy decisions from AM.
Policy Administration Point (PAP)	Entity that manages and stores policy definitions.
Policy Decision Point (PDP)	Entity that evaluates access rights and then issues authorization decisions.
Policy Enforcement Point (PEP)	Entity that intercepts a request for a resource and then enforces policy decisions from a PDP.
Policy Information Point (PIP)	Entity that provides extra information, such as user profile attributes that a PDP needs in order to make a decision.
Principal	Represents an entity that has been authenticated (such as a user, a device, or an application), and thus is distinguished from other entities.

	When a Subject successfully authenticates, AM associates the Subject with the Principal.
Privilege	In the context of delegated administration, a set of administrative tasks that can be performed by specified identities in a given realm.
Provider federation	Agreement among providers to participate in a circle of trust.
Realm	AM unit for organizing configuration and identity information. Realms can be used for example when different parts of an organization have different applications and identity stores, and when different organizations use the same AM deployment. Administrators can delegate realm administration. The administrator assigns administrative privileges to users, allowing them to perform administrative tasks within the realm.
Resource	Something a user can access over the network such as a web page. Defined as part of policies, these can include wildcards in order to match multiple actual resources.
Resource owner	In OAuth 2.0, entity who can authorize access to protected web resources, such as an end user.
Resource server	In OAuth 2.0, server hosting protected web resources, capable of handling access tokens to respond to requests for such resources.
Response attributes	Defined as part of policies, these allow AM to return additional information in the form of "attributes" with the response to a policy decision.
Role based access control (RBAC)	Access control that is based on whether a user has been granted a set of permissions (a role).
Security Assertion Markup Language (SAML)	Standard, XML-based language for exchanging authentication and authorization data between identity providers and service providers.
Service provider (SP)	Entity that consumes assertions about a principal (and provides a service that the principal is trying to access).
Authentication Session	The interval while the user or entity is authenticating to AM.
Session	The interval that starts after the user has authenticated and ends when the user logs out, or when their session is terminated. For browser-based clients, AM manages user sessions across one or more applications by setting a session cookie. See also CTS-based sessions and Client-based sessions.

Session high availability	Capability that lets any AM server in a clustered deployment access shared, persistent information about users' sessions from the CTS token store. The user does not need to log in again unless the entire deployment goes down.
Session token	Unique identifier issued by AM after successful authentication. For a CTS-based sessions, the session token is used to track a principal's session.
Single log out (SLO)	Capability allowing a principal to end a session once, thereby ending her session across multiple applications.
Single sign-on (SSO)	Capability allowing a principal to authenticate once and gain access to multiple applications without authenticating again.
Site	Group of AM servers configured the same way, accessed through a load balancer layer. The load balancer handles failover to provide service-level availability. The load balancer can also be used to protect AM services.
Standard metadata	Standard federation configuration information that you can share with other access management software.
Stateless Service	Stateless services do not store any data locally to the service. When the service requires data to perform any action, it requests it from a data store. For example, a stateless authentication service stores session state for logged-in users in a database. This way, any server in the deployment can recover the session from the database and service requests for any user. All AM services are stateless unless otherwise specified. See also Client-based sessions and CTS-based sessions.
Subject	Entity that requests access to a resource When an identity successfully authenticates, AM associates the identity with the Principal that distinguishes it from other identities. An identity can be associated with multiple principals.
Identity store	Data storage service holding principals' profiles; underlying storage can be an LDAP directory service or a custom `IdRepo` implementation.
Web Agent	Native library installed in a web server that acts as a policy enforcement point with policies based on web page URLs.