



Installation Guide

ForgeRock Access Management 5

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

Copyright © 2011-2017 ForgeRock AS.

Abstract

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



This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

To view a copy of this license, visit <https://creativecommons.org/licenses/by-nc-nd/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

ForgeRock® and ForgeRock Identity Platform™ are trademarks of ForgeRock Inc. or its subsidiaries in the U.S. and in other countries. Trademarks are the property of their respective owners.

UNLESS OTHERWISE MUTUALLY AGREED BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

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

DejaVu Fonts

Bitstream Vera Fonts Copyright

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

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

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

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

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

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

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

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

Arev Fonts Copyright

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

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

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

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

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

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

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

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

Admonition graphics by Yannick Lung. Free for commercial use. Available at FreeCnS.Cumulus.

Table of Contents

Preface	iv
1. Preparing for Installation	1
1.1. Preparing the Environment	1
1.2. Preparing a Web Application Container	8
1.3. Downloading and Deploying	20
1.4. Preparing External Data Stores	23
2. Installing and Starting Servers	43
2.1. Installing a Single Server	44
2.2. Installing Multiple Servers	59
2.3. Installing and Using the Tools	64
2.4. Starting Servers	70
3. Implementing the Core Token Service	75
3.1. General Recommendations for CTS Configuration	75
3.2. CTS Deployment Steps	78
3.3. CTS Backups and OpenDJ Replication Purge Delay	89
3.4. Managing CTS Tokens	90
3.5. CTS Tuning Considerations	91
4. Securing Installations	94
4.1. Avoiding Obvious Defaults	94
4.2. Protecting Network Access	95
4.3. Securing Administration	96
4.4. Securing Communications	97
5. Removing Installations	104
6. Troubleshooting Installations	106
7. Reference	107
7.1. Core Token Service (CTS) Object Identifiers	107
7.2. Command-line Tool Reference	119
A. Getting Support	127
A.1. Accessing Documentation Online	127
A.2. Joining the ForgeRock Community	128
A.3. Getting Support and Contacting ForgeRock	128
B. Supported Scripts	129
C. Supported LDIF Files	131
Glossary	134

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

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

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

Chapter 1

Preparing for Installation

This chapter covers prerequisites for installing OpenAM software, including how to prepare your environment, how to set up your application server to run OpenAM, how to prepare directory servers to store configuration data, and how to prepare an identity repository to handle OpenAM identities.

1.1. Preparing the Environment

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

The topics covered in this section are:

- Preparing a Fully Qualified Domain Name
- Preparing a Java Environment
- Setting Maximum File Descriptors

For more information about supported operating systems and Java requirements, see Section 2.2, "Operating System Requirements" and Section 2.3, "Java Requirements" in the *Release Notes*.

1.1.1. Preparing a Fully Qualified Domain Name

OpenAM requires that you provide a fully qualified domain name (FQDN) when you configure it. Before you set up OpenAM, 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 OpenAM, not even for testing purposes. OpenAM 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 OpenAM 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.

1.1.2. Preparing a Java Environment

OpenAM software depends on a Java runtime environment. Check the output of the **java -version** command to make sure your version is supported according to Section 2.3, "Java Requirements" in the *Release Notes*.

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

- The deployment has a dedicated OpenDJ directory 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 OpenAM server is configured to use stateful 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.

1.1.2.1. 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)

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

-XX:PermSize=256m (when using JDK 7)

Set the permanent generation size to 256 MB.

-XX:MaxPermSize=256m (when using JDK 7)

Set the maximum permanent generation size to 256 MB.

-XX:MetaspaceSize=256m (when using JDK 8)

Set the metaspace memory size to 256 MB.

-XX:MaxMetaspaceSize=256m (when using JDK 8)

Set the maximum metaspace memory size to 256 MB.

For additional JVM tuning and security recommendations, see Section 1.1.2.4, "Tuning Java Virtual Machine Settings".

1.1.2.2. 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)

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

1.1.2.3. Settings for OpenJDK Java Environment

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

-Xmx1024m (minimum)

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

-XX:MetaspaceSize=256m

Set the permanent generation size to 256 MB.

1.1.2.4. Tuning Java Virtual Machine Settings

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

Table 1.1. Heap Size Settings

JVM Parameters	Suggested Value	Description
-Xms & -Xmx	At least 1 GB (2 GB with embedded OpenDJ), in production environments at least 2 GB to 3 GB. This setting depends on the available physical memory,	-

JVM Parameters	Suggested Value	Description
	and on whether a 32- or 64-bit JVM is used.	
<code>-server</code>	-	Ensures the server JVM is used
<code>-XX:PermSize</code> & <code>-XX:MaxPermSize</code> (JDK 7)	Set both to 256 MB	Controls the size of the permanent generation in the JVM
<code>-XX:MetaspaceSize</code> & <code>-XX:MaxMetaspaceSize</code> (JDK 8)	Set both to 256 MB	Controls the size of the metaspace in the JVM
<code>-Dsun.net.client.defaultReadTimeout</code>	60000	Controls the read timeout in the Java HTTP client implementation This applies only to the Sun/Oracle HotSpot JVM.
<code>-Dsun.net.client.defaultConnectTimeout</code>	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.

Table 1.2. Security Settings

JVM Parameters	Suggested Value	Description
<code>-Dhttps.protocols</code>	<code>TLSv1,TLSv1.1,TLSv1.2</code>	Controls the protocols used for outbound HTTPS connections from OpenAM. Specify one or more of the following values, separated by commas: <ul style="list-style-type: none"> • TLSv1 • TLSv1.1 • TLSv1.2 This setting applies only to Sun/Oracle Java environments.

JVM Parameters	Suggested Value	Description
<code>-Dorg.forgerock.openam.ldap.secure.protocol.version</code>	-	<p>Controls the protocol OpenAM uses to connect to various external resources.</p> <p>Specify one or more of the following values, separated by commas:</p> <ul style="list-style-type: none"> • TLSv1 • TLSv1.1 • TLSv1.2

Note

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

For `-Dorg.forgerock.openam.ldap.secure.protocol.version`, see Section 4.4, "Securing Communications" for a list of external resources to which communication is affected.

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

Specify a comma-separated list with multiple protocols if OpenAM will use the most secure protocol supported by the external resources. For example, a value of `TLSv1,TLSv1.1,TLSv1.2` configures OpenAM to attempt to use the TLSv1.2 protocol to connect to external configuration and user data stores. If a TLSv1.2 connection is not supported, OpenAM attempts to use TLSv1.1 to connect. If TLSv1.1 is not supported, OpenAM uses TLSv1.

Table 1.3. Garbage Collection Settings

JVM Parameters	Suggested Value	Description
<code>-verbose:gc</code>	-	Verbose garbage collection reporting
<code>-Xloggc:</code>	<code>\$CATALINA_HOME/logs/gc.log</code>	Location of the verbose garbage collection log file
<code>-XX:+PrintClassHistogram</code>	-	Prints a heap histogram when a SIGTERM signal is received by the JVM
<code>-XX:+PrintGCDetails</code>	-	Prints detailed information about garbage collection
<code>-XX:+PrintGCTimeStamps</code>	-	Prints detailed garbage collection timings
<code>-XX:+HeapDumpOnOutOfMemoryError</code>	-	Out of Memory errors generate a heap dump automatically
<code>-XX:HeapDumpPath</code>	<code>\$CATALINA_HOME/logs/heapdump.hprof</code>	Location of the heap dump

JVM Parameters	Suggested Value	Description
<code>-XX:+UseConcMarkSweepGC</code>	-	Use the concurrent mark sweep garbage collector
<code>-XX:+UseCMSCompactAtFullCollection</code>	-	Aggressive compaction at full collection
<code>-XX:+CMSClassUnloadingEnabled</code>	-	Allow class unloading during CMS sweeps

1.1.3. Setting Maximum File Descriptors

If you use the embedded OpenDJ directory, verify that OpenDJ has enough file descriptors set in the operating system to open many files, especially when handling multiple client connections. For example, Linux systems in particular often set a limit of 1024 per user, which is too low for OpenDJ.

In general, do not set up file descriptors to a same value or higher than the maximum number allowed in the system itself. Please consult your operating system documentation for your particular deployment.

OpenDJ should have access to at least 64K (65536) file descriptors. The embedded OpenDJ directory runs inside the OpenAM process space. When running OpenAM as user `openam` on a Linux system that uses `/etc/security/limits.conf` to set user limits, you can set soft and hard limits by adding these lines to the file.

```
openam soft nofile 65536
openam hard nofile 131072
```

You can verify the new soft limit the next time you log in as user `openam` with the `ulimit -n` command.

```
$ ulimit -n
65536
```

You can check the Linux system overall maximum as follows:

```
$ cat /proc/sys/fs/file-max
204252
```

If the overall maximum is too low, you can increase it as follows:

1. As superuser, edit the `/etc/sysctl.conf` file to set the kernel parameter `fs.file-max` to a higher maximum.
2. Run the `sysctl -p` command to reload the settings in `/etc/sysctl.conf`.
3. Open the `/proc/sys/fs/file-max` file again to confirm that it now corresponds to the new maximum.
4. If you are running a daemon process on some Linux systems, you may need to add a `LimitNOFILE` directive. For example, if running Tomcat, under the Services section in `/usr/lib/systemd/system/tomcat.service`, add the line:

```
LimitNOFILE=65536
```

5. To reload the daemon, run:

```
$ systemctl daemon-reload
```

6. Restart Tomcat:

```
$ systemctl start tomcat && journalctl --follow -u tomcat
```

7. Check the file descriptors:

```
$ cat /proc/<tomcat pid>/limit | grep 'open files'
```

Again, consult your operating system documentation for specifics to your deployment.

1.1.4. Enabling RSA SecurID Support

To use the SecurID authentication module, you must first build an OpenAM 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 OpenAM .war file, `OpenAM-14.0.0.war` (`OpenAM-14.0.0.war` if you obtained the file by unpacking `OpenAM-14.0.0.zip`).

```
$ mkdir /tmp/openam
$ cd /tmp/openam/
$ jar -xf ~/Downloads/openam/OpenAM-14.0.0.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.

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

3. Pack up the OpenAM .war file to deploy.

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

4. Deploy the new .war file. See Section 1.3.2, "Deploying".

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

1.2. Preparing a Web Application Container

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

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 Wrap Encryption

Note

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

For a list of supported web application containers, see Section 2.4, "Web Application Container Requirements" in the *Release Notes*.

1.2.1. Preparing Apache Tomcat

OpenAM 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

OpenAM core services require a minimum JVM heap size of 1 GB, and a permanent generation size of 256 MB. If you are including the embedded OpenDJ directory, OpenAM requires at least a 2 GB heap, as 50% of that space is allocated to OpenDJ. See Section 1.1.2, "Preparing a Java Environment" for details.

Set up the `CATALINA_OPTS` 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 OpenAM resources have names that can contain slash characters (/), for example policy names, application names, and SAML v2.0 entities. These slash characters can cause unexpected behavior when running OpenAM on Tomcat.

To workaroud this issue, 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. For example:

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

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 OpenAM 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 [Procedure 2.3, "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 OpenAM 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" />
```

1.2.1.1. Tuning Apache Multi-Processing Modules

Apache 2.0 and later comes with Multi-Processing Modules (MPMs) that extend the basic functionality of a web server to support the wide variety of operating systems and customizations for a particular site.

The key area of performance tuning for Apache is to run in worker mode ensuring that there are enough processes and threads available to service the expected number of client requests. Apache performance is configured in the `conf/extra/httpd-mpm.conf` file.

The key properties in this file are `ThreadsPerChild` and `MaxClients`. Together the properties control the maximum number of concurrent requests that can be processed by Apache. The default configuration allows for 150 concurrent clients spread across 6 processes of 25 threads each.

```
<IfModule mpm_worker_module>
  StartServers      2
  MaxClients        150
  MinSpareThreads   25
  MaxSpareThreads   75
  ThreadsPerChild   25
  MaxRequestsPerChild 0
</IfModule>
```

Important

For the policy agent notification feature, the `MaxSpareThreads`, `ThreadLimit` and `ThreadsPerChild` default values must *not* be altered; otherwise the notification queue listener thread cannot be registered.

Any other values apart from these three in the worker MPM can be customized. For example, it is possible to use a combination of `MaxClients` and `ServerLimit` to achieve a high level of concurrent clients.

1.2.2. Preparing for JBoss and WildFly

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

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

- Procedure 1.1, "To Prepare JBoss or WildFly"
- Procedure 1.2, "To Prepare for JBoss and WildFly"

Procedure 1.1. To Prepare JBoss or WildFly

1. Stop JBoss or WildFly.
2. The default JVM settings do not allocate sufficient memory to OpenAM. 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 OpenDJ, the minimum heap size may be higher. For details on the JVM options to use, see Section 1.1.2, "Preparing a Java Environment".

When using JDK 7, change the permanent generation size to `-XX:MaxPermSize=256m` if the default size does not exceed this amount.

When using JDK 8, 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. 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.

Procedure 1.2. To Prepare for JBoss and WildFly

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

1. If you have not already done so, create a temporary directory and expand the `OpenAM-14.0.0.war` file (`OpenAM-14.0.0.war` if you obtained the file by unpacking `OpenAM-14.0.0.zip`).

```
$ cd /tmp
$ mkdir /tmp/openam ; cd /tmp/openam
$ jar xvf ~/Downloads/OpenAM-14.0.0.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 OpenAM 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 OpenAM on WildFly.

4. Rebuild the `openam.war` file.

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

5. 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 OpenAM server. See Section 2.2.5, "Handling Multiple Cookie Domains When Using Wildfly" for more information.

1.2.3. Preparing Oracle WebLogic

To deploy OpenAM in WebLogic, perform the following steps:

1. Update the JVM options as described in Section 1.1.2, "Preparing a Java Environment".
2. Customize the `OpenAM-14.0.0.war` file as described in Procedure 1.3, "To Prepare for Oracle WebLogic".

The filename is `OpenAM-14.0.0.war` if you obtained the file by unpacking `OpenAM-14.0.0.zip`.

3. If you are using WebLogic 11g or earlier, prepare it as described in Procedure 1.4, "To Prepare Oracle WebLogic 11g or Earlier".

Procedure 1.3. To Prepare for Oracle WebLogic

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

Change the OpenAM `war` file as follows:

1. Create a temporary directory and expand the `OpenAM-14.0.0.war` file (`OpenAM-14.0.0.war` if you obtained the file by unpacking `OpenAM-14.0.0.zip`):

```
$ cd /tmp
$ mkdir /tmp/openam ; cd /tmp/openam
$ jar xvf ~/Downloads/OpenAM-14.0.0.war
```

2. Locate the `bootstrap.properties` file in the `WEB-INF/classes` directory of the expanded `war` file.
3. 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 \.

4. Rebuild the `openam.war` file:


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

Procedure 1.4. To Prepare Oracle WebLogic 11g or Earlier

To prepare Oracle WebLogic 11g or earlier for an OpenAM deployment, edit the WebLogic domain configuration to allow basic authentication credentials to be passed back to OpenAM. By default, WebLogic attempts to resolve authentication credentials itself. When you change the WebLogic domain configuration, you ensure that the OpenAM OAuth 2.0 providers receive basic authentication credentials for OAuth 2.0 grants that rely on basic authentication.

Note

WebLogic uses its own classes if a class exists in both the parent and child classloaders by default. To map resources defined for OpenAM, OpenAM bundles its own WebLogic deployment descriptor file `weblogic.xml` and automatically places it in the `/WEB-INF` directory automatically. The descriptor file works for WebLogic 11g and 12c deployments.

To edit the WebLogic domain configuration:

1. Stop the WebLogic server.
2. Edit the WebLogic domain configuration, `/path/to/wlsdomain/config/config.xml`, setting `<enforce-valid-basic-auth-credentials>` to `false` in the `<security-configuration>` element.

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

3. When deploying OpenAM on WebLogic 11g (version 10.3.x), use the SOAP with Attachments API for Java (SAAJ) implementation from the Java Runtime Environment, rather than the WebLogic implementation. The WebLogic implementation can cause OpenAM to throw exceptions with the message `java.lang.UnsupportedOperationException: This class does not support SAAJ 1.1`, and to fail to authenticate users in some cases.
4. To use the Oracle Java SAAJ implementation, edit the WebLogic startup script for the domain where OpenAM runs, such as `/path/to/weblogic/user_projects/domains/wlsdomain/startWebLogic.sh`.

Change the following line:

```
${DOMAIN_HOME}/bin/startWebLogic.sh $*
```

To set the `javax.xml.soap.MessageFactory` property.

```
${DOMAIN_HOME}/bin/startWebLogic.sh \
-Djavax.xml.soap.MessageFactory=\
com.sun.xml.internal.messaging.saaj.soap.ver1_1.SOAPMessageFactory1_1Impl $*
```

5. Start the WebLogic server.

1.2.4. Preparing IBM WebSphere

Before you deploy OpenAM, use the Administrator console to update JVM options as described in Section 1.1.2, "Preparing a Java Environment".

In addition, configure WebSphere to load classes from OpenAM bundled libraries before loading classes from libraries delivered with WebSphere. The following steps must be completed after you deploy OpenAM into WebSphere.

1. In WebSphere administration console, browse to Application > Application Type > WebSphere enterprise applications > *OpenAM Name* > Class loading and update detection.
2. Set Class loader order > Classes loaded with local class loader first (parent last).
3. 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**.
4. Save your work.

1.2.5. Enabling CORS Support

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

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

Procedure 1.5. To Enable CORS Support

1. If you have not yet deployed OpenAM:
 - a. Unpack the OpenAM .war file, **OpenAM-14.0.0.war** (**OpenAM-14.0.0.war** if you obtained the file by unpacking **OpenAM-14.0.0.zip**).

```
$ mkdir /tmp/openam
$ cd /tmp/openam/
$ jar -xf ~/Downloads/openam/OpenAM-14.0.0.war
```
 - b. Open the deployment descriptor file **WEB-INF/web.xml** in a text editor.
2. If you have already deployed OpenAM:
 - 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 OpenAM 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 that 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 OpenAM.

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

origins (required)

A comma-separated list of the origins allowed when making CORS requests to OpenAM. 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 OpenAM.

Example: `iPlanetDirectoryPro,Accept-API-Version`

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.

Common headers used when accessing an OpenAM server includes the following:

Table 1.4. Common Headers Used

Header	Information
<code>iPlanetDirectoryPro</code>	Used for session information. See Section 6.1, "Implementing Session State" in the <i>Authentication and Single Sign-On Guide</i> .
<code>X-OpenAM-Username,X-OpenAM-Password</code>	Used to pass credentials in REST calls that use the HTTP POST method.

Header	Information
	See Section A.5, "Authentication and Logout" in the <i>Authentication and Single Sign-On Guide</i> .
Accept-API-Version	Used to request a specific OpenAM endpoint version. See Section A.3, "REST API Versioning" in the <i>Authentication and Single Sign-On Guide</i> .
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 set. See Procedure 2.8, "To Update an UMA Resource Set" in the <i>User-Managed Access (UMA) Guide</i> .

Caution

Do not add Content-Type to the list of allowed headers in the CORS filter. Doing so can expose OpenAM to cross-site request forgery (CSRF) attacks.

expectedHostname (optional)

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

If not specified, any host value is accepted.

If the OpenAM 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 the OpenAM 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>
</filter>
```

```
<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 OpenAM:
 - a. Pack up the OpenAM `.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 Section 1.3.2, "Deploying".

7. If you have already deployed OpenAM:
 - Restart OpenAM or the web container where it runs.

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

1.2.6. Preparing AES Wrap Encryption

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

If your deployment requires a more secure encryption algorithm, OpenAM supports the Advanced Encryption Standard (AES) Key Wrap algorithm (RFC3394). OpenAM'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 Wrap Encryption algorithm is only enabled when installing OpenAM. There is no current upgrade path for existing installations.

Several OpenAM components, such as agents and the SOAP Security Token Service, require JCE encryption and decryption. Because a web container cannot be configured to support both JCE and AES Key Wrap

encryption, you must make sure not to deploy any OpenAM components that require JCE encryption on OpenAM servers that run on web containers configured for AES Key Wrap encryption.

Procedure 1.6. 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=20000"
JAVA_OPTS="$JAVA_OPTS -Dorg.forgerock.openam.encryption.key.size=256"
JAVA_OPTS="$JAVA_OPTS -Dorg.forgerock.openam.encryption.key.digest=SHA512"
```

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.

1.3. Downloading and Deploying

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

The topics covered in this section are:

- Obtaining Software
- Deploying

1.3.1. Obtaining Software

The *ForgeRock BackStage* website hosts downloadable versions of OpenAM, including a `.zip` file with all of the OpenAM components, the `.war` file, OpenAM tools, the configurator, policy agents, OpenIG, and documentation. Verify that you review the Software License and Subscription Agreement presented before you download OpenAM files.

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

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

```
$ cd ~/Downloads
$ mkdir openam ; cd openam
$ unzip ~/Downloads/OpenAM-14.0.0.zip
```


Note

The platform version number that appears in the download file name may differ from the internal version number. The internal version number for this release is `${softwareVersion}`.

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

Table 1.5. Distribution Files

File	Description
<code>OpenAM-\${softwareVersion}.war</code>	The distribution <code>.war</code> file includes the core server code with an embedded OpenDJ directory server, which stores configuration data and simplifies deployments. The distribution includes an administrative graphical user interface (GUI) Web console. During installation, the <code>.war</code> 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 <code>boot.json</code> file in the OpenAM installation directory, from environment variables, or from a combination of the two. This is identical to the <code>AM-\${platform.long.version}.war</code> file provided for download.
<code>ClientSDK-13.5.0-1.jar</code>	OpenAM provides a client SDK for developers to code extensions for their web applications to access OpenAM's services. The client SDK contains the Java packages, classes, property files, and sample code to help you develop your code.
<code>ExampleClientSDK-CLI-13.5.0-1.zip</code>	OpenAM provides client SDK examples to help you run them on OpenAM. The <code>zip</code> distribution file contains setup scripts and samples that you can run to learn how to use the client SDK.
<code>ExampleClientSDK-WAR-13.5.0-1.war</code>	The example client SDK also comes in a <code>.war</code> file, which installs on your container.
<code>Fedlet-\${softwareVersion}.zip</code>	OpenAM provides an OpenAM 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.
<code>IDPDiscovery-\${softwareVersion}.war</code>	OpenAM provides an IdP Discovery Profile (SAMLv2 binding profile) for its IdP Discovery service. The profile keeps track of the identity providers for each user.
<code>OpenAM-Soap-STS-Server-\${softwareVersion}.war</code>	OpenAM provides a SOAP-based security token service (STS) server that issues tokens based on the WS-Security protocol ^a .
<code>SSOAdminTools-\${softwareVersion}.zip</code>	OpenAM provides an <code>ssoadm</code> command-line tool that allows administrators to configure and maintain OpenAM as well as create their own configuration scripts. The <code>zip</code> distribution file contains binaries, properties file, script templates, and setup scripts for UNIX and windows servers.
<code>SSOConfiguratorTools-\${softwareVersion}.zip</code>	OpenAM provides configuration and upgrade tools for installing and maintaining your server. The <code>zip</code> distribution file contains libraries, legal notices, and supported binaries for these configuration tools. Also, you can

File	Description
	view example configuration and upgrade properties files that can be used as a template for your deployments.

^aAM also provides REST-based STS service endpoints, which you can directly utilize on the AM server.

1.3.2. Deploying

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

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

Procedure 1.7. To Deploy an Instance

The `OpenAM-14.0.0.war` file (`OpenAM-14.0.0.war` if you obtained the file by unpacking `OpenAM-14.0.0.zip`) contains OpenAM 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 OpenAM-14.0.0.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 OpenAM in a production environment, do not use a predictable deployment URI such as `/openam` or `/opensso`.

Note

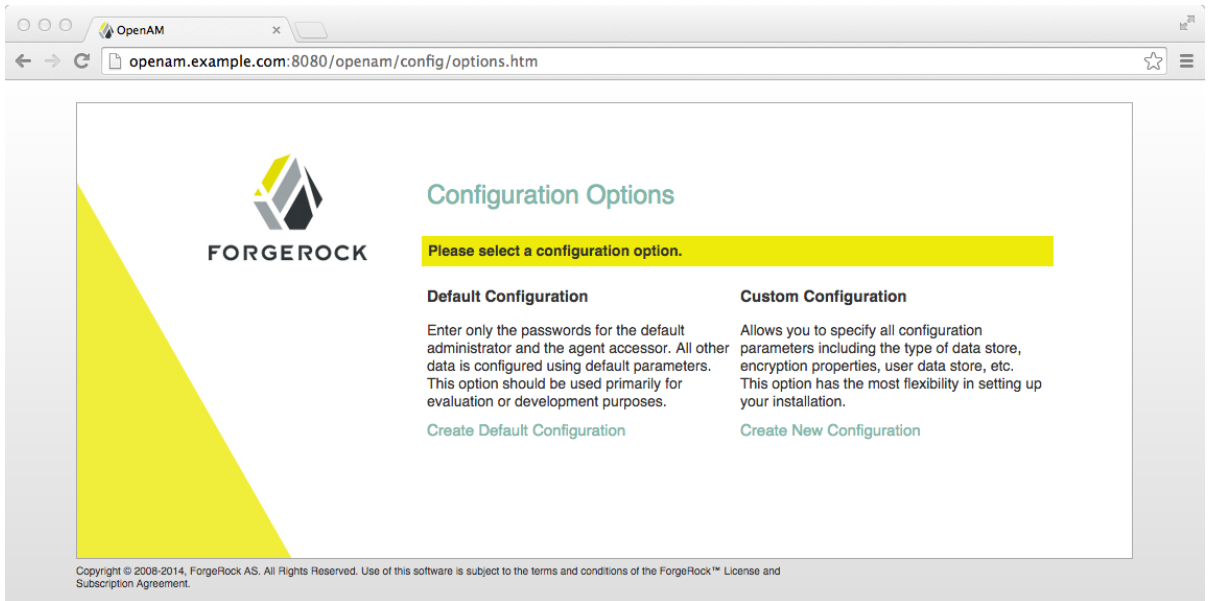
You change the file name to something other than `openam.war` when deploying in Tomcat so that the deployment URI is not `/openam`. For helpful hints on avoiding obvious deployment defaults, see Section 4.1, "Avoiding Obvious Defaults".

Important

To properly configure OpenAM, OpenAM requires a deployment URI with a non-empty string after `/`. Do not deploy OpenAM at the root context. Do not rename the `.war` file to `R00T.war` before deploying on Tomcat, for example.

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

2. Navigate to the initial configuration screen, for example `http://openam.example.com:8080/openam`.



OpenAM is now ready for installation. to proceed, take on of the following actions:

- Configure external data stores using the files created during deployment. See Section 1.4, "Preparing External Data Stores".
- Use the embedded data stores for evaluation purposes, and skip ahead to installing OpenAM. See Section 2.1, "Installing a Single Server".

1.4. Preparing External Data Stores

This section covers setting up external data stores for configuration or identity data.

OpenAM includes embedded data stores for configuration and identity data that can be used for evaluation and testing purposes. In production environments, external data stores are preferred.

The topics covered in this section are:

- Preparing an External Identity Repository
- Preparing an External Configuration Data Store

For a list of supported data stores, see Section 2.5, "Data Store Requirements" in the *Release Notes*.

1.4.1. Preparing an External Identity Repository

OpenAM accesses user identity data from one or more identity repositories. OpenAM ships with an embedded OpenDJ directory server that you can install as part of the OpenAM configuration process.

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

If you are configuring OpenAM to share data with other applications, or if you expect your deployment will have a large amount of users, connect OpenAM to an external identity repository. For a list of supported external identity repositories, see Section 2.5, "Data Store Requirements" in the *Release Notes*.

Important

Storing identity data in a relational database is an Early Access feature, meaning, it is not generally recommended for use in production environments. This section covers preparing directory servers as external identity repositories only. For more information about storing identity data in a relational database, see Section 10.1.3, "Database Repository (Early Access) Configuration Properties" in the *Setup and Maintenance Guide*.

1.4.1.1. Important Considerations for Using External Identity Repositories

OpenAM connects to an external directory by binding to it as a user that you specify in the OpenAM data store configuration. This user is known as the *OpenAM data store administrator*.

Specifying the directory administrator, for example, `cn=Directory Manager` as the OpenAM data store administrator is not recommended for production deployments as it will give OpenAM directory administrator privileges to the identity repository.

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

You need to consider two areas of privileges for the OpenAM data store administrator:

Schema Update Privileges

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

Directory Read and Write Access Privileges

If you want OpenAM to create, update, and delete user entries, then the OpenAM data store administrator 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 OpenAM data store administrator needs read privileges only.

The level of access privileges you give the OpenAM data store administrator is specific to each OpenAM deployment. Work with your directory server administrator to determine the appropriate level of privileges as part of the process of preparing an external identity repository.

1.4.1.2. Preparing Your External Identity Repository

The steps for preparing an external identity repository vary depending on the schema update privileges given to the OpenAM data store administrator.

- If the OpenAM data store administrator has schema update privileges, follow the procedure in Section 1.4.1.2.1, "Preparing an Identity Repository With Dynamic Schema Updates".
- If the OpenAM data store administrator does not have schema update privileges, follow the procedure in Section 1.4.1.2.2, "Preparing an Identity Repository With Manual Schema Updates".

After you have completed one of these two procedures, continue by configuring your external identity repository as an OpenAM data store as described in Section 1.4.1.3, "Configuring Data Stores That Access External Identity Repositories".

Note

Example commands throughout this section use default values for user IDs and port numbers. When running similar commands, be sure to use appropriate values for your directory server.

When running the **ldapmodify** command, you might need to specify the **--trustAll** argument to trust server certificates if your directory server uses self-signed certificates and StartTLS or SSL.

1.4.1.2.1. Preparing an Identity Repository With Dynamic Schema Updates

If the OpenAM data store administrator has schema update privileges, you can configure the OpenAM data store using dynamic schema updates. With dynamic schema updates, OpenAM automatically updates the directory server schema of the external identity repository as needed. Schema updates might occur when you configure a data store as part of initial OpenAM configuration, when you configure a data store after initial OpenAM configuration, or when you upgrade OpenAM.

The following procedure shows you how to prepare an identity repository with dynamic schema updates. The procedure assumes that you have already created an OpenDJ identity repository and populated it with user data. The instructions that follow do not include steps to install OpenDJ, configure directory server backends, and implement replication. For external identity repositories other than OpenDJ, you must perform tasks that are analogous to the ones in the example procedure. Consult the documentation for your directory server software to determine the appropriate actions to take.

Procedure 1.8. To Prepare an External OpenDJ Identity Repository with Dynamic Schema Updates

1. Create the OpenAM data store administrator account.

This example uses `uid=openam,ou=admins,dc=example,dc=com` as the OpenAM data store administrator. It is assumed that the `dc=example,dc=com` suffix already exists in the directory.

First, create an LDIF file that defines the OpenAM data store administrator account and gives the account the following privileges:

- `update-schema`. Allows the account to update the directory schema.
- `subentry-write`. Allows the account to make directory subentry updates.
- `password-reset`. Allows the account to reset other users' passwords. Required for the OpenAM forgotten password feature. This privilege is not required for deployments where the OpenAM data store will not modify user entries.

```
dn: ou=admins,dc=example,dc=com
objectClass: top
objectClass: organizationalunit
ou: OpenAM Administrator

dn: uid=openam,ou=admins,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: OpenAM Administrator
sn: OpenAM
userPassword: changeMe
ds-privilege-name: update-schema
ds-privilege-name: subentry-write
ds-privilege-name: password-reset
```

Then, run the **ldapmodify** command to create the user. The following example assumes that you are using OpenDJ 4.0 and later:

```
$ ldapmodify \
  --hostname opendj.example.com \
  --port 1389 \
  --bindDN "cn=Directory Manager" \
  --bindPassword password \
  openam-ds-admin-account.ldif

Processing ADD request for ou=admins,dc=example,dc=com
ADD operation successful for DN ou=admins,dc=example,dc=com
Processing ADD request for uid=openam,ou=admins,dc=example,dc=com
ADD operation successful for DN uid=openam,ou=admins,dc=example,dc=com
```

2. Add a global ACI that lets the OpenAM administrator account modify the directory schema.

```
$ dsconfig set-access-control-handler-prop \
--hostname opendj.example.com \
--port 4444 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--no-prompt \
--add \
'global-aci:(target="ldap:///cn=schema")(targetattr="attributeTypes||objectClasses")
(version 3.0; acl "Modify schema"; allow (write)
userdn="ldap:///uid=openam,ou=admins,dc=example,dc=com");'
```

If you copy the text from the preceding example, make sure that the value starting with `'global-aci` is all on a single line.

To verify that you have added the global ACI correctly, list the global ACIs.

```
$ dsconfig get-access-control-handler-prop \
--port 4444 \
--hostname opendj.example.com \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--property global-aci
```

The global ACI that allows the OpenAM administrator account to modify schema definitions should appear in the list of global ACIs:

```
"(target="ldap:///cn=schema")(targetattr="attributeTypes||
objectClasses") (version 3.0; acl "Modify schema"; allow
(write) userdn="ldap:///uid=openam,ou=admins,dc=example,dc=com");"
```

3. Allow OpenAM to read the directory schema. OpenAM needs to read the directory schema to ensure that changes made to identities stored in identity repositories remain compliant with the directory schema.

For OpenDJ, no actions are required. Simply retain the default "User-Visible Schema Operational Attributes" global ACI.

4. Give the OpenAM data store administrator appropriate access rights on the directory. When OpenAM connects to an external identity repository, it binds as the OpenAM data store administrator.

For deployments in which OpenAM will read and write user entries, the OpenAM data store administrator needs privileges to create, modify, delete, search, read, and perform persistent searches on user entries in the directory. For deployments in which OpenAM only reads user entries, the OpenAM data store administrator needs privileges to only read, search, and perform persistent searches on user entries in the directory.

To grant the OpenAM data store administrator account privileges to read and write user entries in OpenDJ, create a file with the following LDIF:

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="* || aci")(version 3.0;acl "Allow identity modification";
  allow (write)(userdn = "ldap:///uid=openam,ou=admins,dc=example,dc=com");)
aci: (targetattr!="userPassword||authPassword")(version 3.0;
  acl "Allow identity search"; allow (search, read)
  (userdn = "ldap:///uid=openam,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=openam,ou=admins,dc=example,dc=com");)
aci: (version 3.0;acl "Add or delete identities"; allow (add, delete)
  (userdn = "ldap:///uid=openam,ou=admins,dc=example,dc=com");)
```

To grant the OpenAM data store administrator account privileges to read (but not write) user entries in OpenDJ, create a file with the following LDIF:

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr!="userPassword||authPassword")(version 3.0;
  acl "Allow identity search"; allow (search, read)
  (userdn = "ldap:///uid=openam,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=openam,ou=admins,dc=example,dc=com");)
```

Then, run the **ldapmodify** command to implement the ACIs. The following example assumes that you are using OpenDJ 4.0 and later:

```
$ ldapmodify \
--hostname opendj.example.com \
--port 1389 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
add-acis-for-openam-ds-admin-access.ldif

Processing MODIFY request for dc=example,dc=com
MODIFY operation successful for DN dc=example,dc=com
```

Continue by configuring your external identity repository as an OpenAM data store as described in Section 1.4.1.3, "Configuring Data Stores That Access External Identity Repositories".

1.4.1.2.2. Preparing an Identity Repository With Manual Schema Updates

If the OpenAM data store administrator does not have schema update privileges, you must configure the OpenAM data store by using manual schema updates. To do this, update the directory server schema of the external identity repository manually before you configure a data store as part of initial OpenAM configuration, before you configure a data store after initial OpenAM configuration, and whenever you upgrade OpenAM.

The following procedure shows you how to prepare an identity repository with manual schema updates. The procedure assumes that you have already created an OpenDJ identity repository and

populated it with user data. It therefore does not include steps to install OpenDJ, configure directory server backends, and implement replication. For external identity repositories other than OpenDJ, you must perform tasks that are analogous to the ones in the example procedure. Consult the documentation for your directory server software to determine the appropriate actions to take.

Procedure 1.9. To Prepare an External OpenDJ Identity Repository With Manual Schema Updates

1. Create the OpenAM data store administrator account.

This example uses `uid=openam,ou=admins,dc=example,dc=com` as the OpenAM data store administrator. It is assumed that the `dc=example,dc=com` suffix already exists in the directory.

First, create an LDIF file that defines the OpenAM data store administrator account and gives the account the following privilege:

- **password-reset**. Allows the account to reset other users' passwords. Required for the OpenAM forgotten password feature. For deployments in which OpenAM will not modify user entries, the OpenAM data store administrator does not require this privilege.

```
dn: ou=admins,dc=example,dc=com
objectClass: top
objectClass: organizationalunit
ou: OpenAM Administrator

dn: uid=openam,ou=admins,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: OpenAM Administrator
sn: OpenAM
userPassword: changeMe
ds-privilege-name: password-reset
```

Then, run the **ldapmodify** command to create the user. The following example assumes that you are using OpenDJ 4.0 and later:

```
$ ldapmodify \
--hostname opendj.example.com \
--port 1389 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
openam-ds-admin-account.ldif

Processing ADD request for ou=admins,dc=example,dc=com
ADD operation successful for DN ou=admins,dc=example,dc=com
Processing ADD request for uid=openam,ou=admins,dc=example,dc=com
ADD operation successful for DN uid=openam,ou=admins,dc=example,dc=com
```

2. Using the directory administrator account, add the OpenAM schema extensions to your external identity repository.

First, identify the path that contains LDIF file for OpenAM schema extensions. 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 OpenDJ and Oracle Unified Directory
- `tivoli` for IBM Tivoli Directory Server

Run the **ldapmodify** command to import the user, device print, and dashboard schema extensions. (For more information on the supported LDIF files, see [Appendix C, "Supported LDIF Files"](#).) For example, to add schema extensions for an OpenDJ directory server, run the following **ldapmodify** commands. The following examples assume that you are using OpenDJ 4.0 and later:

```
$ cd /path/to/openam/WEB-INF/template/ldif/opendj

$ ldapmodify \
--hostname opendj.example.com \
--port 1389 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
opendj_user_schema.ldif

$ ldapmodify \
--hostname opendj.example.com \
--port 1389 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
opendj_deviceprint.ldif

$ ldapmodify \
--hostname opendj.example.com \
--port 1389 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
opendj_dashboard.ldif
```

3. Allow OpenAM to read the directory schema. OpenAM needs to read the directory schema to ensure that changes made to identities stored in identity repositories remain compliant with the directory schema.

For OpenDJ, no actions are required. Simply retain the default User-Visible Schema Operational Attributes global ACI.

4. Give the OpenAM data store administrator appropriate access rights on the directory. When OpenAM connects to an external identity repository, it binds as the OpenAM data store administrator.

For deployments in which OpenAM will read and write user entries, the OpenAM data store administrator needs privileges to create, modify, delete, search, read, and perform persistent searches on user entries in the directory. For deployments in which OpenAM only reads user entries, the OpenAM data store administrator needs privileges to only read, search, and perform persistent searches on user entries in the directory.

To grant the OpenAM data store administrator account privileges to read and write user entries in OpenDJ, create a file with the following LDIF:

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="* || aci")(version 3.0;acl "Allow identity modification";
  allow (write)(userdn = "ldap:///uid=openam,ou=admins,dc=example,dc=com");)
aci: (targetattr!="userPassword|authPassword")(version 3.0;
  acl "Allow identity search"; allow (search, read)
  (userdn = "ldap:///uid=openam,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=openam,ou=admins,dc=example,dc=com");)
aci: (version 3.0;acl "Add or delete identities"; allow (add, delete)
  (userdn = "ldap:///uid=openam,ou=admins,dc=example,dc=com");)
```

To grant the OpenAM data store administrator account privileges to read (but not write) user entries in OpenDJ, create a file with the following LDIF:

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr!="userPassword|authPassword")(version 3.0;
  acl "Allow identity search"; allow (search, read)
  (userdn = "ldap:///uid=openam,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=openam,ou=admins,dc=example,dc=com");)
```

Run the **ldapmodify** command to implement the ACIs:

```
$ ldapmodify \
--hostname opendj.example.com \
--port 1389 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
add-acis-for-openam-ds-admin-access.ldif

Processing MODIFY request for dc=example,dc=com
MODIFY operation successful for DN dc=example,dc=com
```

1.4.1.3. Configuring Data Stores That Access External Identity Repositories

Now that you have prepared your external identity repository, you can configure the directory as an OpenAM data store by using one of the following methods:

- By specifying your user directory in the User Data Store Settings dialog box when installing OpenAM core services.

If you are using dynamic schema updates, the OpenAM configurator loads required schema definitions into your user directory. If you are using manual schema updates, you already loaded the required schema definitions into your user directory.

For more information about running the OpenAM configurator, see [Section 2.1, "Installing a Single Server"](#).

- By defining a data store after you have installed OpenAM core services.

If you are using dynamic schema updates and you specify the Load schema when finished option, OpenAM loads required schema definitions into your user directory. If you are using manual schema updates, you will have already loaded the required schema definitions into your user directory.

For more information about defining OpenAM data stores, see [Chapter 3, "Setting Up Identity Data Stores"](#) in the *Setup and Maintenance Guide*.

1.4.1.4. Indexing External Identity Repositories Attributes

After you have configured a data store to access an external identity repository, you must complete identity repository preparation by indexing several attributes.

Procedure 1.10. To Index External Identity Repository Attributes

- Create equality indexes for the `iplanet-am-user-federation-info-key` and `sun-fm-saml2-nameid-infokey` attributes. To create the indexes, run the **dsconfig** command twice. Bind to your user directory as the directory administrator.

The **dsconfig** subcommand used to create the index depends on the version of OpenDJ directory server.

- Use the following commands with OpenDJ 2.6:

```
$ dsconfig \
create-local-db-index \
--port 4444 \
--hostname opendj.example.com \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--backend-name userRoot \
--index-name iplanet-am-user-federation-info-key \
--set index-type:equality \
--no-prompt

$ dsconfig \
create-local-db-index \
--port 4444 \
--hostname opendj.example.com \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--backend-name userRoot \
--index-name sun-fm-saml2-nameid-infokey \
--set index-type:equality \
--no-prompt
```

- Use the following commands with OpenDJ 3 and later:

```
$ dsconfig \
create-backend-index \
--port 4444 \
--hostname opendj.example.com \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--backend-name userRoot \
--index-name iplanet-am-user-federation-info-key \
--set index-type:equality \
--no-prompt

$ dsconfig \
create-backend-index \
--port 4444 \
--hostname opendj.example.com \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--backend-name userRoot \
--index-name sun-fm-saml2-nameid-infokey \
--set index-type:equality \
--no-prompt
```

1.4.1.5. Testing External Identity Repository Access

Prior to working actively with external identity repositories, you should verify that you have configured the repository and administrator privileges correctly. You can test configuration as follows:

- Attempt to create an OpenAM user from the Realms > *Realm Name* > Subjects tab in the AM console. Run this test only if you have given the OpenAM data store administrator write privileges to your identity repository.

- Attempt to access an OpenAM user from the Realms > *RealM Name* > Subjects tab in the AM console.

If you receive an LDAP error code 65 while attempting to create a user, it indicates that you did not correctly prepare the external identity repository. Error code 65 is an LDAP object class violation and often indicates a problem with the directory schema. Common reasons for this error while attempting to create a user include the following:

- If you configured the external data store after initial configuration, you might have simply forgotten to check the "Load schema when finished" option. In this case, select this option and resave the data store configuration.
- The OpenAM administrator account might not have adequate rights to update the directory schema. Review the OpenDJ **access** log and locate the log records for the schema update operation to determine OpenDJ's access rights.

1.4.2. Preparing an External Configuration Data Store

OpenAM stores its configuration in an LDAP directory server. OpenAM ships with an embedded OpenDJ directory server that you can install as part of the OpenAM configuration process. By default, OpenAM installs the embedded directory server and its configuration settings in the **\$HOME** directory of the user running OpenAM and runs the embedded directory server in the same JVM memory space as OpenAM.

OpenAM connects to the embedded OpenDJ directory as directory superuser, bypassing access control evaluation because OpenAM manages the directory as its private store. Be aware that you cannot configure directory failover and replication when using the embedded store.

When OpenAM starts up, it requires the password of the **cn="Directory Manager"** user to unlock the configuration data store. This password is stored in OpenAM's JCEKS keystore as the **configstorepwd** password-protected string alias, and it must be updated every time the **cn="Directory Manager"** user's password changes. For more information about the **configstorepwd** alias, see Section 5.4, "Configuring Password String Aliases" in the *Setup and Maintenance Guide*.

By default, OpenAM also stores data managed by the Core Token Service (CTS) pertaining to user logins—OpenAM stateful sessions, logout blacklists, and several types of authentication tokens—in the same embedded OpenDJ directory that holds the OpenAM configuration. You can choose to create a separate directory store for CTS data. For information about creating a separate directory store for CTS data, see the chapter, *Chapter 3, "Implementing the Core Token Service"*.

Before deploying OpenAM in production, measure the impact of using the embedded directory not only for relatively static configuration data, but also for volatile session and token data. Your tests should subject OpenAM to the same load patterns you expect in production. If it looks like a better choice to use an external directory server, then deploy OpenAM with an external configuration store.

Tip

If you are the directory administrator and do not yet know directory servers very well, take some time to read the documentation for your directory server, especially the sections covering directory schema and procedures on how to configure access to directory data.

Procedure 1.11. To Install an External OpenDJ Directory Server

The following example procedure shows how to prepare a single OpenDJ directory server instance as an external configuration data store. The OpenDJ instance implements a single backend for the OpenAM configuration data. The procedure assumes that you have also prepared an external identity repository and an external CTS store, separate from the configuration data store.

Note

Example commands throughout this section use example values for user IDs and port numbers. When running similar commands, be sure to use appropriate values for your directory server.

When running the **ldapmodify** or **dsconfig** commands, you might need to specify the **--trustAll** argument to trust server certificates if your directory server uses self-signed certificates and StartTLS or SSL.

1. Prepare your OpenDJ installation, then download the OpenDJ software. See the OpenDJ documentation about [Installing OpenDJ Servers](#).

```
$ cd /path/to/openssl
$ ./setup --cli
```

Example options are as follows:

Table 1.6. Example OpenDJ Setup Parameters

Parameter	Example Inputs
Accept License	Yes
Root User DN	cn=Directory Manager
Root User DN Password	(arbitrary)
Fully Qualified Domain Name	opendj.example.com
LDAP Port	1389
Administration Connector Port	4444
Create Base DN	No. This will be created in a later step.
Enable SSL	If you choose this option, make sure that OpenAM can trust the OpenDJ certificate.
Enable TLS	If you choose this option, make sure that OpenAM can trust the OpenDJ certificate.
Start Server After Config	Yes

2. Change to the OpenDJ directory.

```
$ cd /path/to/opendj
```

3. Create a directory server backend, and call it `cfgStore`.

The **dsconfig** command used to create the backend depends on the version of OpenDJ directory server.

- Use the following command with OpenDJ 2.6:

```
$ dsconfig create-backend \
--backend-name cfgStore \
--set base-dn:dc=example,dc=com \
--set enabled:true \
--type local-db \
--port 4444 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--no-prompt
```

- Use the following command with OpenDJ 3.0 and later to create a backend:

```
$ dsconfig create-backend \
--backend-name cfgStore \
--set base-dn:dc=example,dc=com \
--set enabled:true \
--type je \
--port 4444 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--no-prompt
```

4. Create an LDIF file to add the initial entries for the configuration store, and save the file as `add-config-entries.ldif`. The entries include the base DN suffix, an organizational unit entry, and the OpenAM user entry needed to access the directory server.

When OpenAM connects as `uid=openam,ou=admins,dc=example,dc=com` to an external directory server to store its data, it requires read, write, persistent search, and server-side sorting access privileges. You add these privileges by setting access control instructions (ACIs) on the base distinguished name (DN) entry (`dc=example,dc=com`). If your OpenAM user has a DN other than `uid=openam,ou=admins,dc=example,dc=com`, adjust the ACIs where appropriate.

You must also give privileges to the OpenAM user to modify the schema and write to subentries, such as the schema entry. To grant these privileges, you include the following attributes on the OpenAM user entry: `ds-privilege-name: subentry-write` and `ds-privilege-name: update-schema`.


```
dn: dc=example,dc=com
objectclass: top
objectclass: domain
dc: example
aci: (targetattr="*)(version 3.0;acl "Allow CRUDQ operations";
  allow (search, read, write, add, delete)
  (userdn = "ldap:///uid=openam,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=openam,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=openam,ou=admins,dc=example,dc=com");)

dn: ou=admins,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: admins

dn: uid=openam,ou=admins,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: openam
sn: openam
uid: openam
userPassword: secret12
ds-privilege-name: subentry-write
ds-privilege-name: update-schema
```

5. Add the initial entries LDIF file using the **ldapmodify** command. The following example assumes that you are using OpenDJ 4.0 and later:

If you are having trouble with the preceding LDIF file, consider removing the line feeds for the ACI attributes and let it wrap to the next line. If you are still having trouble using the **ldapmodify** command, you can use the **import-ldif** command, although you may have to re-apply the **targetcontrol** ACI attribute.

```
$ bin/ldapmodify \
  --port 1389 \
  --bindDN "cn=Directory Manager" \
  --bindPassword password \
  add-config-entries.ldif
```

6. Add the Global Access Control Instruction (ACI) to the access control handler. The Global ACI gives OpenAM the privileges to modify the schema definitions for the custom configuration where the OpenAM entry has DN **uid=openam,ou=admins,dc=example,dc=com**.

Note

These access rights are only required during configuration, and only if the directory administrator does not add the OpenAM directory schema definitions manually.

If you copy the text from the following example, make sure that the value of `global-aci` is all on a single line.

```
$ bin/dsconfig set-access-control-handler-prop \
--add global-aci:'(target = "ldap:///cn=schema")(targetattr = "attributeTypes ||
objectClasses")(version 3.0; acl "Modify schema"; allow (write)
(userdn = "ldap:///uid=openam,ou=admins,dc=example,dc=com");)'
--port 4444 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--no-prompt
```

7. At this point, deploy the OpenAM server if you have not done so already. For additional details on deploying OpenAM, see Section 1.3.2, "Deploying".
8. OpenAM requires additional schema definitions for attributes used to search for user and configuration data:

Table 1.7. Configuration Data Store Attributes

Attribute	Index Type	Description
CTS attributes		Specifies the CTS attributes required for stateful session high availability and persistence. Located in the <code>WEB-INF/template/ldif/sfha/cts-add-schema.ldif</code> file.
<code>iplanet-am-user-federation-info-key</code>	equality	Specifies a configuration setting to store an account's federation information key, which is used internally. Located in <code>WEB-INF/template/ldif/opendj/opendj_user_schema.ldif</code> file.
<code>sun-fm-saml2-nameid-infokey</code>	equality	Specifies an information key common to an IdP and SP to link accounts. Located in <code>WEB-INF/template/ldif/opendj/opendj_user_schema.ldif</code> file.
<code>sunxmlkeyvalue</code>	equality, substring	Stores configuration values that may be looked up through searches. Located in <code>WEB-INF/template/ldif/opendj/opendj_config_schema.ldif</code> .

Add the required CTS schema definitions. You can find the CTS schema definitions at `/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/sfha/cts-add-schema.ldif`.

```
$ cp /path/to/tomcat/webapps/openam/WEB-INF/template/ldif/sfha/cts-add-schema.ldif /tmp
```

9. Add the schema file to the directory server.

```
$ bin/ldapmodify \
--port 1389 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
/tmp/cts-add-schema.ldif
```

10. Add the required user store schema definitions. You can find the schema definitions at [/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opensj/opensj_user_schema.ldif](#).

```
$ cp /path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opensj/opensj_user_schema.ldif /tmp
```

11. Add the schema file to the directory server.

```
$ bin/ldapmodify \
--port 1389 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
/tmp/opensj_user_schema.ldif
```

12. Add the schema definitions to the configuration repository. You can find the schema definitions at [/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opensj/opensj_config_schema.ldif](#).

```
$ cp /path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opensj/opensj_config_schema.ldif /tmp
```

13. Add the schema file to the directory server.

```
$ bin/ldapmodify \
--port 1389 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
/tmp/opensj_config_schema.ldif
```

14. OpenAM uses the attributes in Table 1.7, "Configuration Data Store Attributes" to search for configuration data. On the OpenDJ directory server, use the **dsconfig** command to add these indexes to your external configuration store. Repeat this step to index the [iplanet-am-user-federation-info-key](#) and [sun-fm-saml2-nameid-infokey](#) attributes if you are deploying federation.

The **dsconfig** subcommand used to create the index depends on the version of OpenDJ directory server.

- Use the following commands with OpenDJ 2.6.x:

```
$ dsconfig create-local-db-index \
--port 4444 \
--hostname opendj.example.com \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--backend-name cfgStore \
--index-name sunxmlkeyvalue \
--set index-type:equality \
--set index-type:substring \
--no-prompt

$ dsconfig create-local-db-index \
--port 4444 \
--hostname opendj.example.com \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--backend-name cfgStore \
--index-name iplanet-am-user-federation-info-key \
--set index-type:equality \
--no-prompt

$ dsconfig create-local-db-index \
--port 4444 \
--hostname opendj.example.com \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--backend-name cfgStore \
--index-name sun-fm-saml2-nameid-infokey \
--set index-type:equality \
--no-prompt
```

- Use the following commands with OpenDJ 3 and later:

```
$ dsconfig create-backend-index \
--port 4444 \
--hostname opendj.example.com \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--backend-name cfgStore \
--index-name sunxmlkeyvalue \
--set index-type:equality \
--set index-type:substring \
--no-prompt

$ dsconfig create-backend-index \
--port 4444 \
--hostname opendj.example.com \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--backend-name cfgStore \
--index-name iplanet-am-user-federation-info-key \
--set index-type:equality \
--no-prompt

$ dsconfig create-backend-index \
--port 4444 \
--hostname opendj.example.com \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--backend-name cfgStore \
--index-name sun-fm-saml2-nameid-infokey \
--set index-type:equality \
--no-prompt
```

15. Rebuild the indexes using the `rebuild-index` command. You can stop the server and run `rebuild-index` in offline mode, or you can run `rebuild-index` online using a task as follows:

```
$ bin/rebuild-index --port 4444 --hostname opendj.example.com \
--bindDN "cn=Directory Manager" --bindPassword password \
--baseDN dc=example,dc=com --rebuildAll \
--start 0
```

16. Verify the indexes. Note that if you are running OpenDJ 3 and later, you need to stop OpenDJ before running this command.

```
$ bin/verify-index --baseDN dc=example,dc=com
```

You have successfully installed and prepared the directory server for an external configuration store. When installing the OpenAM server, you need to specify the host name, port and root suffix of the external directory server on the Configuration Data Store Settings screen of the OpenAM Configurator. See Procedure 2.3, "To Custom Configure an Instance" for more information.

1.4.3. Setting the Configuration Store Heartbeat

OpenAM supports heartbeat monitoring for the configuration data store to prevent idle connections. Idle connections may occur if the external data store is behind a load balancer or firewall. Once the

connection is in an idle state, OpenAM no longer receive notification of configuration changes and may become out-of-sync with other servers sharing the configuration store.

By default, OpenAM issues a heartbeat every ten seconds to monitor for idle connections to the configuration data store.

Two JVM options can be used to override the default heartbeat interval or to disable it completely:

- **org.forgerock.openam.ldap.sm.heartbeat.interval**. Sets the heartbeat interval. The default interval is 10 seconds. If you set the JVM property to 0, it will disable the heartbeat.
- **org.forgerock.openam.ldap.sm.heartbeat.unit**. Sets the time unit of the heartbeat interval. Default is **SECONDS**. Possible values also include: **DAYS**, **HOURS**, **MICROSECONDS**, **MILLISECONDS**, **MINUTES**, **NANOSECONDS**, and **SECONDS**.

Important

You must grant access permissions to all heartbeat requests to the configuration stores that go through firewalls or load balancers.

Procedure 1.12. To Set the Heartbeat Interval

1. Set the heartbeat interval using the JVM startup property **org.forgerock.openam.ldap.sm.heartbeat.interval** in your container.

For example, to set the heartbeat interval to 20 seconds, edit **%CATALINA_HOME%/bin/setenv.sh** (Unix) or **%CATALINA_HOME%\bin\setenv.bat** (Windows) on Tomcat by adding the following line:

```
set CATALINA_OPTS=-Dorg.forgerock.openam.ldap.sm.heartbeat.interval=20
```

2. Make sure that the heartbeat requests have the permissions necessary for a firewall or load balancer.

Procedure 1.13. To Disable the Heartbeat Interval

- Disable the heartbeat interval by setting the JVM startup property **org.forgerock.openam.ldap.sm.heartbeat.interval** to 0.

For example, edit **%CATALINA_HOME%/bin/setenv.sh** (Unix) or **%CATALINA_HOME%\bin\setenv.bat** (Windows) on Tomcat by adding the following line:

```
set CATALINA_OPTS=-Dorg.forgerock.openam.ldap.sm.heartbeat.interval=0
```

Chapter 2

Installing and Starting Servers

This chapter covers installation and startup.

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

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

Table 2.1. Installation Options

Installation Action	Documentation Reference
Install quickly for evaluation using default settings	Procedure 2.1, "To Configure With Defaults" Alternatively, follow the full example in Quick Start Guide.
Install OpenAM server, choosing settings	Procedure 2.3, "To Custom Configure an Instance"
Start an OpenAM server	Section 2.4, "Starting Servers"
Erase the configuration and start over	Procedure 2.2, "To Delete a Configuration Before Redeploying"
Add an OpenAM server to a site	Procedure 2.4, "To Add a Server to a Site"
Implement the Core Token Service	Chapter 3, "Implementing the Core Token Service"
Troubleshoot an OpenAM installation	Procedure 6.1, "To Troubleshoot an Installation"
Install ssoadm for CLI configuration	Section 2.3, "Installing and Using the Tools"
Perform a command-line install	Section 2.3.3, "Installing Silently"
Removing OpenAM	Chapter 5, "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 OpenIG or OpenAM policy agents.

OpenIG 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.

OpenAM policy agents provide policy enforcement on supported web servers and Java EE containers, and are tightly integrated with OpenAM. See the *OpenAM Web Policy Agent User's Guide*, or the

OpenAM Java EE Policy Agent User's Guide for instructions on installing OpenAM policy agents in supported web servers and Java EE application containers.

2.1. Installing a Single Server

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

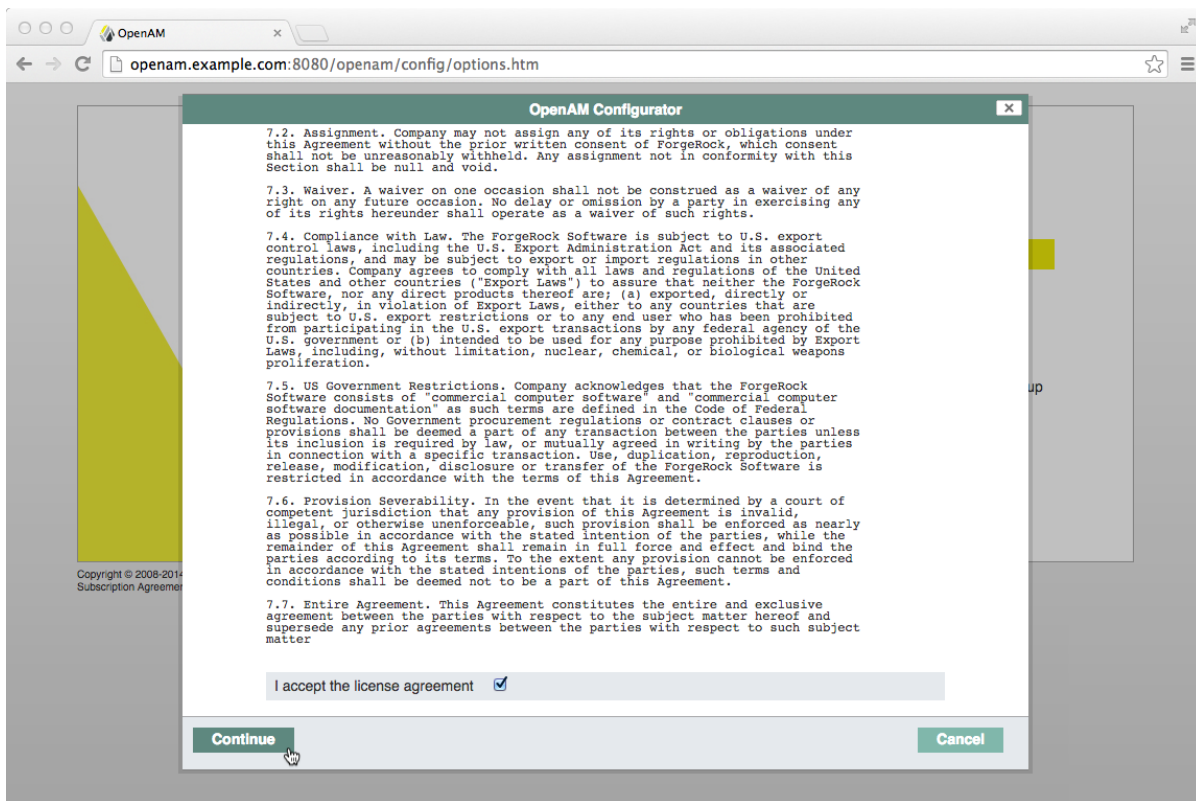
Procedure 2.1. To Configure With Defaults

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

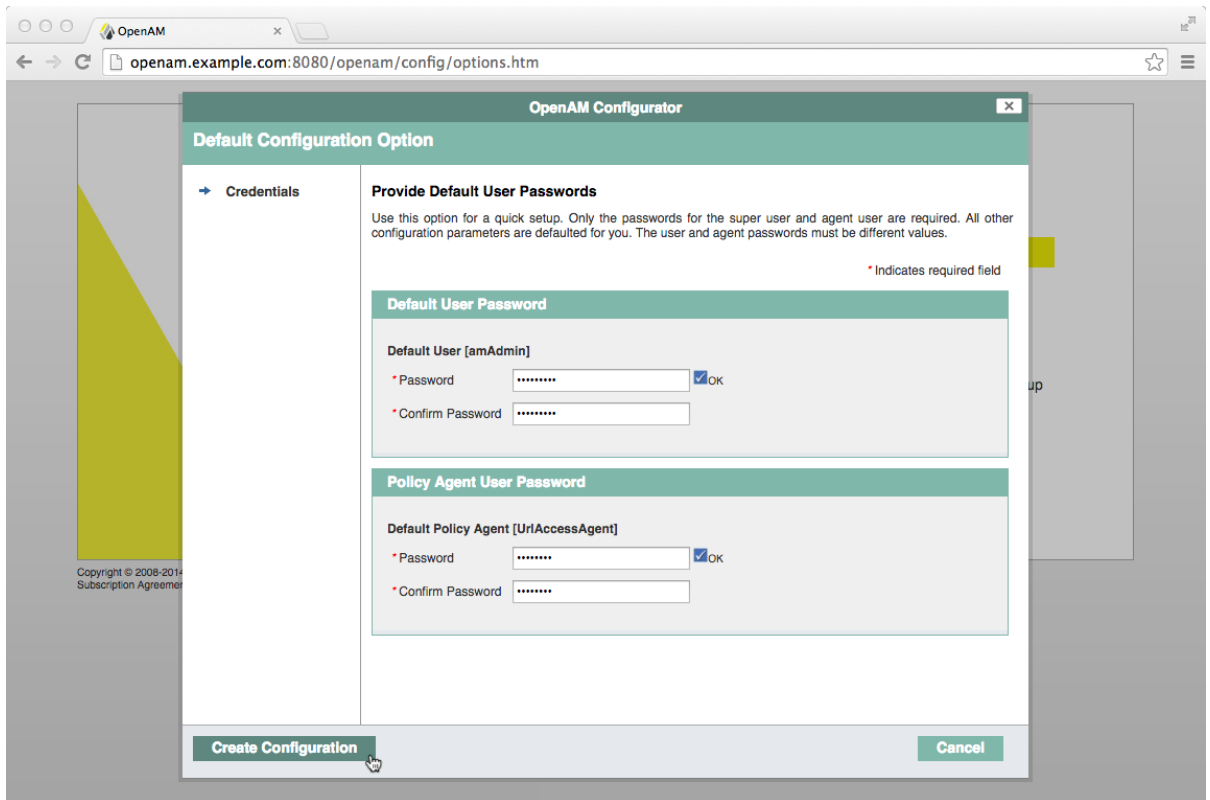
The default configuration sets the cookie domain based on the full URL that was used to access the configurator.

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 OpenAM 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.



3. Provide different passwords for the default OpenAM administrator, **amadmin**, and default Policy Agent users.



The screenshot shows a web browser window with the URL `openam.example.com:8080/openam/config/options.htm`. The page is titled "OpenAM Configurator" and displays the "Default Configuration Option" for "Credentials".

Default Configuration Option

→ Credentials

Provide Default User Passwords

Use this option for a quick setup. Only the passwords for the super user and agent user are required. All other configuration parameters are defaulted for you. The user and agent passwords must be different values.

* Indicates required field

Default User Password

Default User [amAdmin]

* Password ☒ OK

* Confirm Password

Policy Agent User Password

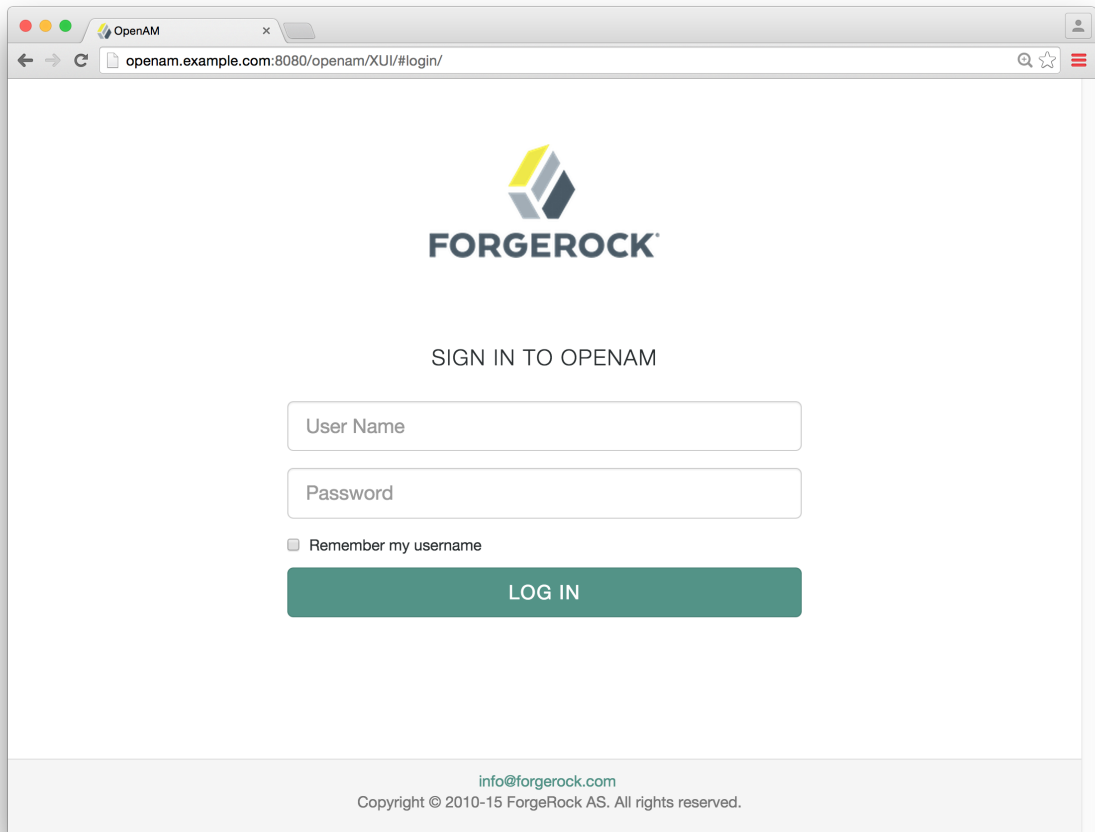
Default Policy Agent [UrlAccessAgent]

* Password ☒ OK

* Confirm Password


Create Configuration **Cancel**

- When the configuration completes, click Proceed to Login, and then login as the OpenAM administrator with the first of the two passwords you provided.



OpenAM

openam.example.com:8080/openam/XUI/#login/



FORGEROCK

SIGN IN TO OPENAM

User Name

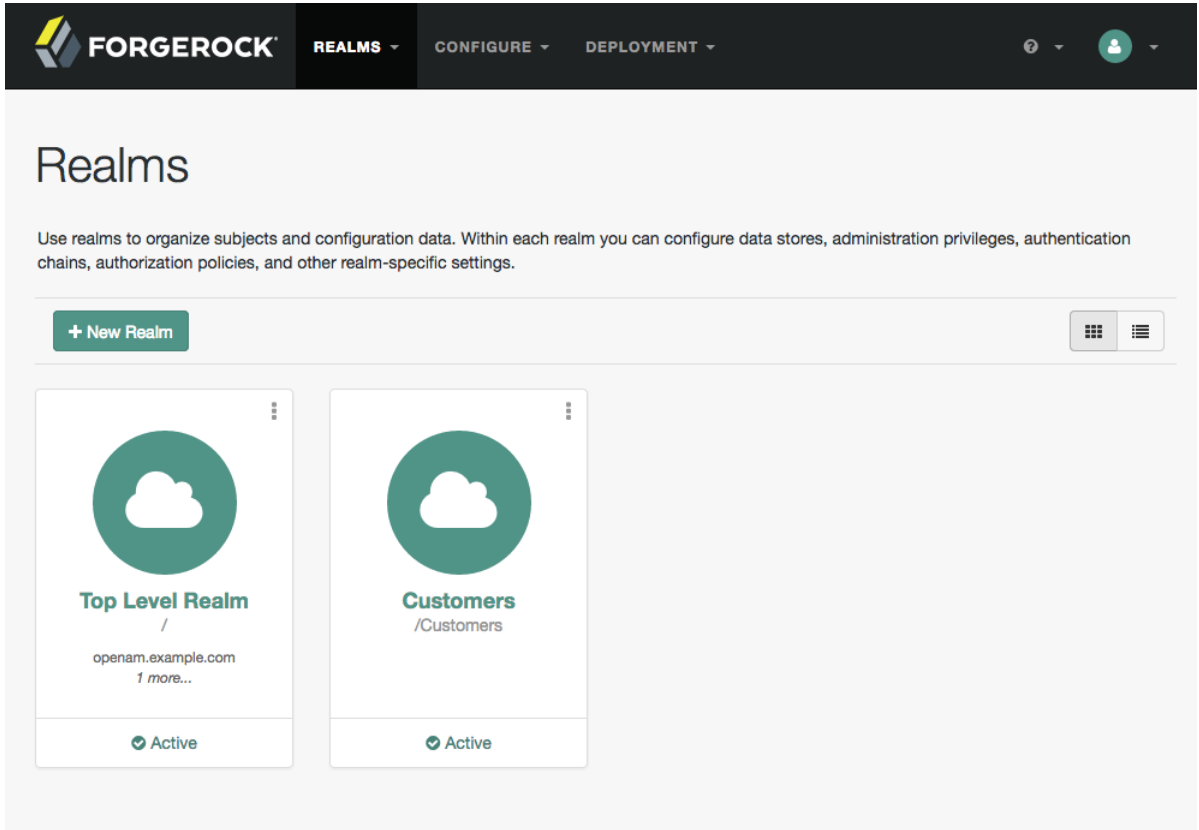
Password

☐ Remember my username

LOG IN

info@forgerock.com
Copyright © 2010-15 ForgeRock AS. All rights reserved.

After successful login, OpenAM redirects you to OpenAM Realms.



Procedure 2.2. 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 OpenAM web application to clear the configuration held in memory.

The following example shuts 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 OpenAM 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 OpenAM 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 OpenAM suffix (by default `dc=openam,dc=forgerock,dc=org`).

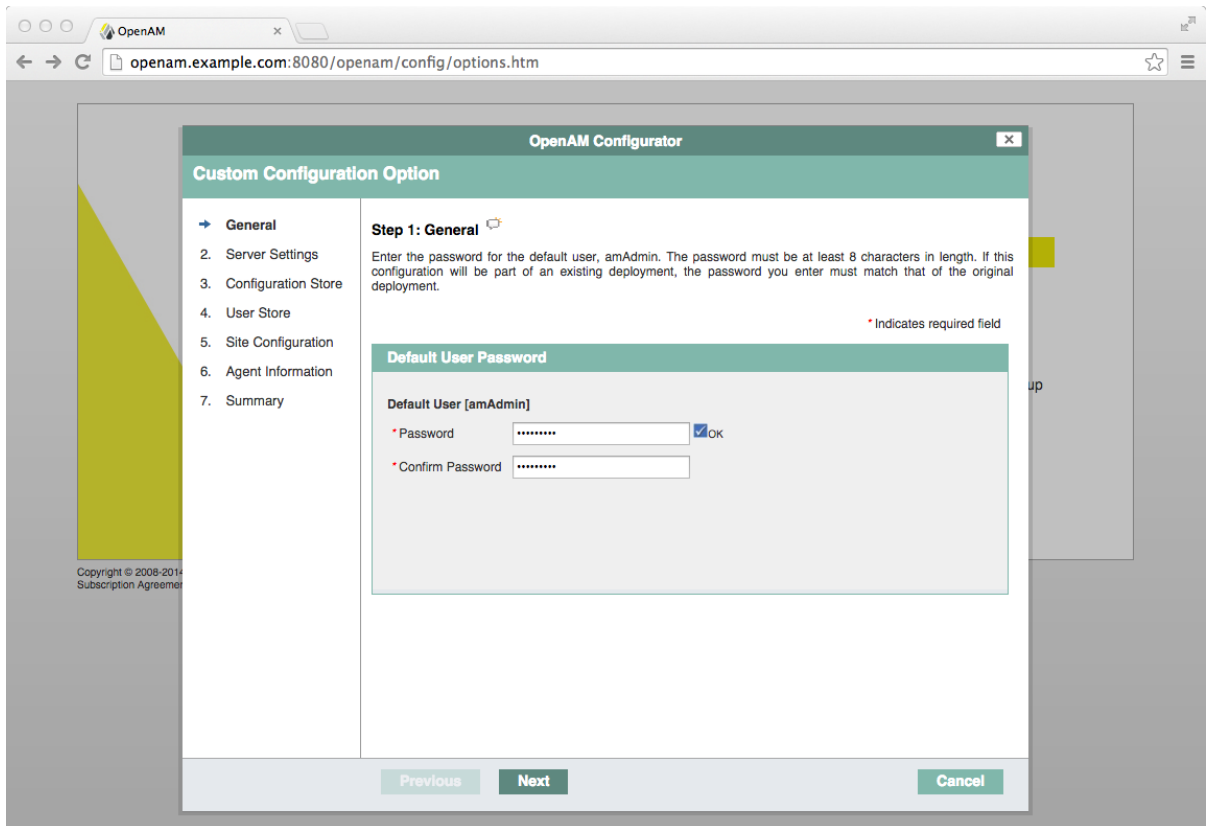
3. Restart the OpenAM web application.

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
```

Procedure 2.3. 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 OpenAM Administrator, `amadmin`.



OpenAM Configurator

Custom Configuration Option

- ➔ General
- 2. Server Settings
- 3. Configuration Store
- 4. User Store
- 5. Site Configuration
- 6. Agent Information
- 7. Summary

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

Previous Next Cancel

Copyright © 2008-2014
Subscription Agreement

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

The screenshot shows a web browser window with the URL `openam.example.com:8080/openam/config/options.htm`. The main content is the 'OpenAM Configurator' dialog box, titled 'Custom Configuration Option'. It has a sidebar with a list of steps: 1. General, 2. Server Settings (selected), 3. Configuration Store, 4. User Store, 5. Site Configuration, 6. Agent Information, and 7. Summary. The main area is titled 'Step 2: Server Settings' and contains the instruction 'Confirm the following settings to use for the server.' Below this is a 'Server Settings' form with four fields: 'Server URL' (http://openam.example.com:8080), 'Cookie Domain' (example.com), 'Platform Locale' (en_US), and 'Configuration Directory' (/home/forgerock/openam). A legend indicates that an asterisk (*) denotes a required field. At the bottom of the dialog are three buttons: 'Previous' (highlighted), 'Next', and 'Cancel'. A copyright notice for 2008-2014 is visible in the bottom left corner of the dialog.

Server URL

Provide a valid URL to the base of your OpenAM 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 OpenAM 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`.

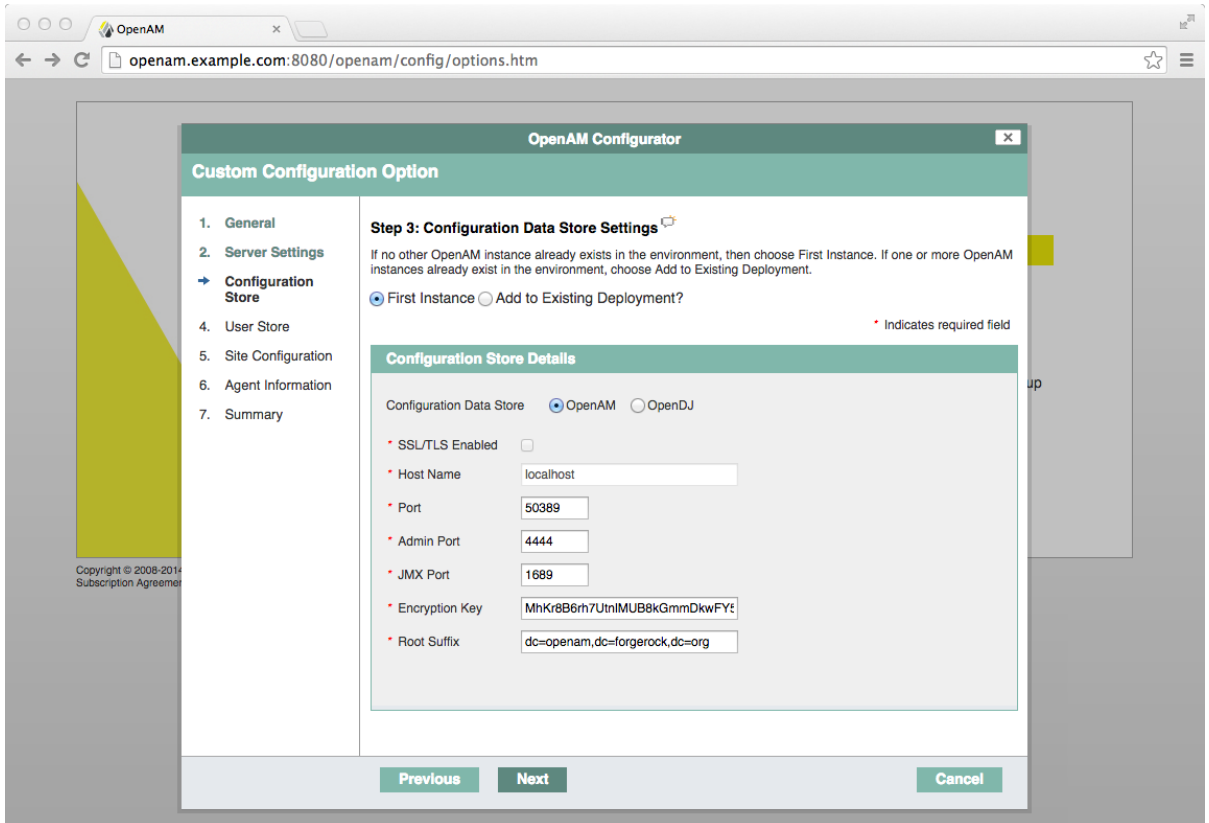
Platform Locale

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 OpenAM configuration files. OpenAM must be able to write to this directory.

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



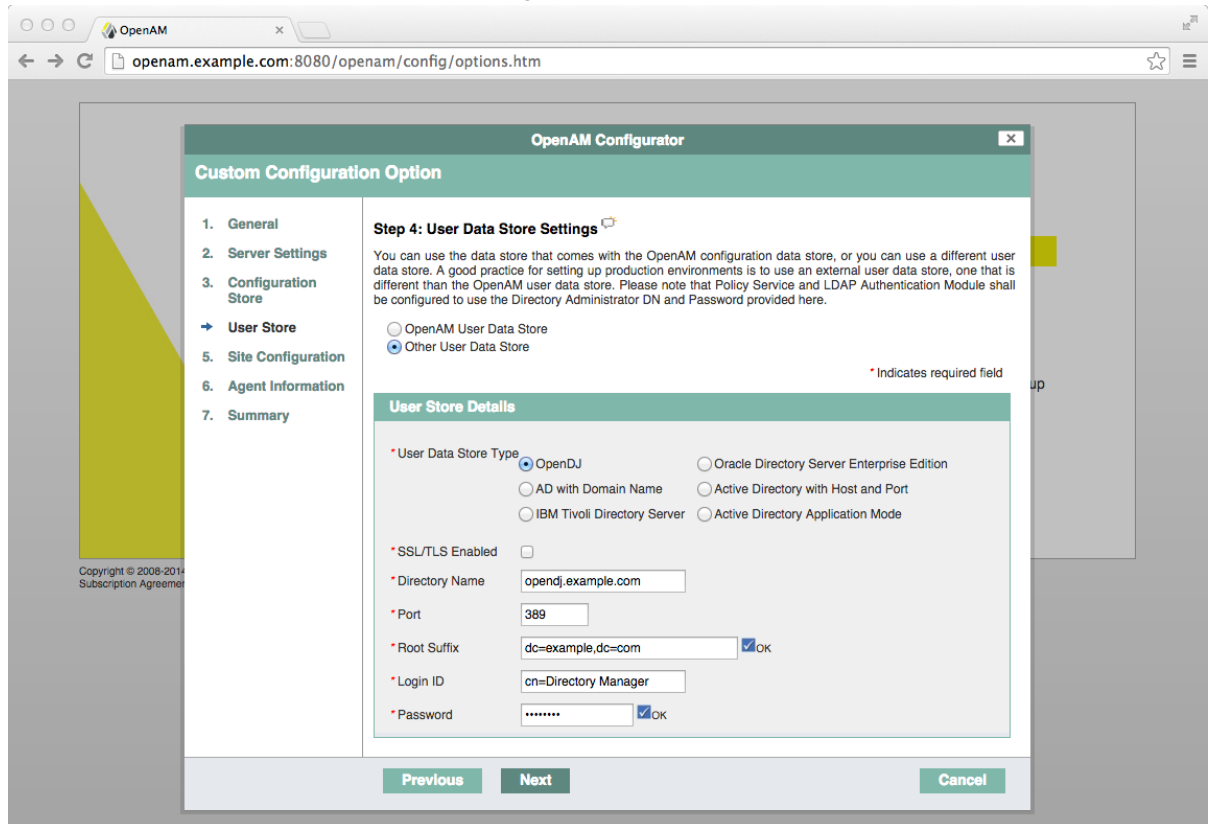
The screenshot shows a web browser window with the URL `openam.example.com:8080/openam/config/options.htm`. The main content area is titled "OpenAM Configurator" and "Custom Configuration Option". On the left is a navigation menu with items: 1. General, 2. Server Settings, 3. Configuration Store (selected), 4. User Store, 5. Site Configuration, 6. Agent Information, and 7. Summary. The main panel is titled "Step 3: Configuration Data Store Settings" and includes instructions: "If no other OpenAM instance already exists in the environment, then choose First Instance. If one or more OpenAM instances already exist in the environment, choose Add to Existing Deployment." Below this are two radio buttons: "First Instance" (selected) and "Add to Existing Deployment?". A red asterisk indicates required fields. The "Configuration Store Details" section contains the following fields: "Configuration Data Store" (radio buttons for "OpenAM" and "OpenDJ", with "OpenAM" selected), "SSL/TLS Enabled" (checkbox, unchecked), "Host Name" (text box with "localhost"), "Port" (text box with "50389"), "Admin Port" (text box with "4444"), "JMX Port" (text box with "1689"), "Encryption Key" (text box with "MhKr8B6rh7UtnIMUB8kGmmDkwFYt"), and "Root Suffix" (text box with "dc=openam,dc=forgerock,dc=org"). At the bottom are "Previous", "Next", and "Cancel" buttons.

You can also add this OpenAM installation to an existing deployment, providing the URL of the site. See Procedure 2.4, "To Add a Server to a Site" for details.

Alternatively, if you already manage an OpenDJ deployment, you can store OpenAM configuration data in your existing directory service. You must, however, create the suffix to store configuration data on the directory server before you configure OpenAM. OpenAM does not create the suffix when you use an external configuration store. For instructions to create a configuration store backend, see Step 3 in Procedure 1.11, "To Install an External OpenDJ Directory Server".

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

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



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 OpenAM 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.

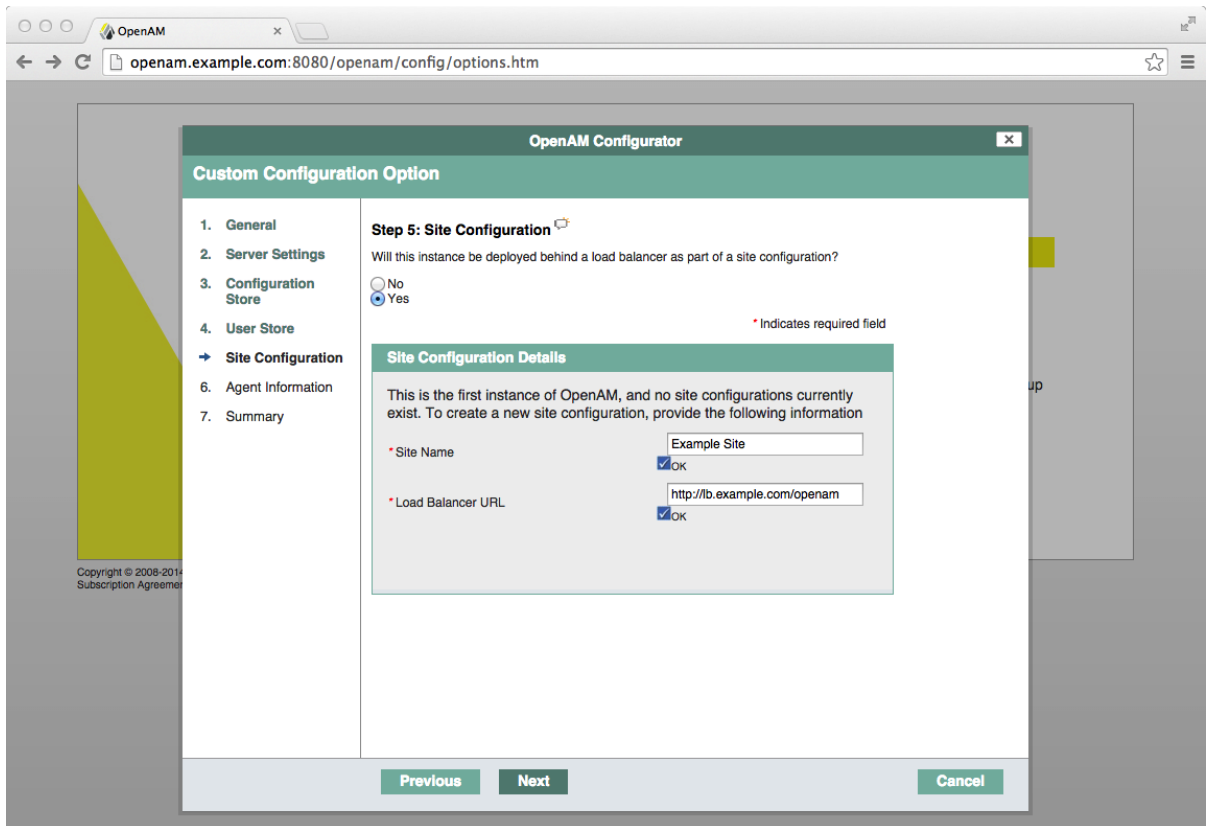
Password

Password for the directory administrator user.

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

When you deploy multiple servers, OpenAM automatically enables session high availability.¹ OpenAM stores session data in a directory service that is shared by multiple OpenAM servers. The shared storage means that if an OpenAM server fails, other OpenAM 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 OpenAM to store sessions *statefully* or *statelessly*. Stateful sessions are stored in the Core Token Service; stateless sessions are stored in HTTP cookies on the client. Because stateless 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. OpenAM does not store stateless sessions in the CTS store. For more information about stateful and stateless sessions, see Section 1.8.1, "Session State" in the *Authentication and Single Sign-On Guide*.



OpenAM Configurator

Custom Configuration Option

- General
- Server Settings
- Configuration Store
- User Store
- Site Configuration**
- Agent Information
- Summary

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

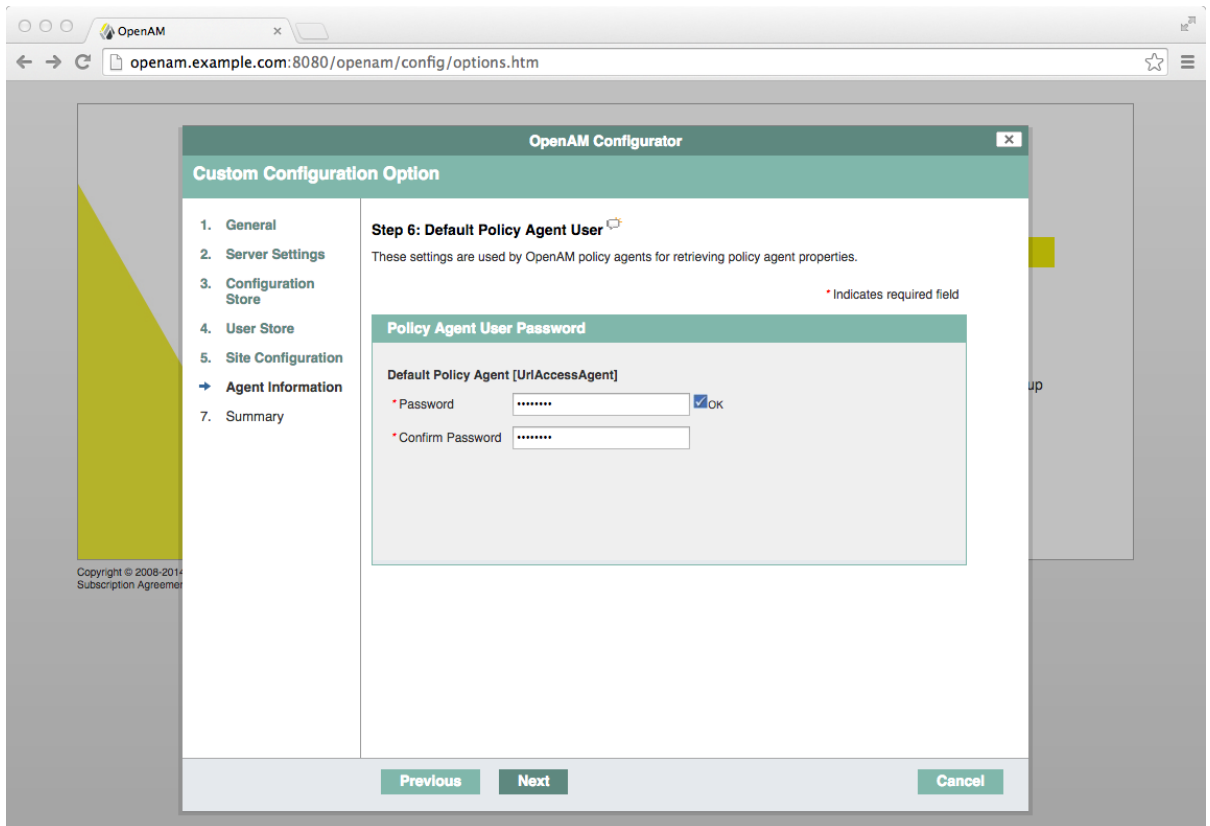
This is the first instance of OpenAM, and no site configurations currently exist. To create a new site configuration, provide the following information

* Site Name ☒ OK

* Load Balancer URL ☒ OK

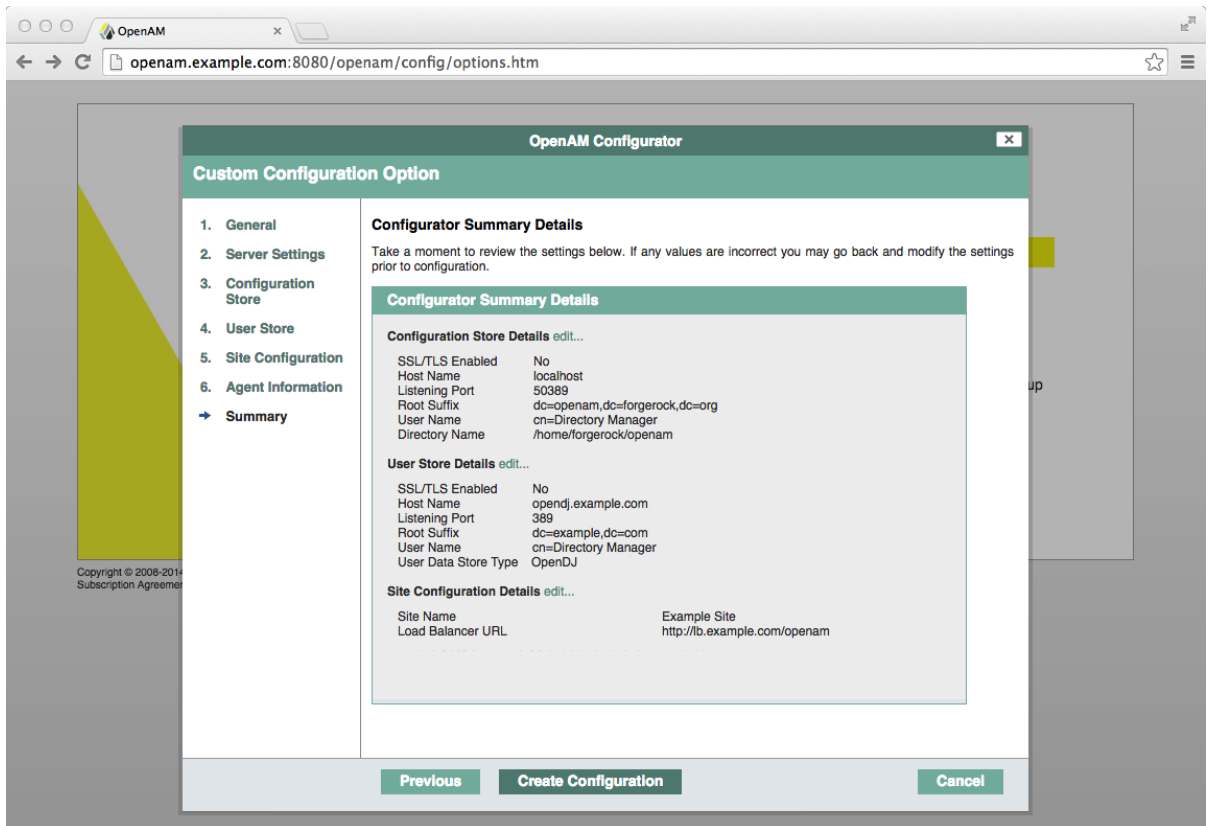
It is possible to set up a site after initial installation and configuration. See Section 2.3, "Deployment Configuration" in the *Reference* for more information.

- In the Agent Information screen, provide a password with at least eight characters to be used by policy agents to connect to OpenAM.

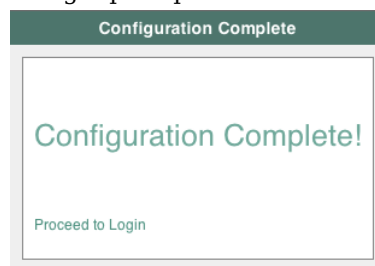


The screenshot shows a web browser window with the URL `openam.example.com:8080/openam/config/options.htm`. The main content area is titled "OpenAM Configurator" and "Custom Configuration Option". On the left, a navigation menu lists steps 1 through 7, with "Agent Information" selected. The main panel displays "Step 6: Default Policy Agent User" with a sub-header "Policy Agent User Password". Below this, there are two password fields: "Default Policy Agent [UrlAccessAgent]" and "Confirm Password". Both fields are marked with a red asterisk indicating they are required. The "Password" field has an "OK" checkbox next to it. At the bottom of the configurator window, there are "Previous", "Next", and "Cancel" buttons. A copyright notice "Copyright © 2008-2014 Subscription Agreement" is visible in the bottom left corner of the configurator window.

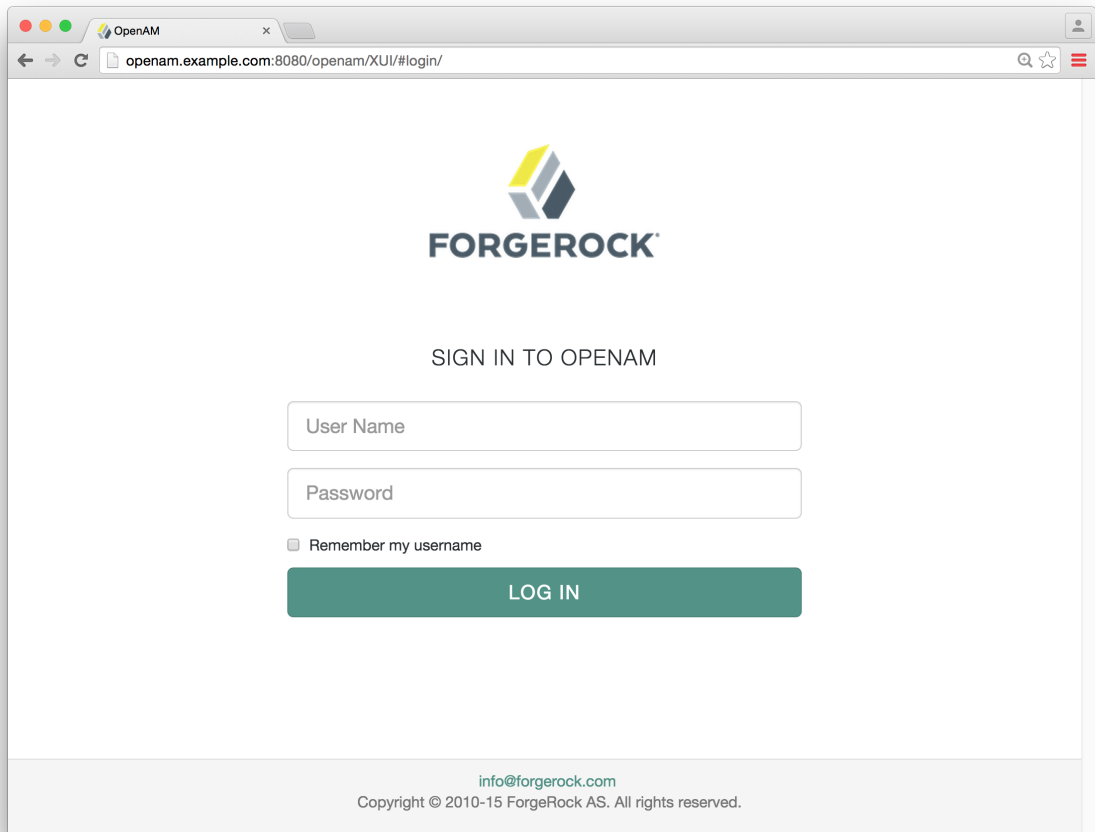
9. Check the summary screen, and if necessary, click Previous to return to earlier screens to fix any configuration errors as needed.



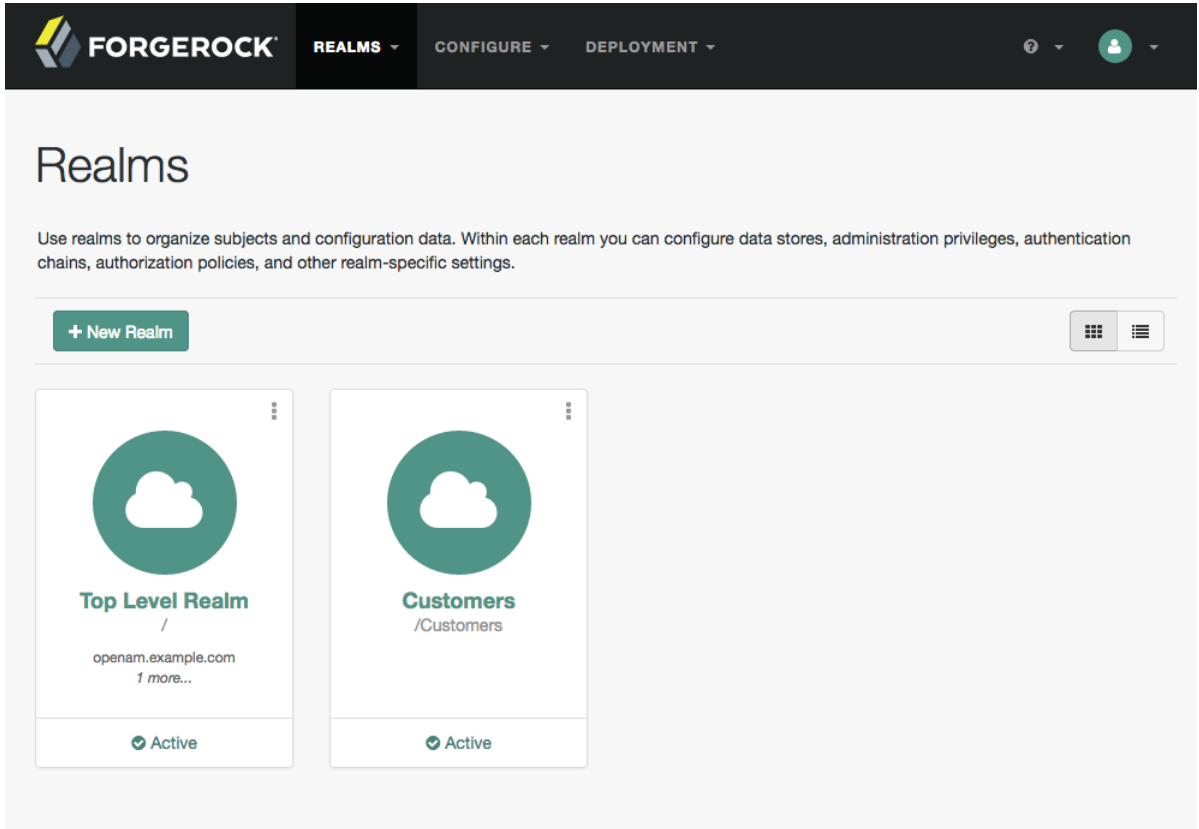
After you click Create Configuration in the summary screen, configuration proceeds, logging progress that you can read in your browser and later, in the installation log. The process ends, and OpenAM shows the Proceed to Login prompt.



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



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



You can also access the AM console by browsing to the console URL, for example <http://openam.example.com:8080/openam/console>.

11. 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.
12. If you specified the Other User Data Store option in the User Data Store Settings screen, you must index several attributes in your external identity repository. See Procedure 1.10, "To Index External Identity Repository Attributes" for more information.

2.2. Installing Multiple Servers

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

2.2.1. 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 OpenAM to route around servers that are down.

OpenAM uses a *site* for this purpose. In an OpenAM site, multiple OpenAM 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 OpenAM software. When installed properly, a site configuration improves service availability, as the load balancer routes around OpenAM 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 OpenAM services as well. The load balancer can restrict access to OpenAM 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 OpenAM with a load balancer or proxy service, configure your container so that OpenAM can trust the load balancer or proxy service.
- OpenAM authentication can depend on information about the user to authenticate, such as the IP address where the request originated. When OpenAM is accessed through a load balancer or proxy layer, pass this information along using request headers. Also, configure OpenAM to consume and to forward the headers as necessary. See Section 2.2.4, "Handling HTTP Request Headers" for details.

2.2.2. Configuring Sites

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

Procedure 2.4. To Add a Server to a Site

High availability requires redundant servers in case of failure. With OpenAM, you configure an OpenAM 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:

² Technically, it is possible to configure a site with only one OpenAM server.

1. If the site is already using keystore keys in the configuration, such as for signing stateless cookies, SAML v2.0 assertions, and others, copy the keystore from any of the servers of the site to the configuration directory of the new instance:
 - a. Create the configuration directory of the new instance, for example, `/path/to/openam/openam`. This directory must be the same as the configuration directory used by other AM servers in the site.
 - b. Log in to the AM console and navigate to Deployment > Servers > *Server Name* > Security > Key Store and find the following properties:
 - Keystore File
 - Keystore Password File
 - Private Key Password File

These properties configure the name and path of the keystore files you need to copy to the new instance. By default, they are located in the `/path/to/openam/openam` directory.
 - c. Copy the keystore files from one of the instances of the site to the `/path/to/openam/openam` directory created for the new instance.
2. Navigate to the deployment URL of the new instance. You should see the AM configurator page.
3. In the initial configuration screen, under Custom Configuration, click Create New Configuration.
4. In the first screen, enter the same password entered for the OpenAM Administrator, `amadmin`, when you configured the first server in the site.
5. Configure server settings as required.

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

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

The directory used to store configuration data should use the same directory service used for this purpose by other OpenAM servers in the site. If you use the embedded OpenDJ directory 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.

7. 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.

8. In the summary screen, verify the settings you chose, and then click Create Configuration.
9. When the configuration process finishes, click Proceed to Login, and then login as the OpenAM administrator to access the AM console.

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:

Procedure 2.5. To Configure a Site With the First Server

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

1. Log in to the AM console as administrator, by default `amadmin`, and then navigate to Deployment > Sites.
2. Click New 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 OpenAM servers in the site. For example, if your load balancer listens for HTTPS on host `lb.example.com` and port `443` with OpenAM under `/openam`, then your site URL is `https://lb.example.com/openam`.

Client applications and policy agents access the servers in the site through the site URL.

4. Click Save to keep the site configuration.
5. Navigate to Deployment > Servers > *Server Name* > General.
6. Set the Parent Site to the name of the site you just created, and then select Save Changes.

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

For all additional servers in the OpenAM site, add them to the site at configuration time as described in Procedure 2.4, "To Add a Server to a Site".

2.2.3. Configuring Load Balancing for a Site

Load balancer configuration requirements differ for OpenAM sites configured to use stateful and stateless sessions.³ For more information about OpenAM session types, see Section 1.8.1, "Session State" in the *Authentication and Single Sign-On Guide*.

³ Some OpenAM deployments use both stateful and stateless sessions. If your deployment uses a substantial number of stateful sessions, follow the recommendations for deployments with stateful sessions.

2.2.3.1. Load Balancer Configuration for Stateful Sessions

An OpenAM server that authenticates a user stores the session authoritatively in the Core Token Service token store and caches it in its memory heap. An OpenAM site configured to use stateful sessions achieves the best performance when the server that originally authenticated a user continually manages that user's session, unless that server is no longer available.

To achieve optimal performance, configure your load balancer for sticky sessions as follows:

Procedure 2.6. To Configure Site Load Balancing for Deployments With Stateful Sessions

1. For each OpenAM server in the site, navigate to Deployment > Servers > *Server Name* > General and set Parent Site to the site you created. Then, save your work.
2. Make the `amlbcookie` value unique for each OpenAM server.
 - a. For each OpenAM server console in the site, navigate to Deployment > Servers > *Server Name* > Advanced and set the `com.ipplanet.am.lbcookie.value` property to a unique value.

By default, the cookie value is set to the OpenAM server ID.

Changes take effect only after you restart the OpenAM server.

- b. Restart each OpenAM server where you changed the cookie value. You can then check the cookie value by logging in to the AM console, and examining the `amlbcookie` cookie in your browser.
3. 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 OpenAM server.

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

2.2.3.1.1. Load Balancer Termination

When traffic to and from the load balancer is protected with HTTPS, the approach described in Procedure 2.6, "To Configure Site Load Balancing for Deployments With Stateful Sessions" requires that you terminate the connection on the load balancer. You then either re-encrypt the traffic from the load balancer to OpenAM, or make connections from the load balancer to OpenAM over HTTP.

2.2.3.1.2. Request Forwarding Caveats

Sticky load balancing based on the value of the `amlbcookie` cookie does not guarantee request forwarding to the corresponding OpenAM 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 OpenAM server on which a user's session is cached.

The OpenAM 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.

2.2.3.2. Load Balancer Configuration for Stateless Sessions

An OpenAM site configured to use stateless sessions does not require any special load balancer configuration.

A request from a user to an OpenAM site does not need to be processed on the OpenAM server that originally authenticated the user. Any server in the site can accept a request from an OpenAM user with no performance degradation because the user's session resides in an HTTP cookie—not on the server—and is passed to the OpenAM server along with the request.

2.2.4. 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 OpenAM 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 OpenAM to consume and to forward the headers as necessary. For example, to configure OpenAM 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 OpenAM server can forward requests to another OpenAM 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**.

2.2.5. Handling Multiple Cookie Domains When Using Wildfly

If you are using Wildfly as the OpenAM 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**.

2.3. 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 OpenAM tools are:

`SSOAdminTools-14.0.0.zip`

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

See Section 2.3.1, "Setting up Administration Tools".

`SSOConfiguratorTools-14.0.0.zip`

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

See Section 2.3.2, "Setting up Configuration Tools" and Section 2.3.3, "Installing Silently".

2.3.1. Setting up Administration Tools

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

Procedure 2.7. To Set up Administration Tools

1. Verify that OpenAM 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-14.0.0.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. 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 OpenAM 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 OpenAM:

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

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

8. If you connect to OpenAM over HTTPS, and if your web container uses a self-signed certificate as described in [Procedure 4.1, "To Set Up With HTTPS and Self-Signed Certificates"](#), then the **ssoadm** command will not be able to trust the certificate.

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. 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 OpenAM 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
http://openam.example.com:8080/openam
```

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

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

```
-D"com.ipplanet.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 OpenAM server against which you set up the **ssoadm** command.

Important

The **ssoadm** command is dependent on the OpenAM 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 OpenAM server against which you set up the **ssoadm** command is at <http://openam.example.com:8080/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=http://openam.example.com:8080/openam"
```

Repeat this step for each OpenAM 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 OpenAM server.

2.3.2. Setting up Configuration Tools

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

Procedure 2.8. 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 OpenAM:

```
$ cd /path/to/openam-tools/config  
$ unzip ~/Downloads/openam/SSOConfiguratorTools-14.0.0.zip  
Archive: ~/Downloads/openam/SSOConfiguratorTools-14.0.0.zip  
  creating: legal-notice/  
    inflating: legal-notice/LICENSE.DOM-software.html  
    inflating: legal-notice/NOTICE.resolver.txt  
    inflating: legal-notice/LICENSE.DOM-documentation.html  
  ... (more output) ...  
extracting: lib/xml-apis-2.11.0.jar  
extracting: openam-configurator-tool-14.0.0.jar  
extracting: lib/servlet-api-2.5.jar
```


2.3.3. Installing Silently

Use the OpenAM configurator tool, `openam-configurator-tool-14.0.0.jar`, to silently install OpenAM. The OpenAM server must be deployed and running, but not yet configured, when you use the tool.

Procedure 2.9. 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 OpenAM server. For property file options, see `configurator.jar(1)`.

Copy the sample configuration property file provided with OpenAM, 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
AMLDAPUSERPASSWD=secret12
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, OpenAM auto-accepts the software licensing agreement and suppresses display of the license acceptance screen during silent installation.
 - When installing OpenAM 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 OpenAM configurator tool, `openam-configurator-tool-14.0.0.jar`:

```
java -jar openam-configurator-tool-14.0.0.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 OpenAM to support HTTPS. Specify the OpenAM web container's trust store for `PATH_TO_JKS_TRUSTSTORE`.

Output similar to the following appears:

```
$ java -jar openam-configurator-tool-14.0.0.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 xxxxxxxx --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!
```

2.4. Starting Servers

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

At the beginning of its startup process, OpenAM performs an operation called *bootstrapping*, during which OpenAM obtains startup settings from a bootstrap file in its configuration directory, then uses

those settings to initiate its operation. OpenAM creates the bootstrap file during installation, based on values you provide when you run the OpenAM configurator.

After every successful startup, OpenAM rewrites the bootstrap file with its initial contents.

2.4.1. Overriding Startup Settings

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

- Section 2.4.1.1, "Replacing the Bootstrap File" covers how to specify a custom bootstrap file, and describes all the startup settings in the bootstrap file.
- Section 2.4.1.2, "Overriding Startup Settings by Using Environment Variables" covers how to dynamically override startup settings in the bootstrap file with environment variables.
- Section 2.4.1.3, "Overriding Startup Settings by Using Java Properties" covers how to dynamically override startup settings in the bootstrap file with Java properties.

2.4.1.1. Replacing the Bootstrap File

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

- In the Configuration Directory field on the Server Settings page when using GUI installation. See Procedure 2.3, "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 OpenAM's startup configuration, modify the bootstrap file, `boot.json`, and then overwrite the existing bootstrap file with your modified file *prior to every OpenAM restart*. You must overwrite the file each time you start OpenAM because after startup, OpenAM overwrites the bootstrap file with the initial startup settings created during OpenAM 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—OpenAM 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, OpenAM will not be able to start up.

The following is an example OpenAM bootstrap file:

```
{
  "instance" : "http://openam.example.com:8080/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 OpenAM bootstrap file has the following properties:

Table 2.2. Startup Settings in the Bootstrap File

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

Property	Description and Derivation
<code>keystores.default.keyStoreType</code>	OpenAM key store type. Currently, the only valid value is <code>JCEKS</code> .
<code>keystores.default.keyStoreFile</code>	<p>Path to the OpenAM keystore. Always created initially as <code>/path/to/openam/openam/keystore.jceks</code>.</p> <p>The OpenAM keystore is required for startup because it contains the password of the directory manager user of the OpenAM configuration store.</p>
<code>configStoreList[*]</code>	Array of one or more objects that describe OpenAM 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.
<code>configStoreList[*].baseDN</code>	<p>Root suffix of the OpenAM configuration store.</p> <p>Defaults to the Root Suffix field on the Configuration Data Store Settings page (GUI configurator) or the <code>ROOT_SUFFIX</code> configuration property (command-line configurator).</p>
<code>configStoreList[*].dirManagerDN</code>	<p>DN of the configuration store directory manager user.</p> <p>Defaults to <code>cn=Directory Manager</code> (GUI configurator) or the <code>DS_DIRMGRDN</code> configuration property (command-line configurator).</p>
<code>configStoreList[*].ldapHost</code>	<p>Fully-qualified domain name (FQDN) of the configuration store's host.</p> <p>Defaults to the Host Name field on the Configuration Data Store Settings page (GUI configurator) or the <code>DIRECTORY_SERVER</code> configuration property (command-line configurator).</p>
<code>configStoreList[*].ldapPort</code>	<p>LDAP or LDAPS port number on which to access the configuration store.</p> <p>Defaults to the Port field on the Configuration Data Store Settings page (GUI configurator) or the <code>DIRECTORY_PORT</code> configuration property (command-line configurator).</p>
<code>configStoreList[*].ldapProtocol</code>	<p>Protocol with which to access the directory service running the configuration store. The value can be <code>ldap</code> or <code>ldaps</code>.</p> <p>Defaults to the SSL/TLS Enabled field on the Configuration Data Store Settings page (GUI configurator) or the <code>DIRECTORY_SSL</code> configuration property (command-line configurator).</p>

2.4.1.2. 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 OpenAM 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 OpenAM instance URL as follows:

Procedure 2.10. To Override Startup Settings by Using Environment Variables

1. Set an environment variable named `MY_INSTANCE` in the shell that starts OpenAM.
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 OpenAM.

2.4.1.3. 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 OpenAM.

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 OpenAM keystore's path to the user's home directory as follows:

Procedure 2.11. To Override Startup Settings by Using Environment Variables

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

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

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

Chapter 3

Implementing the Core Token Service

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

CTS supports *session high availability*, which lets OpenAM manage a session as long as one of the OpenAM servers in a clustered deployment is available. After a user has successfully authenticated, OpenAM creates a stateful session and stores it in the CTS. Any OpenAM 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:

- OpenAM stateful sessions
- OAuth 2.0
- UMA
- Session blacklist (if enabled for stateless sessions)
- SAML v2.0 (if enabled for Security Token Service token validation and cancellation)
- OAuth 2.0 stateless sessions and session blacklist
- Push notification during authentication
- Cluster-wide notification

3.1. 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:

- **Only Use the Embedded Configuration Store for Limited, Single-Server Test Cases.** By default, OpenAM writes CTS entries in the OpenAM configuration store: either an embedded or external configuration store. If you configured OpenAM 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.
- **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

¹Prior to AM 5, the authoritative source for stateful sessions was the memory heap of OpenAM's web container.

²Prior 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.

but much less volatile than CTS data. Therefore, isolating the user, configuration, and CTS data from OpenAM into separate stores allow for different tuning and storage settings per token store type.

- **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. Note that CTS data are highly volatile with high writes (about 90%) and low reads (about 10%).

Also, a requirement for global replication of the tokens stored in CTS justifies a move to dedicated systems, which provide an extra level of control over the amount of replication that is occurring.

- **Properly Tune Your OpenDJ Directory 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 Section 3.4, "Managing CTS Tokens". When enabled, token compression reduces load requirements on the network connection between token stores in exchange for processing time-compressing tokens.
- **Determine the CTS Deployment Architecture.** There are two options for deploying CTS token stores:
 - Active/passive deployments, in which OpenAM's connection to the CTS token store is limited to a single master instance with failover instances.

Active/passive deployments are slightly simpler to deploy. Active/passive deployments are a good fit for deployments with only a couple of OpenAM servers.

- Affinity deployments, in which OpenAM connects to one or more writable directory server instances. Each instance acts as the master for a subset of CTS tokens. In this architecture, CTS tokens are described as having an *affinity* for a given directory server instance. Specifically, OpenAM routes requests with the same target DN to the same directory server.

Affinity deployments allow you to spread requests to the CTS token store across multiple directory master instances. Affinity deployments are a good fit for deployments with many OpenAM servers.

Do not deploy CTS token stores behind a load balancer. Instead, specify connections to the directory server instances that comprise the CTS token store by using the Connection String(s) property in the CTS configuration.

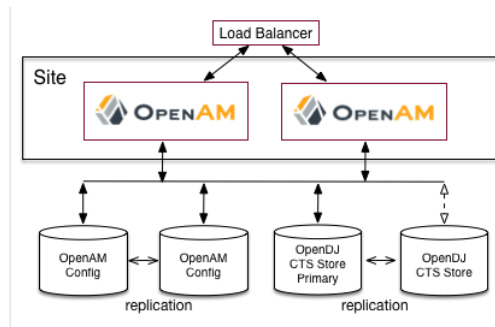
Performance depends on the characteristics of your deployment, but generally is comparable for both architectures.

- **Do Not Use a Load Balancer in Front of the CTS Stores.** To connect OpenAM to the CTS store, specify the main external OpenDJ directory server for the CTS store on the AM console and designate additional OpenDJ instances for failover using the Connection String(s) property. This property allows you to configure multiple OpenDJ directory servers for your CTS token stores without a load balancer.

The following diagram shows a simple OpenAM deployment with the following characteristics:

- A single load balancer receives requests for a cluster of two OpenAM servers.
- Four OpenDJ servers provide services for the OpenAM servers.
- Two of the OpenDJ servers act as the configuration store. Either OpenAM server can write to either configuration store server. The two OpenDJ servers use multimaster replication for consistency.
- The other two OpenDJ servers, deployed in an active/passive topology, serve as the Core Token Service store. Changes from either OpenAM server are written to the primary OpenDJ CTS server (on the left) and then replicated to the other OpenDJ CTS server (on the right).

Figure 3.1. A Simple CTS Deployment

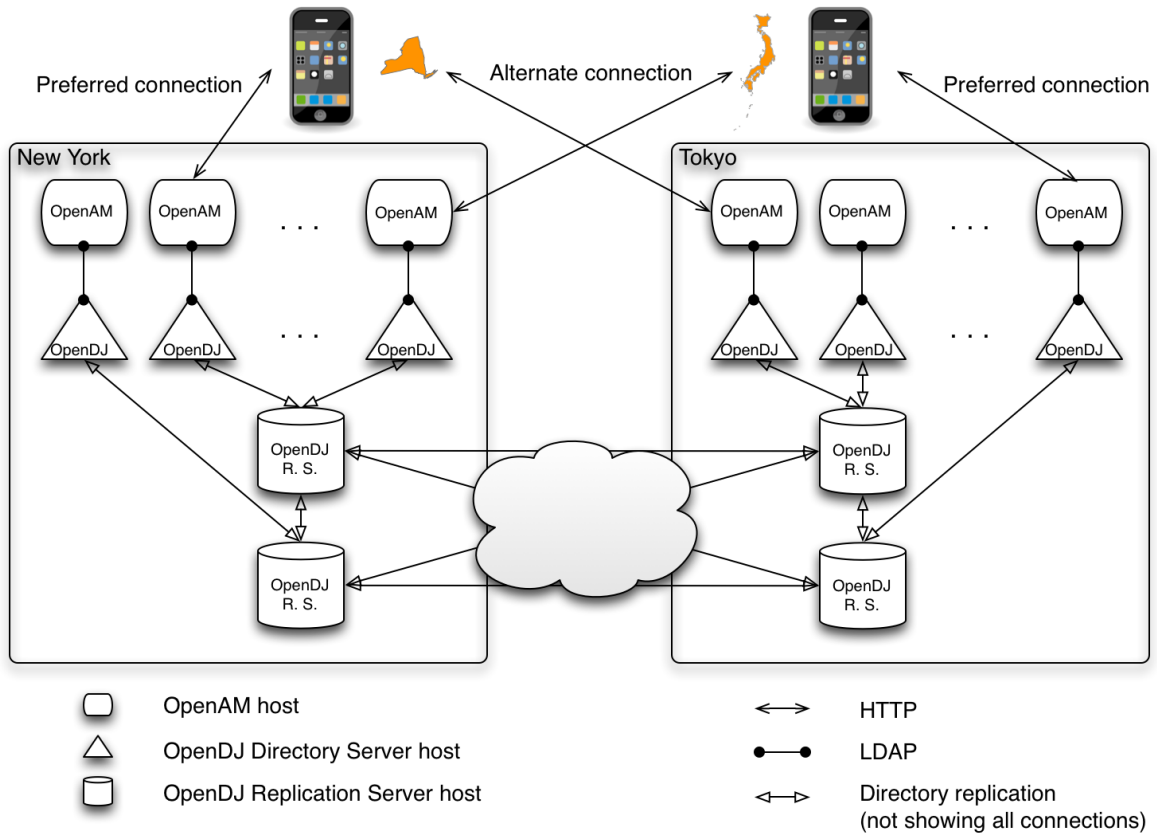


When OpenAM writes to a directory server in the external CTS store, directory server replication pushes the write to other directory servers in the same replication group. Under load, operations in an OpenAM server can happen more quickly than the network can push replication updates. Therefore, balancing the LDAP traffic from OpenAM 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.

- **Consider Dedicated Replication Servers.** Once configured, the OpenDJ directory service replicates CTS data transmitted from OpenAM servers to connected OpenDJ servers. The amount of replication traffic can be significant, especially if replication proceeds over a WAN. You can limit this replication traffic by separating OpenDJ instances into directory and replication servers. For more information on how this is done with OpenDJ, see the OpenDJ documentation on *Standalone Replication Servers*.
- **Replicate CTS Across Sites to Support Global Session Availability.** 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, OpenAM 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 OpenAM server in an active data center as shown in Figure 3.2, "Core Token Service For Global Session Failover":

Figure 3.2. Core Token Service For Global Session Failover



For more information on CTS for global session high availability with OpenDJ directory server, see the OpenDJ documentation on [Managing Data Replication](#).

3.2. CTS Deployment Steps

The Default Configuration option installs OpenAM with an embedded OpenDJ directory server that stores both configuration and CTS data. The default option is suitable for OpenAM evaluation purposes, or for single site or smaller-scale environments where lower volume write loads and replication traffic occur.

In general, CTS causes more volatile replication traffic due to the nature of its short-lived tokens compared to regular configuration data. To handle the data volatility, you can configure OpenAM to use the embedded directory server as a dedicated configuration token store, while using an external OpenDJ directory server instance as a CTS store. This type of deployment is useful if you have multiple OpenAM instances in a fully-replicated topology communicating with an external CTS token store over a WAN.

You can deploy CTS using an external directory server by running the instructions in the following sections:

- Section 3.2.1, "Prepare the OpenDJ Directory Service for CTS"
- Section 3.2.2, "Import CTS Files"
- Section 3.2.3, "Non-Admin User Creation and ACI Import"
- Section 3.2.4, "CTS Index Import and Build"
- Section 3.2.5, "CTS Configuration"
- Section 3.2.6, "Testing Session High Availability"

This section assumes that you have deployed two OpenAM instances in a site. If you have not completed these steps, see [Procedure 2.6, "To Configure Site Load Balancing for Deployments With Stateful Sessions"](#). It is also assumed that both OpenAM instances communicate with the CTS instance, [cts.example.com](#) on port 1389.

3.2.1. Prepare the OpenDJ Directory Service for CTS

The following instructions show how to download, install, and set up the OpenDJ directory server.

Procedure 3.1. To Download and Install OpenDJ

1. Go to the [ForgeRock BackStage](#) website, and then download a supported version of OpenDJ directory server.
2. Unzip the OpenDJ distribution and run **setup**, which launches a GUI application called the QuickSetup Wizard. If you want to run **setup** interactively from the command line, use **setup --cli**.
3. Install OpenDJ with the installation parameters necessary for your deployment. Note, however, that SSL may be required in production deployments. This example uses the following parameters:

```

Accept license?: yes
Initial Root User DN for the Directory Server: cn=Directory Manager
Password for the Initial Root User: <password value>
Fully Qualified Hostname: cts.example.com
LDAP Listening Port: 1389
Administration Connector Port: 4444
Create Base DNs: yes
Backend Type*: JE Backend ([1])
Base DN for Directory Data: dc=cts,dc=example,dc=com
Option for Populating Database: Option 2 - Only create base entry
Do You Want to Enable SSL: no (may be required for your deployment)
Do You Want to Enable StartTLS: no (may be required for your deployment)
Do You Want To Start The Server: yes
What Would You Like To Do: 1 - Set up server with parameters above

```

* The Backend Type choice is available for OpenDJ 3.0 directory server and later.

3.2.2. Import CTS Files

Once the OpenDJ installation is complete and the instance is operational, import the schema, multivalue, index and container files for CTS as shown in the procedure below.

Procedure 3.2. To Import the CTS Configuration

1. Copy the CTS schema and then add it the repository.

```

$ TOMCAT_OPENAM_WEBAPP=/path/to/tomcat/webapps/openam
$ T=/tmp/ldif
$ rm -rf $T
$ mkdir $T
$ cp $TOMCAT_OPENAM_WEBAPP/WEB-INF/template/ldif/sfha/cts-add-schema.ldif $T/cts-add-schema.ldif

```

If you are using OpenDJ 4.0 or later:

```

$ ./ldapmodify
\
--port 1389
\
--bindDN "cn=Directory Manager"
\
--bindPassword password \
$T/cts-add-schema.ldif

```

If you are using OpenDJ 3.5 or earlier:

```
$ ./ldapmodify
\
--port 1389
\
--bindDN "cn=Directory Manager"
\
--bindPassword password
\
--filename $T/cts-add-schema.ldif
```

The output should be:

```
Processing MODIFY request for cn=schema
MODIFY operation successful for DN cn=schema
```

2. Copy the multivalue file and then add it to the repository.

```
$ cp $TOMCAT_OPENAM_WEBAPP/WEB-INF/template/ldif/sfha/cts-add-multivalue.ldif $T/cts-add-multivalue
.ldif
```

If you are using OpenDJ 4.0 or later:

```
$ ./ldapmodify
\
--port 1389
\
--bindDN "cn=Directory Manager"
\
--bindPassword password \
$T/cts-add-multivalue.ldif
```

If you are using OpenDJ 3.5 or earlier:

```
$ ./ldapmodify
\
--port 1389
\
--bindDN "cn=Directory Manager"
\
--bindPassword password
\
--filename $T/cts-add-multivalue.ldif
```

3. Copy the multivalue index file, and then replace the @DB_NAME@ variable with your repository in the file. Then, add the file to the repository.

```
$ cat $TOMCAT_OPENAM_WEBAPP/WEB-INF/template/ldif/sfha/cts-add-multivalue-indices.ldif \
| sed -e 's/@DB_NAME@/userRoot/' > $T/cts-add-multivalue-indices.ldif
```

If you are using OpenDJ 4.0 or later:

```
$ ./ldapmodify
\
--port 1389
\
--bindDN "cn=Directory Manager"
\
--bindPassword password \
$T/cts-add-multivalue-indices.ldif
```

If you are using OpenDJ 3.5 or earlier:

```
$ ./ldapmodify
\
--port 1389
\
--bindDN "cn=Directory Manager"
\
--bindPassword password
\
--filename $T/cts-add-multivalue-indices.ldif
```

4. Copy the CTS index file, and then replace the @DB_NAME@ variable with your repository in the file. Then, add the file to the repository.

```
$ cat $TOMCAT_OPENAM_WEBAPP/WEB-INF/template/ldif/sfha/cts-indices.ldif \
| sed -e 's/@DB_NAME@/userRoot/' > $T/cts-indices.ldif
```

If you are using OpenDJ 4.0 or later:

```
$ ./ldapmodify
\
--port 1389
\
--bindDN "cn=Directory Manager"
\
--bindPassword password \
$T/cts-indices.ldif
```

If you are using OpenDJ 3.5 or earlier:

```
$ ./ldapmodify
\
--port 1389
\
--bindDN "cn=Directory Manager"
\
--bindPassword password
\
--filename $T/cts-indices.ldif
```

5. Copy the container file, and then replace the `@SM_CONFIG_ROOT_SUFFIX@` variable with the base DN defined during the external OpenDJ installation procedure, for example, `dc=example,dc=com`. Then, add the file to the repository.

```
$ ROOT_SUFFIX="dc=example,dc=com"
$ cat $TOMCAT_OPENAM_WEBAPP/WEB-INF/template/ldif/sfha/cts-container.ldif | sed -e 's/
@SM_CONFIG_ROOT_SUFFIX@/$ROOT_SUFFIX/' > $T/cts-container.ldif
```

If you are using OpenDJ 4.0 or later:

```
$ ./ldapmodify
\
--port 1389
\
--bindDN "cn=Directory Manager"
\
--bindPassword password \
$T/cts-container.ldif
```

If you are using OpenDJ 3.5 or earlier:

```
$ ./ldapmodify
\
--port 1389
\
--bindDN "cn=Directory Manager"
\
--bindPassword password
\
--filename $T/cts-container.ldif
```

The output should be:

```
Processing ADD request for ou=tokens,dc=cts,dc=example,dc=com
ADD operation successful for DN ou=tokens,dc=cts,dc=example,dc=com
Processing ADD request for ou=openam-session,ou=tokens,dc=cts,dc=example,dc=com
ADD operation successful for DN ou=openam-session,ou=tokens,dc=cts,dc=example,dc=com
Processing ADD request for ou=famrecords,ou=openam-session,ou=tokens,dc=cts,dc=example,dc=com
ADD operation successful for DN ou=famrecords,ou=openam-session,ou=tokens,dc=cts,dc=example,dc=com
```

6. If OpenAM is binding to CTS as the Directory Manager user, you can jump to section Section 3.2.4, "CTS Index Import and Build".

To create a non-admin user, follow the instructions in the next section.

3.2.3. Non-Admin User Creation and ACI Import

As a best practice, the use of `cn=Directory Manager` is not recommended. Instead, you can create a new user with limited privileges as shown below.

Procedure 3.3. To Create a Non-Admin User

1. Create an LDIF file called `cts_user.ldif` that defines the CTS non-admin user. The following sample LDIF creates a user called `openam_cts` and assigns the `update-schema`, `subentry-write`, and `password-reset` privileges.

The LDIF file also overrides the default lookthrough limit of 5000 for this non-admin user to unlimited (0) and sets the maximum number of entries returned for a search to 5000 (default, 1000). The `ds-rlim-size-limit: 5000` is arbitrary and can be any value larger than the default maximum number of entries returned for a search, for example, value `>= 1001`. Setting the maximum number of entries for a search to 5000 ensures that the CTS reaper can properly delete returned tokens when large bursts of CTS tokens (`> 5000` per interval between CTS reaping) are returned. For more information on OpenDJ resource limits, see [Setting Resource Limits](#) on the *OpenDJ Administration Guide*.

If there are more than 100K of expired tokens in the CTS, the search from the CTS reaper will be treated as non-indexed and will fail if the non-admin user does not have the `unindexed-search` privilege. Therefore, you should add the `unindexed-search` privilege to the user's entry.

Finally, make sure that you replace the `<password>` tag with your actual password:

```
dn: ou=admins,dc=cts,dc=example,dc=com
objectClass: top
objectClass: organizationalunit
ou: OpenAM Administrator

dn: uid=openam_cts,ou=admins,dc=cts,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: OpenAM Non-Admin-User
sn: OpenAM
userPassword: password
ds-privilege-name: update-schema
ds-privilege-name: subentry-write
ds-privilege-name: password-reset
ds-privilege-name: unindexed-search
ds-rlim-lookthrough-limit: 0
ds-rlim-size-limit: 5000
```


2. Add the new user to the CTS repository. The following example assumes that you are using OpenDJ 4.0 and later:

```
./ldapmodify \
--port 1389 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
cts_user.ldif
```

If you are using OpenDJ 3.5 or earlier:

```
./ldapmodify \
--defaultAdd
--port 1389 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--filename cts_user.ldif
```

The output should be:

```
Processing ADD request for ou=admins,dc=cts,dc=example,dc=com
ADD operation successful for DN ou=admins,dc=cts,dc=example,dc=com
Processing ADD request for uid=openam_cts,ou=admins,dc=cts,dc=example,dc=com
ADD operation successful for DN uid=openam_cts,ou=admins,dc=cts,dc=example,dc=com
```

3. Add a global ACI to allow the `openam_cts` user to modify schema:

```
./dsconfig \
set-access-control-handler-prop \
--no-prompt \
--hostname cts.example.com \
--port 4444 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--add 'global-aci:(target = "ldap:///cn=schema")(targetattr = "attributeTypes ||
objectClasses")(version 3.0; acl "Modify schema"; allow (write)
userdn = "ldap:///uid=openam_cts,ou=admins,dc=cts,dc=example,dc=com");'
```

4. Use `dsconfig` to check that the global ACI has been applied:

```
./dsconfig \
get-access-control-handler-prop \
--hostname cts.example.com \
--port 4444 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--no-prompt \
--property global-aci
```

Verify that the following entry is present:

```
"(target = "ldap:///cn=schema")(targetattr = "attributeTypes || objectClasses")
(version 3.0; acl "Modify schema"; allow (write) userdn =
"ldap:///uid=openam_cts,ou=admins,dc=cts,dc=example,dc=com");)",
```

5. Create an LDIF file called `cts_acis.ldif` to add the ACIs to allow the CTS user to create, search, modify, delete, and allow persistent search to the CTS repository:

```
dn: dc=cts,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="*)(version 3.0;acl "Allow entry search";
allow (search, read)(userdn = "ldap:///uid=openam_cts,ou=admins,dc=cts,dc=example,dc=com");)
aci: (targetattr="*)(version 3.0;acl "Modify entries"; allow (write)
(userdn = "ldap:///uid=openam_cts,ou=admins,dc=cts,dc=example,dc=com");)
aci: (targetcontrol="2.16.840.1.113730.3.4.3")(version 3.0;
acl "Allow persistentsearch";allow (search, read)
(userdn = "ldap:///uid=openam_cts,ou=admins,dc=cts,dc=example,dc=com");)
aci: (version 3.0;acl "Add config entry"; allow (add)(userdn =
"ldap:///uid=openam_cts,ou=admins,dc=cts,dc=example,dc=com");)
aci: (version 3.0;acl "Delete entries"; allow (delete)(userdn =
"ldap:///uid=openam_cts,ou=admins,dc=cts,dc=example,dc=com");)
```

6. Import the ACIs into the CTS repository:

```
./ldapmodify \
--hostname cts.example.com \
--port 1389 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
cts_acis.ldif
```

If you are using OpenDJ 3.5 or earlier:

```
./ldapmodify \
--hostname cts.example.com \
--port 1389 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--filename cts_acis.ldif
```

The output should be:

```
Processing MODIFY request for dc=cts,dc=example,dc=com
MODIFY operation successful for DN dc=cts,dc=example,dc=com
```

3.2.4. CTS Index Import and Build

Procedure 3.4. To Import and Rebuild the CTS Indexes

1. Open the `/tomcat/webapps/openam/WEB-INF/template/ldif/sfha/cts-indices.ldif` file. Apply each index to the CTS repository using the **dsconfig** command. Note that these indexes may require further tuning depending on environmental load testing.

For example, you can apply the first index `coreTokenExpirationDate` as shown below. Then, apply the other indexes individually in the same manner:

```
./dsconfig \
--port 4444 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--backend-name userRoot \
--index-name coreTokenExpirationDate \
--set index-type:ordering \
--trustAll \
--no-prompt
```

Or, you can obtain a copy of a **dsconfig** batch file, which adds all of your indexes to the CTS repository at one time. Copy the `cts-add-indexes.txt` in Appendix B, "*Supported Scripts*", save it locally, then run **dsconfig** in batch mode:

```
./dsconfig \
--port 4444 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--batchFilePath cts-add-indexes.txt \
--trustAll \
--no-prompt
```

2. Rebuild all indexes and then verify them:

```
./rebuild-index \
--port 4444 \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--baseDN "dc=cts,dc=example,dc=com" \
--rebuildAll
--start 0

./verify-index --baseDN "dc=cts,dc=example,dc=com"
```

3. Restart the OpenDJ instance.

3.2.5. CTS Configuration

At this stage, you have successfully set up the external OpenDJ directory server. You must now set up the CTS repository on OpenAM using the AM console.

Procedure 3.5. To Configure CTS

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:

Table 3.1. CTS Token Store Parameters

Parameter	Value	Notes
Store Mode	External Token Store	
Root Suffix	dc=cts,dc=example,d=com	
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:

Table 3.2. External Store Configuration Parameters

Parameter	Value	Notes
SSL/TLS Enabled	False	
Connection String(s)	cts.example.com:1389	
Login ID	uid=openam_cts,ou=admins,dc=cts,dc=example,dc=com	
Password	password	
Heartbeat	10	For production, this value needs to be tuned.

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

3.2.6. 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.

Procedure 3.6. To Test Session High Availability

1. In Chrome, log in to the second OpenAM instance with the `amadmin` user, select the realm, and then click on `sessions`.
2. In Firefox, log in to the first OpenAM instance with a test user.
3. In Chrome, verify that the test user exists in the first OpenAM instance's session list and not in the second instance.
4. Shut down the first OpenAM instance.
5. In Firefox, rewrite the URL to point to the second OpenAM 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 OpenAM instance to complete the testing.

3.3. CTS Backups and OpenDJ Replication Purge Delay

Replication is the process of copying updates between directory servers to help all servers converge to identical copies of directory, token, and session / SAML v2.0 / OAuth 2.0 data. OpenDJ 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 OpenDJ is 3 days. Unless you have configured a separate OpenDJ 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 OpenDJ 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 OpenDJ backups that are older than the `replication-purge-delay` are useless and must be discarded. You can use the OpenDJ **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/openssl/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 data are not lost.

3.4. Managing CTS Tokens

The following properties are associated with token encryption, compression, and token cleanup frequency, which are disabled by default. The properties are as follows:

`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`.

To enable the encryption/compression options, navigate to Configure > Server Defaults > Advanced. On the Advanced page, you will see these entries in the `Property Name` column with the corresponding value in the `Property Value` column. To enable them, change `false` to `true` in the Property Value column associated with the desired property, and click Save.

Note

If you want to enable compression or encryption, you must enable the same property on every OpenAM instance within the site, otherwise they will not function correctly together. You must also restart the servers for the changes to take effect.

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.

3.5. 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.

3.5.1. Reaper Cache Size

OpenAM implements a reaper service that deletes expired tokens from the CTS store.

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

Use of the local reaper cache means fewer searches of the CTS store to determine expired tokens to delete, improving 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 size of the reaper cache is controlled by the `org.forgerock.services.cts.reaper.cache.size` advanced property. The default size is 500000 tokens.

If an OpenAM 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 OpenAM 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.

3.5.2. Queue Size and Timeout

One tuning consideration is to manage the CTS queue size and timeout for efficient throughput. OpenAM makes CTS requests from the following components:

- OpenAM stateful sessions
- OAuth 2.0
- UMA
- Session blacklist (if enabled for stateless sessions)
- SAML v2.0 (if enabled for Security Token Service token validation and cancellation)
- OAuth 2.0 stateless sessions and session 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 freed 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, OpenAM 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 OpenAM 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 Section 8.4.1.2.3, "Tuning LDAP CTS and Configuration Store Settings" in the *Setup and Maintenance Guide*.

3.5.3. Virtual Attributes

OpenDJ supports a number of virtual attributes, which dynamically generate entry values that are not persisted in the token store. By default, the following OpenDJ 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 OpenDJ 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 OpenDJ **entity-tag-virtual-attribute** is required and should never be disabled.

Chapter 4

Securing Installations

This chapter identifies best practices for securing your OpenAM installation.

4.1. Avoiding Obvious Defaults

OpenAM 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 OpenAM (`com.iplanet.am.cookie.name`) and in your policy agent profiles (`com.sun.identity.agents.config.cookie.name`).
- When installing OpenAM, 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.
- Set the OpenAM advanced property `openam.auth.soap.rest.generic.authentication.exception` to `true`. This causes OpenAM to return the same exception both when the user does not exist, and also when the password is not valid.
- Remove the demo user account. For example, if you configure the embedded OpenDJ directory 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 > Subjects > User.
- Set the list of Valid `goto` URL Resources. By default, OpenAM redirects the user to the URL specified in the `goto` and `gotoOnFail` query string parameters supplied to the authentication interface in the login URL.

To increase security against possible phishing attacks through open redirect, you can specify a list of valid URL resources against which OpenAM validates these URLs. OpenAM only redirects a user if the `goto` and `gotoOnFail` URL matches any of the resources specified in this setting. If no setting is present, it is assumed that the `goto` or `gotoOnFail` URL is valid.

To set the Valid `goto` URL Resources, use the OpenAM console, and navigate to Realms > *Realm Name* > Services. Click Add a Service, select Validation Service, and add one or more valid `goto` URLs, and then click Create.

When setting valid `goto` URLs, you can use the `"*"` wildcard, where `"*"` matches all characters except `"?"`. For more specific patterns, use resource names with wildcards as described in the procedure, Section 8.1.3, "Constraining Post-Login Redirects" in the *Authentication and Single Sign-On Guide*.

- Disable module based authentication for all OpenAM 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 OpenAM console, then select Authentication > Settings > Security and clear the Module Based Authentication check box.

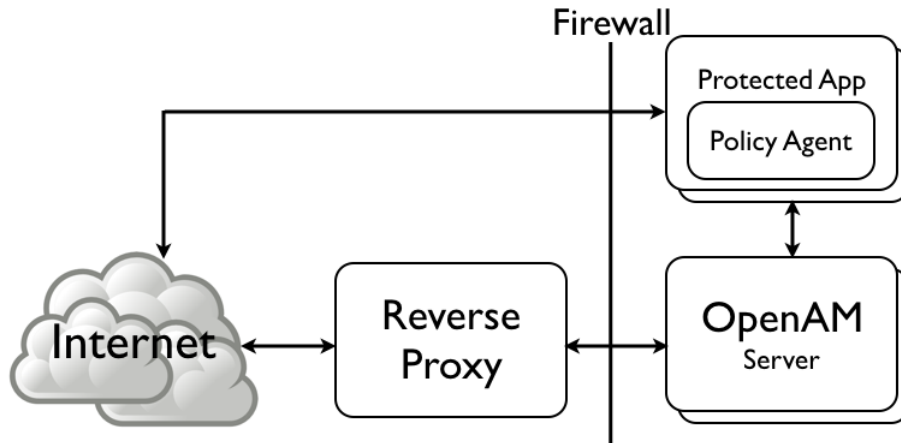
4.2. Protecting Network Access

Anytime users interact with a web service, there are risks. With OpenAM, you can reduce those risks by limiting what is exposed through the firewall using the following strategy:

- Use a reverse proxy in front of OpenAM 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.

Figure 4.1. Exposing Only a Reverse Proxy to the Internet



Architecture protecting your
OpenAM services behind an
Internet-facing reverse proxy

- Leave the deprecated `ssoadm.jsp` page disabled in production. (Advanced property: `ssoadm.disabled=true`).
- 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 [Chapter 5, "Service Endpoints"](#) in the *Reference*.

4.3. Securing Administration

Create realms for your organization(s) and separate administrative users from end users. For instructions, see [Chapter 2, "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 a policy agent profile for each policy agent. See Chapter 4, "Setting Up Policy Agent Profiles" in the *Setup and Maintenance Guide* for instructions.

4.4. 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 OpenAM 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 OpenAM 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 OpenAM and one or more of the following:
 - An external OpenAM configuration store
 - An external CTS token store
 - An external user store
 - An Active Directory repository used for OpenAM authentication with the Active Directory module
 - An LDAP repository used for OpenAM authentication with the LDAP module
 - A repository used for OpenAM 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 Table 1.2, "Security Settings".

- Where possible, use subdomain cookies, and control subdomains in a specific DNS master.

- Use cookie hijacking protection with restricted tokens, where each policy agent uses different SSO tokens for the same user. See Procedure 7.7, "To Protect Against Cookie Hijacking" in the *Authentication and Single Sign-On Guide* for instructions.
- Use your own key, not the `test` key provided with OpenAM, to sign:
 - SAML 2.0 authentication requests, authentication responses, and single logout requests
 - XUI authentication IDs

See Procedure 5.3, "To Change Default test Signing Key" in the *Setup and Maintenance Guide* for instructions.

- 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.

4.4.1. 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 OpenAM. OpenAM's core services and Java EE policy agents depend on the certificates installed for use with the web application container in which they run. OpenAM web policy 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 OpenAM 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 OpenAM.

4.4.2. Using Self-Signed Certificates

You can use either CA or self-signed certificates with OpenAM, 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 OpenAM in a secure container with a self-signed certificate, see [Procedure 4.1, "To Set Up With HTTPS and Self-Signed Certificates"](#). For more information about sharing self-signed certificates among applications, see [Procedure 4.2, "To Share Self-Signed Certificates"](#).

Procedure 4.1. To Set Up With HTTPS and Self-Signed Certificates

The container in which you install OpenAM 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:

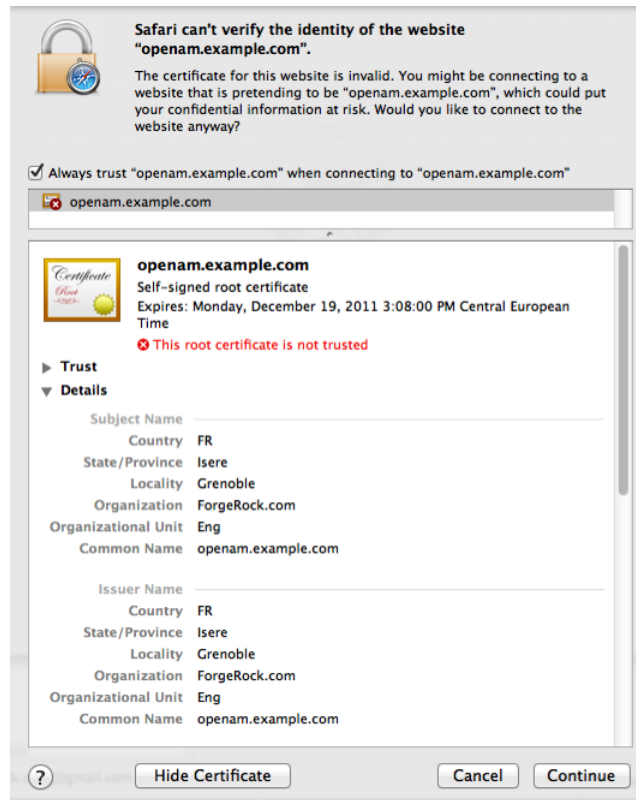
```
<!-- Define a SSL HTTP/1.1 Connector on port 8443
This connector uses the JSSE configuration, when using APR, the
connector should be using the OpenSSL style configuration
described in the APR documentation -->
<!--
-->
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
    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.

Figure 4.2. Unknown Certificate



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 OpenAM.
7. To share the self-signed certificate in your container with other applications or servers, see Procedure 4.2, "To Share Self-Signed Certificates".

Procedure 4.2. To Share Self-Signed Certificates

How you configure the containers where OpenAM 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 OpenAM, OpenAM 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 OpenAM.

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 OpenAM RESTful API without having the client recognize the self-signed server certificate:

```
$ curl \
  "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=AQIC5wM2LY4SfczMax8jeggSiaigB96N0WylLilsd0PUMjY.*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:

```
$ cd /path/to/tomcat/conf/
$ keytool \
  -exportcert \
  -alias openam.example.com \
  -file openam.crt \
  -storetype JCEKS
Enter keystore password:
Certificate stored in file <openam.crt>
```

¹Alternatively, 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`.

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
```

Chapter 5

Removing Installations

This chapter shows you how to uninstall OpenAM.

For instructions on removing OpenAM agents, see the *OpenAM Web Policy Agent User's Guide*, or the *OpenAM Java EE Policy Agent User's Guide*.

Procedure 5.1. To Remove an Instance

After you have deployed and configured OpenAM, you may have as many as four locations where OpenAM files are stored on your system.

Following the steps below removes the OpenAM software and the internal configuration store. If you used an external configuration store, you can remove OpenAM configuration data after removing all the software.

1. Shut down the web application container in which you deployed OpenAM.

```
$ /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 OpenAM by removing the configuration files found in the \$HOME directory of the user running the web application container.

A full uninstall of OpenAM 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 OpenAM configuration is `dc=openam,dc=forgerock,dc=org`.

Note

At this point, you can restart the web container and configure OpenAM anew if you only want to start over with a clean configuration rather than removing OpenAM completely.

3. Undeploy the OpenAM 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 6

Troubleshooting Installations

OpenAM can capture information in debug log files that are useful when troubleshooting OpenAM problems. Section 9.2, "Debug Logging" in the *Setup and Maintenance Guide* describes how to enable debug logging after OpenAM has been started.

It is also possible to capture debug logs while installing OpenAM. This can be useful if you need to troubleshoot an installation problem.

Procedure 6.1. To Troubleshoot an Installation

Follow these steps to capture debug logs while installing OpenAM 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 OpenAM installation. Debug log files containing troubleshooting information appear in the `/path/to/openam/openam/debug` directory.
 4. When you have completed OpenAM installation and no longer need to capture debug logs, stop Tomcat, revert the debug logging options, and restart Tomcat.

Chapter 7

Reference

This reference section covers settings and other information relating to installing OpenAM.

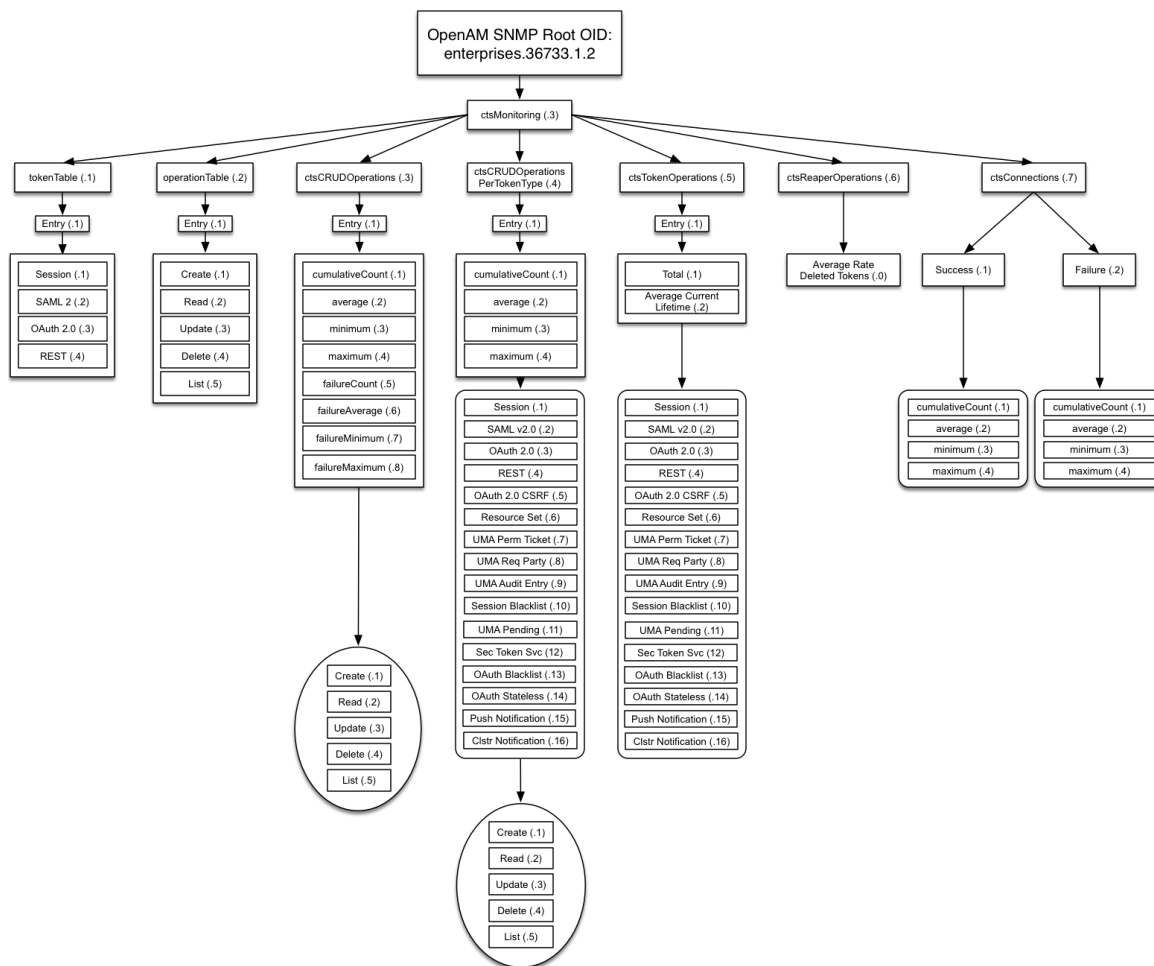
7.1. Core Token Service (CTS) Object Identifiers

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 OpenAM (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.

Figure 7.1. Diagram of CTS OIDs



7.1.1. 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**, OpenAM **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**.

Table 7.1. 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	Resource Set
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 Stateless
enterprises.36733.1.2.3.1.1.15	Push Notification
enterprises.36733.1.2.3.1.1.16	Cluster-wide Notification

7.1.2. 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.

Table 7.2. 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

7.1.3. 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.

7.1.4. 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.

Table 7.3. 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`

Table 7.4. CTS CRUD Operation Table Cumulative Operations

Cumulative Count Operations OID	Data Type	Description
<code>enterprises.36733.1.2.3.3.1.1.1</code>	Counter64	Cumulative count of CREATE operations
<code>enterprises.36733.1.2.3.3.1.1.2</code>	Counter64	Cumulative count of READ operations
<code>enterprises.36733.1.2.3.3.1.1.3</code>	Counter64	Cumulative count of UPDATE operations
<code>enterprises.36733.1.2.3.3.1.1.4</code>	Counter64	Cumulative count of DELETE operations
<code>enterprises.36733.1.2.3.3.1.1.5</code>	Counter64	Cumulative count of LIST operations

Table 7.5. CTS CRUD Operation Table Average Operations (In Period)

Average Number Operations OID	Data Type	Description
<code>enterprises.36733.1.2.3.3.1.2.1</code>	Float2dp	Average number of CREATE operations (in period)
<code>enterprises.36733.1.2.3.3.1.2.2</code>	Float2dp	Average number of READ operations (in period)
<code>enterprises.36733.1.2.3.3.1.2.3</code>	Float2dp	Average number of UPDATE operations (in period)
<code>enterprises.36733.1.2.3.3.1.2.4</code>	Float2dp	Average number of DELETE operations (in period)
<code>enterprises.36733.1.2.3.3.1.2.5</code>	Float2dp	Average number of LIST operations (in period)

Table 7.6. 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)

Table 7.7. 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)

Table 7.8. 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)

Table 7.9. 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)

Table 7.10. 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)

Table 7.11. 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
<code>enterprises.36733.1.2.3.3.1.8.3</code>	Counter64	Maximum number of UPDATE operations failures (in period)
<code>enterprises.36733.1.2.3.3.1.8.4</code>	Counter64	Maximum number of DELETE operations failures (in period)
<code>enterprises.36733.1.2.3.3.1.8.5</code>	Counter64	Maximum number of LIST operations failures (in period)

7.1.5. 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:

Table 7.12. 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:

Table 7.13. 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	Resource Set

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 Stateless
15	Push Notification
16	Cluster-wide Notification

The final OID suffix specifies an operation:

Table 7.14. 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.

Table 7.15. 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)

7.1.6. 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.

Table 7.16. 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 Resource Set 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 Stateless 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

Table 7.17. 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 Resource Set 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 Stateless 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

7.1.7. CTS Reaper Run Information

The CTS reaper deletes unused or expired tokens. Unless OpenAM is in a shutdown cycle, the CTS reaper is designed to run continuously. By default, the CTS reaper runs in fixed intervals, unless OpenAM 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.

7.1.8. CTS Connection Factory OIDs

Every request for a CTS token is a request to the `CTSTokenConnectionFactory`. Such requests can either succeed or fail. The following OIDs provide measures for both such connections. The `CTSTokenConnectionFactory` 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 `CTSTokenConnectionFactory` 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 `CTSTokenConnectionFactory`.

Table 7.18. CTSTokenConnectionFactory, Successful Connections

Successes, CTSTokenConnectionFactory	Data Type	Description
<code>enterprises.36733.1.2.3.7.1.1.0</code>	<code>Counter64</code>	Cumulative number of successful connections
<code>enterprises.36733.1.2.3.7.1.2.0</code>	<code>Float2dp</code>	Average number of successful connections (in period)
<code>enterprises.36733.1.2.3.7.1.3.0</code>	<code>Counter64</code>	Minimum number of successful connections (in period)
<code>enterprises.36733.1.2.3.7.1.4.0</code>	<code>Counter64</code>	Maximum number of successful connections (in period)

Table 7.19. CTSTokenConnectionFactory, Failed Connections

Failures, CTSTokenConnectionFactory	Data Type	Description
<code>enterprises.36733.1.2.3.7.2.1.0</code>	<code>Counter64</code>	Cumulative number of failed connections
<code>enterprises.36733.1.2.3.7.2.2.0</code>	<code>Float2dp</code>	Average number of failed connections (in period)

Failures, CTSCConnectionFactory	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)

7.2. Command-line Tool Reference

Name

configurator.jar — install or upgrade OpenAM using a configuration file

Synopsis

```
configurator.jar {options}
```

Description

This executable .jar file, openam-configurator-tool-14.0.0.jar, lets you perform silent installation, configuring a deployed OpenAM server by applying settings from a configuration file.

Options

The following options are supported.

-f | --file *configuration-file*

Configure a deployed OpenAM 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 OpenAM, and using the hints in this section, or the comments included in the file.

Server Properties

These properties pertain to the OpenAM server instance.

SERVER_URL

URL to the web container where you want OpenAM to run, such as **http://openam.example.com:8080**

DEPLOYMENT_URI

URI where you want to deploy OpenAM on the web container, such as **/openam**

BASE_DIR

Configuration directory where OpenAM 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 OpenAM 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 `06QWwHP04os+zEz3Nqn/2daAYWyIFE32`. If left blank, installing OpenAM 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 OpenAM administrator user `amadmin`, which must be at least 8 characters in length and must match that of other servers in a multiserver deployment

AMLDAPUSERPASSWD

Password of the default policy agent `UrlAccessAgent`, which must be at least 8 characters in length and must not be the same as the value of `ADMIN_PWD`

COOKIE_DOMAIN

Name of the trusted DNS domain OpenAM 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-14.0.0.jar` file.

Configuration Store Properties

These properties pertain to the directory server where OpenAM stores its configuration.

DATA_STORE

Type of the configuration data store. The value `embedded` means set up OpenAM with an embedded, OpenDJ based configuration store. The value `dirServer` means an external directory server, such as OpenDJ, or Sun Java System Directory Server. If you set this to `dirServer`, and the configuration

store contains the configuration of other OpenAM 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 OpenDJ 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 OpenAM stores user profiles. If you do not include these properties, or you leave these properties commented out, then OpenAM 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

- **LDAPv3For0DSEE**: 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 OpenAM 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 OpenAM 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 OpenAM server you are installing, such as 58989

DS_EMB_REPL_HOST2

Host name of an existing OpenAM 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 OpenAM server, such as 4444

DS_EMB_REPL_REPLPORT2

Replication port number for the configuration store directory server used by the existing OpenAM server, such as 50899

existingserverid

Full URL of the existing OpenAM server, such as `http://server1.example.com:8080/openam`

Upgrade Configuration File

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

Upgrade Properties

SERVER_URL

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

DEPLOYMENT_URI

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

ACCEPT_LICENSES

Optional boolean property that can be set to always auto-accept the software license agreement and suppress displaying the license acceptance screen to the user. A value of `true` auto-accepts the license; any other value will be assumed to equal `false`, resulting in the presentation of the license. Default value is `false`. This property takes precedence over the `--acceptLicense` option, which can also be passed in to the application with the `openam-configurator-tool-14.0.0.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
AMLDAPUSERPASSWD=secret12
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=06QwwHP04os+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 OpenAM server.

```
SERVER_URL=https://openam.example.com:8080
DEPLOYMENT_URI=/openam
ACCEPT_LICENSE=true
```

The following example uses a configuration file with the `--acceptLicense` option on the command line.

```
$ java \
-jar openam-configurator-tool-14.0.0.jar \
-f config.file \
--acceptLicense
```

Appendix A. Getting Support

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

A.1. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

- The ForgeRock [Knowledge Base](#) offers a large and increasing number of up-to-date, practical articles that help you deploy and manage ForgeRock software.
- ForgeRock core documentation, such as this document, aims to be technically accurate and complete with respect to the software documented. It is visible to everyone and covers all product features and examples of how to use them.

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

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

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

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

A.2. Joining the ForgeRock Community

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

A.3. Getting Support and Contacting ForgeRock

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

ForgeRock has staff members around the globe who support our international customers and partners. For details, visit <https://www.forgerock.com>, or send an email to ForgeRock at info@forgerock.com.

Appendix B. Supported Scripts

You can use the following script, `cts-add-indexes.txt`, to add the set of CTS indexes for your CTS data store using **dsconfig** batch mode:

```
# cts-add-indexes.txt
#
# DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS HEADER.
#
# Copyright (c) 2011-2016 ForgeRock AS. All Rights Reserved
#
# The contents of this file are subject to the terms
# of the Common Development and Distribution License
# (the License). You may not use this file except in
# compliance with the License.
#
# You can obtain a copy of the License at
# http://forgerock.org/license/CDDLv1.0.html
# See the License for the specific language governing
# permission and limitations under the License.
#
# When distributing Covered Code, include this CDDL
# Header Notice in each file and include the License file
# at http://forgerock.org/license/CDDLv1.0.html
# If applicable, add the following below the CDDL Header,
# with the fields enclosed by brackets [] replaced by
# your own identifying information:
# "Portions Copyrighted [year] [name of copyright owner]"
#

# dsconfig batch file to add CTS indexes
# 1. Save this file locally.
# 2. On OpenDJ server, run:
#    dsconfig -p 4444 -D "cn=Directory Manager" -w password \
#    -F cts-add-indexes.txt -X -n
```

```
create-backend-index --backend-name userRoot --index-name coreTokenExpirationDate \  
  --set index-type:ordering  
create-backend-index --backend-name userRoot --index-name coreTokenUserId \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenString01 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenString02 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenString03 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenString05 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenString08 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenString09 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenString10 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenString14 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenString15 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenInteger01 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenInteger02 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenInteger03 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenInteger04 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenInteger05 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenInteger06 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenInteger07 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenInteger08 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenInteger09 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenInteger10 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenDate01 \  
  --set index-type:ordering  
create-backend-index --backend-name userRoot --index-name coreTokenDate02 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenDate03 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenDate04 \  
  --set index-type:equality  
create-backend-index --backend-name userRoot --index-name coreTokenDate05 \  
  --set index-type:equality
```

Appendix C. Supported LDIF Files

OpenAM installation deploys several LDIF files that can be used to create the schemas required by OpenAM 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, OpenDJ, Oracle Unified Directory, and IBM Tivoli Directory Server.

The following tables provide descriptions for each LDIF file:

Table C.1. 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.

Table C.2. Microsoft Active Directory Lightweight Directory Services LDIF Files

LDIF File	Description
adam_dashboard.ldif	LDIF to support the dashboard service.

LDIF File	Description
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.

Table C.3. Oracle Directory Server Enterprise Edition LDIF Files

LDIF File	Description
amsdk_plugin	Folder containing the OpenAM 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 repository indexes.
odsee_user_schema.ldif	LDIF for the user repository schema.
odsee_userinit.ldif	LDIF for the setting up user session initialization.

Table C.4. OpenDJ 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_config_schema.ldif	LDIF for the OpenDJ configuration schema.
opendj_dashboard.ldif	LDIF to support the dashboard service.
opendj_deviceprint.ldif	LDIF to support the device print service.

LDIF File	Description
opendj_embinit.ldif	LDIF for the OpenDJ user management and SMS/configuration datastore schema for embedded OpenDJ 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_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 add labels for the UMA service resource sets.
opendj_uma_resource_sets.ldif	LDIF to add resource sets to the UMA service resource.
opendj_user_index.ldif	LDIF for the user repository indexes.
opendj_user_schema.ldif	LDIF for the user repository schema.
opendj_userinit.ldif	LDIF for the setting up user session initialization.

Table C.5. 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.

Glossary

Access control	Control to grant or to deny access to a resource.
Account lockout	The act of making an account temporarily or permanently inactive after successive authentication failures.
Actions	Defined as part of policies, these verbs indicate what authorized subjects can do to resources.
Advice	In the context of a policy decision denying access, a hint to the policy enforcement point about remedial action to take that could result in a decision allowing access.
Agent administrator	User having privileges only to read and write policy agent profile configuration information, typically created to delegate policy agent profile creation to the user installing a policy agent.
Agent authenticator	Entity with read-only access to multiple agent profiles defined in the same realm; allows an agent to read web service profiles.
Application	<p>In general terms, a service exposing protected resources.</p> <p>In the context of OpenAM policies, the application is a template that constrains the policies that govern access to protected resources. An application can have zero or more policies.</p>
Application type	<p>Application types act as templates for creating policy applications.</p> <p>Application types define a preset list of actions and functional logic, such as policy lookup and resource comparator logic.</p>

	Application types also define the internal normalization, indexing logic, and comparator logic for applications.
Attribute-based access control (ABAC)	Access control that is based on attributes of a user, such as how old a user is or whether the user is a paying customer.
Authentication	The act of confirming the identity of a principal.
Authentication chaining	A series of authentication modules configured together which a principal must negotiate as configured in order to authenticate successfully.
Authentication level	Positive integer associated with an authentication module, usually used to require success with more stringent authentication measures when requesting resources requiring special protection.
Authentication module	OpenAM authentication unit that handles one way of obtaining and verifying credentials.
Authorization	The act of determining whether to grant or to deny a principal access to a resource.
Authorization Server	In OAuth 2.0, issues access tokens to the client after authenticating a resource owner and confirming that the owner authorizes the client to access the protected resource. OpenAM can play this role in the OAuth 2.0 authorization framework.
Auto-federation	Arrangement to federate a principal's identity automatically based on a common attribute value shared across the principal's profiles at different providers.
Bulk federation	Batch job permanently federating user profiles between a service provider and an identity provider based on a list of matched user identifiers that exist on both providers.
Circle of trust	Group of providers, including at least one identity provider, who have agreed to trust each other to participate in a SAML v2.0 provider federation.
Client	In OAuth 2.0, requests protected web resources on behalf of the resource owner given the owner's authorization. OpenAM can play this role in the OAuth 2.0 authorization framework.
Conditions	<p>Defined as part of policies, these determine the circumstances under which which a policy applies.</p> <p>Environmental conditions reflect circumstances like the client IP address, time of day, how the subject authenticated, or the authentication level achieved.</p>

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

Metadata	Federation configuration information for a provider.
Policy	Set of rules that define who is granted access to a protected resource when, how, and under what conditions.
Policy Agent	Agent that intercepts requests for resources, directs principals to OpenAM for authentication, and enforces policy decisions from OpenAM.
Policy Administration Point (PAP)	Entity that manages and stores policy definitions.
Policy Decision Point (PDP)	Entity that evaluates access rights and then issues authorization decisions.
Policy Enforcement Point (PEP)	Entity that intercepts a request for a resource and then enforces policy decisions from a PDP.
Policy Information Point (PIP)	Entity that provides extra information, such as user profile attributes that a PDP needs in order to make a decision.
Principal	<p>Represents an entity that has been authenticated (such as a user, a device, or an application), and thus is distinguished from other entities.</p> <p>When a Subject successfully authenticates, OpenAM associates the Subject with the Principal.</p>
Privilege	In the context of delegated administration, a set of administrative tasks that can be performed by specified subjects in a given realm.
Provider federation	Agreement among providers to participate in a circle of trust.
Realm	<p>OpenAM unit for organizing configuration and identity information.</p> <p>Realms can be used for example when different parts of an organization have different applications and user data stores, and when different organizations use the same OpenAM deployment.</p> <p>Administrators can delegate realm administration. The administrator assigns administrative privileges to users, allowing them to perform administrative tasks within the realm.</p>
Resource	<p>Something a user can access over the network such as a web page.</p> <p>Defined as part of policies, these can include wildcards in order to match multiple actual resources.</p>
Resource owner	In OAuth 2.0, entity who can authorize access to protected web resources, such as an end user.

Resource server	In OAuth 2.0, server hosting protected web resources, capable of handling access tokens to respond to requests for such resources.
Response attributes	Defined as part of policies, these allow OpenAM to return additional information in the form of "attributes" with the response to a policy decision.
Role based access control (RBAC)	Access control that is based on whether a user has been granted a set of permissions (a role).
Security Assertion Markup Language (SAML)	Standard, XML-based language for exchanging authentication and authorization data between identity providers and service providers.
Service provider (SP)	Entity that consumes assertions about a principal (and provides a service that the principal is trying to access).
Session	The interval that starts with the user authenticating through OpenAM and ends when the user logs out, or when their session is terminated. For browser-based clients, OpenAM manages user sessions across one or more applications by setting a session cookie. See also Stateful session and Stateless session .
Session high availability	Capability that lets any OpenAM server in a clustered deployment access shared, persistent information about users' sessions from the CTS token store. The user does not need to log in again unless the entire deployment goes down.
Session token	Unique identifier issued by OpenAM after successful authentication. For a Stateful session , the session token is used to track a principal's session.
Single log out (SLO)	Capability allowing a principal to end a session once, thereby ending her session across multiple applications.
Single sign-on (SSO)	Capability allowing a principal to authenticate once and gain access to multiple applications without authenticating again.
Site	<p>Group of OpenAM servers configured the same way, accessed through a load balancer layer.</p> <p>The load balancer handles failover to provide service-level availability. Use sticky load balancing based on <code>amlbcookie</code> values to improve site performance.</p> <p>The load balancer can also be used to protect OpenAM services.</p>
Standard metadata	Standard federation configuration information that you can share with other access management software.
Stateful session	An OpenAM session that resides in the Core Token Service's token store. Stateful sessions might also be cached in memory on one or

more OpenAM servers. OpenAM tracks stateful sessions in order to handle events like logout and timeout, to permit session constraints, and to notify applications involved in SSO when a session ends.

Stateless session

An OpenAM session for which state information is encoded in OpenAM and stored on the client. The information from the session is not retained in the CTS token store. For browser-based clients, OpenAM sets a cookie in the browser that contains the session information.

Subject

Entity that requests access to a resource

When a subject successfully authenticates, OpenAM associates the subject with the [Principal](#) that distinguishes it from other subjects. A subject can be associated with multiple principals.

User data store

Data storage service holding principals' profiles; underlying storage can be an LDAP directory service, a relational database, or a custom [IdRepo](#) implementation.

Web policy agent

Native library installed in a web server that acts as a policy agent with policies based on web page URLs.