



Installation Guide

/ ForgeRock Access Management 7.1.4

Latest update: 7.1.4

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

Copyright © 2011-2021 ForgeRock AS.

Abstract

Guide showing you how to install ForgeRock® Access Management (AM). ForgeRock Access Management provides intelligent authentication, authorization, federation, and single sign-on functionality.



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.

© Copyright 2010-2020 ForgeRock, Inc. All rights reserved. ForgeRock is a registered trademark of ForgeRock, Inc. Other marks appearing herein may be trademarks of their respective owners.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, and distribution. No part of this product or document may be reproduced in any form by any means without prior written authorization of ForgeRock and its licensors, if any.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESSED OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

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@gnome.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.

FontAwesome Copyright

Copyright (c) 2017 by Dave Gandy, <https://fontawesome.com/>.

This Font Software is licensed under the SIL Open Font License, Version 1.1. See <https://opensource.org/licenses/OFL-1.1>.

Table of Contents

Overview	iv
1. Downloading AM	1
2. Customizing the AM WAR File	3
Enabling RSA SecurID Support	3
3. Preparing the Environment	4
Preparing an FQDN	4
Preparing a Java Environment	5
Setting Maximum File Descriptors and Processes Per User	8
4. Preparing a Web Application Container	10
Preparing Apache Tomcat	10
Preparing for JBoss and WildFly	12
Preparing IBM WebSphere	14
Configuring AM's Container for HTTPS	16
Preparing AES Key Wrap Encryption	20
5. Deploying AM	23
6. Preparing External Stores	25
Preparing a Truststore	27
Preparing Configuration Stores	29
Preparing Identity Repositories	30
Supported LDIF Files	37
7. Installing Instances	41
Installing an Instance	42
Configuring Sites and Adding Servers to Sites	54
Installing Silently	58
Deleting and Redeploying AM	60
8. Starting Instances	62
Overriding Startup Settings	62
9. Setting Up Administration Tools	68
10. Next Steps	72
11. Uninstall AM	74
12. Troubleshooting	76
Glossary	77




Overview

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

Unless you are planning a demo or test installation, read the [Release notes](#) before you get started.

Perform the following tasks to install AM in your environment:

Quick Start

 <p>Prepare the Environment for Deployment</p> <p>Learn about the prerequisites for deploying Access Management software.</p>	 <p>Prepare the Java Container</p> <p>Prepare the Java container of your choosing to deploy AM.</p>	 <p>Install AM</p> <p>Install a single instance, or install multiple instances, and add them to an AM site for high-availability.</p>
 <p>Prepare External Stores</p> <p>Learn how to prepare external identity, configuration, and policy/application stores.</p>	 <p>Remove Installations</p> <p>Learn how to uninstall AM if you want to start over with a clean configuration.</p>	 <p>Troubleshoot Installations</p> <p>Follow the steps to capture debug logs in order to troubleshoot or repair an AM installation problem.</p>

About ForgeRock Identity Platform™ Software

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

Chapter 1

Downloading AM

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

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

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

```
$ cd ~/Downloads
$ mkdir openam ; cd openam
$ unzip ~/Downloads/AM-7.1.4.zip
```

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

Distribution Files

File	Description
<code>AM-7.1.4.war</code>	The distribution <code>.war</code> file includes the core server code with an embedded DS server. 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 AM installation directory, from environment variables, or from a combination of the two. This file is also available to download individually.
<code>AM-crypto-tool-7.1.4.war</code>	AM provides a utility with some cryptographic functionality used for creating Docker images. This utility is strictly for future use, and is not currently supported.
<code>AM-Soap-STs-Server-7.1.4.war</code>	AM provides a SOAP-based security token service (STS) server that issues tokens based on the WS-Security protocol ^a .
<code>AM-SSOAdminTools-5.1.3.19.zip</code>	AM provides an <code>ssoadm</code> command-line tool that allows administrators to configure and maintain AM as well as create their own configuration scripts. The <code>zip</code> distribution file contains

File	Description
	binaries, properties file, script templates, and setup scripts for UNIX and windows servers.
AM-SSOConfiguratorTools-5.1.3.19.zip	AM provides configuration and upgrade tools for installing and maintaining your server. The zip distribution file contains libraries, legal notices, and supported binaries for these configuration tools. Also, you can view example configuration and upgrade properties files that can be used as a template for your deployments.
Config-Upgrader-7.1.4.zip	AM provides a configuration file upgrade tool. For more information on converting configuration files for import into AM, see the README.md file in the Config-Upgrader-7.1.4.zip file.
Fedlet-7.1.4.zip	AM provides an AM Fedlet, a light-weight SAML v2.0 service provider. The Fedlet lets you set up a federated deployment without the need of a fully-featured service provider.
IDPDiscovery-7.1.4.war	AM provides an IDP Discovery Profile (SAMLv2 binding profile) for its IDP Discovery service. The profile keeps track of the identity providers for each user.
sample-trees-7.1.4.zip	Clean installs of AM with an embedded data store provide ready-made sample authentication trees to demonstrate how they can be put together. These sample trees are not installed by default on installs of AM with an external configuration store, or if you are upgrading an existing instance of AM. The sample-trees-7.1.4.zip file contains the sample trees in JSON files, ready for import by <i>Amster</i> command-line interface. For information on importing files by using <i>Amster</i> , see Importing Configuration Data in the <i>Amster 7.1 User Guide</i> .
Truststore-Utility-7.1.4.zip	AM provides a utility to help with creating a trust store for use with web authentication. See the readme.md in the ZIP file for instructions, and " <i>MFA: Web Authentication (WebAuthn)</i> " in the <i>Authentication and Single Sign-On Guide</i> for more information.

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

Chapter 2

Customizing the AM WAR File

The most basic installations of AM do not require you to customize the AM WAR file. If you need to customize the AM extension points, add the new classes to the WAR file before deploying it.

Tip

To customize the secure cookie filter, see "Managing the Secure Cookie Filter" in the *Security Guide*.

You can also customize the AM user pages and package them into the WAR file. For more information, see *UI Customization Guide*

Enabling RSA SecurID Support

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

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

```
$ mkdir /tmp/openam
$ cd /tmp/openam/
$ jar -xf ~/Downloads/openam/AM-7.1.4.war
```

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

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

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

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

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

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

Chapter 3

Preparing the Environment

Perform the following tasks to prepare your environment for AM:



Prepare an FQDN

AM relies on browser cookies, which are returned based on the domain name. Therefore, you need to assign an FQDN to your AM instances.



Prepare a Java Environment

As a Java application, AM requires a JDK to run. Ensure that you configure the recommended settings to avoid performance issues.



Configure Maximum File Descriptors

The DS configuration store, CTS store, and the application stores require a number of file descriptors to manage concurrent connections to AM.

Tip

For more information about supported operating systems and Java requirements, see [Operating systems and Java](#).

Preparing an FQDN

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

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

Important

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

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

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

Preparing a Java Environment

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

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.

The following table summarizes the high-level tasks required to configure your Java environment:

Task	Resources
<p>Prepare the JDK of your choosing for AM</p> <p>The suggestions in these sections pertain to AM deployments with the following characteristics:</p> <ul style="list-style-type: none"> The deployment has a dedicated DS server for the Core Token Service. The host running this directory server is a high-end server with a large amount of memory and multiple CPUs. The AM server is configured to use CTS-based sessions. 	<ul style="list-style-type: none"> "Settings For Oracle Java Environments" "Settings For IBM Java Environments" "Settings for OpenJDK Java Environment"
<p>Tune the JVM for AM</p> <p>ForgeRock provides guidance on how to tune the JVM for production, but you still need to tune it for garbage collection based on your environment.</p>	<ul style="list-style-type: none"> "Tuning JVM Settings"

Tip

To configure JVM properties for proxy support, see "Configuring AM for Outbound Communication" in the *Security Guide*.

Settings For Oracle Java Environments

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

-Xmx1g (minimum)

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

`-XX:MetaspaceSize=256m`

Set the metaspace memory size to 256 MB.

`-XX:MaxMetaspaceSize=256m`

Set the maximum metaspace memory size to 256 MB.

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

Settings For IBM Java Environments

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

`-DamCryptoDescriptor.provider=IBMJCE`

`-DamKeyGenDescriptor.provider=IBMJCE`

Use the IBM Java Cryptography Extensions.

`-Xmx1g` (**minimum**)

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

Settings for OpenJDK Java Environment

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

`-Xmx1024m` (**minimum**)

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

`-XX:MetaspaceSize=256m`

Set the initial metadata space size to 256 MB.

Tuning JVM Settings

This section gives some initial guidance on configuring the JVM for running AM when the deployment has a dedicated CTS token store, and AM is configured to use CTS-based sessions.

These settings provide a strong foundation to the JVM before a more detailed garbage collection tuning exercise, or as best practice configuration for production:

Heap Size Settings

JVM Parameters	Suggested Value	Description
<code>-Xms</code> & <code>-Xmx</code>	At least 1 GB (2 GB with embedded DS), in production environments at least 2 GB to 3 GB. This setting depends on the available physical memory, and on whether a 32- or 64-bit JVM is used.	-
<code>-XX:MetaspaceSize</code> & <code>-XX:MaxMetaspaceSize</code>	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.

Security Settings

JVM Parameters	Suggested Value	Description
<code>-Dhttps.protocols</code>	<code>TLSv1.2</code>	Controls the protocols used for outbound HTTPS connections from AM. Specify one or more of the following values, separated by commas: <ul style="list-style-type: none"> • TLSv1.2 • TLSv1.3 This setting applies only to Sun/Oracle Java environments.

JVM Parameters	Suggested Value	Description
<code>-Dorg.forgerock.openam.ldap.secure.protocol.version</code>	TLsv1.2	<p>Controls the protocol AM 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"> • TLsv1.2 • TLsv1.3 <p>This setting overrides the default server value. For details, refer to <i>Advanced Properties</i> in the <i>Reference</i></p>

Garbage Collection Settings

JVM Parameters	Suggested Value	Description
<code>-verbose:gc</code>	-	Verbose garbage collection reporting.
<code>-Xlog:gc*</code>	<code>-Xlog:gc=info:file=\$CATALINA_HOME/logs/gc-info.log</code>	Logs detailed information about garbage collection. When using the <code>-Xlog:gc</code> option, you can also specify the level, and output file.
<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.
<code>-XX:+PrintClassHistogram</code>	-	Prints a heap histogram when the JVM receives a SIGTERM signal.

Setting Maximum File Descriptors and Processes Per User

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

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

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

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

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

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

Important

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

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

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

For more information about setting up your environment for DS, refer to *Operating Systems* in the *DS 7.1 Release Notes*.

Chapter 4

Preparing a Web Application Container

As a Java application, AM must be deployed in a Java container. For a list of supported containers, refer to [Application containers](#).

The following table summarizes the high-level tasks required to prepare your container:

Task	Resources
Prepare the Container for AM Follow the instructions pertaining to your container.	<ul style="list-style-type: none"> "Preparing Apache Tomcat" "Preparing for JBoss and WildFly" "Preparing IBM WebSphere"
Secure the Container There are many ways to deploy and configure your environment for AM, but ForgeRock recommends that you enforce HTTPS connections to AM. If a Java Security Manager is enabled for your web application container, add permissions before installing AM.	<ul style="list-style-type: none"> "Configuring AM's Container for HTTPS" "Preparing AES Key Wrap Encryption"

Preparing Apache Tomcat

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

JVM start up

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

Set a `CATALINA_OPTS` environment variable with the appropriate tuning for your environment. For example, add the following in the `$(CATALINA_BASE)/bin/setenv.sh` file:

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

Some versions of Internet Explorer and Microsoft Edge support the `Expires` header attribute instead of the `Max-Age` header attribute, which may cause SAML v2.0 and agent logout sequences to fail.

If you have set the `org.apache.catalina.STRICT_SERVLET_COMPLIANCE` Tomcat property to `true`, add the `org.apache.tomcat.util.http.ServerCookie.ALWAYS_ADD_EXPIRE` property in the `$CATALINA_BASE/bin/setenv.sh` file, to add the `Expires` attribute to the headers:

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

Slashes in Resource Names

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

In development environments, a possible workaround is to configure Tomcat to allow encoded slash characters by adding the `org.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH=true` property to the `CATALINA_OPTS` variable; for example:

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

Warning

Do *not* enable `org.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH` when running AM in production as it introduces a security risk.

For details, refer to [How do I safely enable the `org.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH` setting in AM/OpenAM \(All Versions\)?](#) in the *ForgeRock Knowledge Base*.

Cookie Domains

Set the cookie domain name value to an empty string (for *host-only* cookies) or to any non-top level domain (for *domain* cookies). For example, if you install AM on `openam.example.com`, you can set the cookie domain name to `example.com`.

Note

Because host-only cookies are more secure than domain cookies, you *should* use host-only cookies unless you have a good business case for using domain cookies.

Refer to "To Configure an Instance" to configure the cookie domain during installation.

Encoding and Security

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

Set the `sslProtocol` property to `TLS` to disable 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" />
```

When you have finished setting up Apache Tomcat, ForgeRock recommends that you enforce HTTPS connections to AM. For more information, refer to "Configuring AM's Container for HTTPS".

Preparing for JBoss and WildFly

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

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

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

To Prepare JBoss or WildFly

1. Stop JBoss or WildFly.
2. The default JVM settings do not allocate sufficient memory to AM. This step shows one method that you can use to modify the JVM settings. For other methods, refer to the [JBoss Application Server Documentation](#) 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 evaluating AM and using the embedded version of DS, the minimum heap size may be higher. For details on the JVM options to use, refer to "Preparing a Java Environment".

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

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

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


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

3. Edit the WildFly configuration to allow HTTP connections from any IP address.

In the `/path/to/wildfly/standalone/configuration/standalone.xml` file, locate the `<interface name="public">` interface (around line 512 of the file) and change the value to `<any-address/>`:

```
<interface name="public">
  <any-address/>
</interface>
```

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

- a. Ensure the Wildfly server is running.
- b. Go to the Wildfly Path.
- c. In the `$JBOSS_HOME/bin` directory, run the `jboss-cli.sh` script file:

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

- d. Type "connect" to connect to the server.
- e. Enable use of the equals (=) symbol in cookies by running the following command:

For example:

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

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

To Prepare for JBoss and WildFly

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

1. If you have not already done so, create a temporary directory and expand the `AM-7.1.4.war` file. For example:

```
$ cd /tmp
$ mkdir /tmp/openam ; cd /tmp/openam
$ jar xvf ~/Downloads/AM-7.1.4.war
```

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

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

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

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

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

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

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

Preparing IBM WebSphere

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

1. Update the JVM options as described in "Preparing a Java Environment".
2. Customize the `AM-7.1.4.war` file as described in "To Prepare for IBM WebSphere".
3. After deploying AM, configure WebSphere as described in "To Prepare WebSphere".

To Prepare for IBM WebSphere

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

Change the AM `war` file as follows:

Note

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

1. Create a temporary directory and expand the `AM-7.1.4.war` file. For example:

```
$ cd /tmp
$ mkdir /tmp/openam ; cd /tmp/openam
$ jar xvf ~/Downloads/AM-7.1.4.war
```

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

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

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

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

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

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

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

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

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

4. Rebuild the `AM-7.1.4.war` file:

```
$ jar cvf ../AM-7.1.4.war *
```

To Prepare WebSphere

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

1. Load classes from AM bundled libraries before loading classes from libraries delivered with WebSphere:
 - a. In the WebSphere administration console, navigate to Application > Application Type > WebSphere enterprise applications > *AM Name* > Class loading and update detection.
 - b. Set Class loader order > Classes loaded with local class loader first (parent last).
 - c. Ensure that the value of the *WAR class loader policy* property is set to the default value: **Class loader for each WAR file in application**.
 - d. Save your work.
2. (Optional) If your environment uses SOAP, perform the following steps to add SOAP-related properties to the JVM:
 - a. In the WebSphere administration console, select Servers.
 - b. Expand Server Type, and select WebSphere application servers.
 - c. Select your WebSphere server name.
 - d. Expand Java and Process Management, and select Process Definition.
 - e. Under the Additional Properties section, select Java Virtual Machine.
 - f. Locate the Generic JVM argument box and add the following properties:

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

- g. Save your work.

Configuring AM's Container for HTTPS

There are many ways to deploy and configure your environment for AM, but we recommend that you enforce HTTPS connections to AM. For more information about securing AM, see the Security Guide.

The container where AM runs requires a certificate in order to set up secure connections. You can install CA-signed or self-signed certificates in the container where AM runs. Remember that you must configure your applications to trust your self-signed certificates.

The following is an example about how to configure Apache Tomcat for HTTPS:

To Configure Apache Tomcat for HTTPS

There are several ways of completing the tasks ahead, and it is beyond the scope of this document to explore them all. If this procedure does not suit your environment, refer to your CA vendor documentation, the **keytool** command documentation, or Java container documentation for more information.

1. Stop Apache Tomcat.
2. Ensure you have a keystore containing the appropriate certificates:
 - a. (Optional) If you have CA certificates, import them in a new keystore using the **keytool** command. For example, if you have root, intermediate, and primary certificates, import them in the same keystore you used when generating your certificate signing request (CSR):

```
$ keytool -importcert -alias root -file myrootCA.cert \  
-keystore /path/to/tomcat_keystore.pfx -storetype PKCS12  
  
$ keytool -importcert -alias intermediate -file myintCA.cert \  
-keystore /path/to/tomcat_keystore.pfx -storetype PKCS12  
  
$ keytool -importcert -alias mysite.example.com -file mypriCA.cert \  
-keystore /path/to/tomcat_keystore.pfx -storetype PKCS12
```

- b. If you need a self-signed certificate, create a new self-signed key pair with the **keytool** command. For example:

```
$ cd /path/to/tomcat/conf/  
$ keytool -genkey -alias openam.example.com -storetype PKCS12 -keyalg RSA -validity 730 \  
-keysize 2048 -keystore tomcat_keystore.pfx -dname 'CN=openam.example.com' -ext  
'san=dns:openam.example.com'
```

3. Create an SSL connector configuration in Apache Tomcat's `conf/server.xml` configuration file, and specify your keystore file, type, and password.

Note that there are different types of SSL connectors, and that implementation details may change across Tomcat versions. This example creates a JSSE connector in Tomcat as provided as part of the Java runtime:

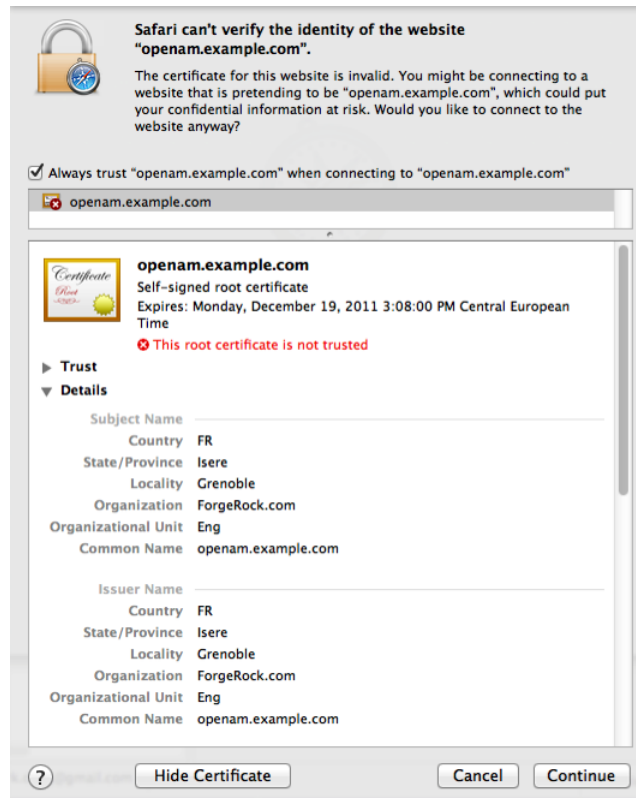
```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"  
maxThreads="200" SSLEnabled="true" scheme="https" secure="true"  
keystoreFile="/path/to/tomcat_keystore.pfx"  
keystoreType="PKCS12" keystorePass="keystore_password"  
clientAuth="false" sslProtocol="TLS" />
```

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 Apache Tomcat on port 8443 over HTTPS.

If you used self-signed certificates, your browser would not trust the certificate, because the certificate is self-signed and not signed by any of the CAs stored in your browser.

Unknown Certificate



If you recognize the subject and issuer of your certificate, and so can choose to trust the certificate, save it into your browser's trust store.

6. Deploy and configure AM.
7. (Optional) To share the self-signed certificate in your container with other applications or servers, see "To Share Self-Signed Certificates".

To Share Self-Signed Certificates

How you configure the containers where AM and your applications run to use self-signed certificates depends on your web application server or web server software. The following basic principles apply:

- First, your container requires its own certificate for setting up secure connections.
- Second, the clients connecting must be able to trust the container's certificate. Generally, this means that clients recognize the container's certificate because they have a copy of the public certificate stored somewhere the client trusts.
- Third, if you use certificate authentication in AM, AM must also be able to find a copy of the client's public certificate to trust the client, most likely by finding a match with the certificate stored in the client profile from the identity repository. How you include client certificates in their identity repository entries depends on your identity repository more than it depends on AM.

Some client applications let you trust certificates blindly. This can be helpful when working in your lab or test environment with self-signed certificates. For example, you might want to use HTTPS with the AM RESTful API without having the client recognize the self-signed server certificate:

```
$ curl \
--request POST \
--header "Content-Type: application/json" \
--header "X-OpenAM-Username: demo" \
--header "X-OpenAM-Password: {amDemoPwd}" \
--header "Accept-API-Version: resource=2.0, protocol=1.0" \
'https://openam.example.com:8443/openam/json/realms/root/realms/alpha/authenticate'
{
  curl: (60) Peer certificate cannot be authenticated with known CA certificates
}
```

The **curl** command performs SSL certificate verification by default, using a "bundle" of CA-signed public keys. If the default bundle file is not 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 would like to turn off curl's verification of the certificate *for test purposes only*, use the **--insecure** option.

```
$ curl \
--request POST \
--insecure \
--header "Content-Type: application/json" \
--header "X-OpenAM-Username: demo" \
--header "X-OpenAM-Password: {amDemoPwd}" \
--header "Accept-API-Version: resource=2.0, protocol=1.0" \
'https://openam.example.com:8443/openam/json/realms/root/realms/alpha/authenticate'
{
  "tokenId": "rMLhJjWvo...MAALMxAAA.*",
  "successUrl": "/openam/console",
  "realm": "/alpha"
}
```

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 truststore used by the client. By default, Java applications can use the `$JAVA_HOME/lib/security/cacerts` store. The default password is `changeit`.¹ The following steps 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
-keystore keystore.jceks
Enter keystore password:
Certificate stored in file <openam.crt>;
```

2. Import the certificate into the trust store:

```
$ keytool \
-importcert \
-alias openam.example.com \
-file openam.crt
-trustcacerts \
-keystore $JAVA_HOME/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
```

Preparing AES Key Wrap Encryption

AM encrypts and decrypts system passwords and the keys used in the configuration, and by components such as agents. The default encryption algorithm is Java Cryptography Extension (JCE) `PBEWithMD5AndDES`.

If you need a more secure encryption algorithm, use the Advanced Encryption Standard (AES) Key Wrap algorithm (RFC3394). AM's implementation of AES Key Wrap uses the Password-Based Key

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`.

Derivation Function 2 (PBKDF2) (RFC2898) with HMAC-SHA1. This lets you choose key size hash algorithms, such as SHA256, SHA384, or SHA512.

Important

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

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

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

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="$JAVA_OPTS -  
Dcom.ipplanet.security.encryptor=org.forgerock.openam.shared.security.crypto.AESWrapEncryption" ❶  
JAVA_OPTS="$JAVA_OPTS -Dorg.forgerock.openam.encryption.key.iterations=10000" ❷  
JAVA_OPTS="$JAVA_OPTS -Dorg.forgerock.openam.encryption.useextractandexpand=true" ❸  
JAVA_OPTS="$JAVA_OPTS -Dorg.forgerock.openam.encryption.key.size=256" ❹  
JAVA_OPTS="$JAVA_OPTS -Dorg.forgerock.openam.encryption.key.digest=SHA512" ❺  
JAVA_OPTS="$JAVA_OPTS -Dorg.forgerock.openam.encryption.padshortinputs" ❻
```

- ❶ Enables use of AES Key Wrap encryption.
- ❷ Specifies the iteration count of the encryption key.

Large iteration counts, for example, of 20,000, slow down brute-force attacks when passwords are of low quality (less than 20 characters and easy to predict).

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

- The number of iterations is less than 10,000.
- The AM encryption key is less than 20 characters long.
- ❸ Enables the algorithm introduced in AM 7.1 that reduces the performance cost of AES Key Wrap encryption even when high iteration counts are used.

If this property is unset, and you configured a large iteration count, AM startup times may see a performance impact if there are many agents in your deployment.

Determine the optimal iteration count based on the security and performance requirements of your deployment.

- ❹ Specifies the size of the encryption key.

- 5 Configure the key size to meet the needs of your deployment.
- 6 Specifies the digest algorithm. Possible values are [SHA1](#), [SHA256](#), [SHA384](#), or [SHA512](#).
- 6 Configure the digest algorithm to meet the needs of your deployment.
- 6 For systems running Java 17, this property pads short inputs (less than 8 bytes). If you are using Java 17 with AES Key Wrap Encryption, enable this system property and re-encrypt any short system passwords that have already been encrypted. If you do not do this, AM will be unable to decrypt the short values.

Configure the digest algorithm to meet the needs of your deployment.

Caution

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

To Configure ssoadm for AES Key Wrap Encryption

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

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

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

Chapter 5

Deploying AM

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

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

Important

After deploying AM, but before installation, your application container serves AM's installer (or upgrader, if performing an upgrade) user interfaces.

We recommend that any external network access to the application container is suspended until the install, or upgrade, is complete. When complete, AM prevents access to the installer, or upgrader UI itself.

To Deploy an Instance

The `AM-7.1.4.war` file contains the AM server. How you deploy the `.war` file depends on your web application container.

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

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

```
$ cp AM-7.1.4.war /path/to/tomcat/webapps/openam.war
```

In development or demonstration deployments, change the WAR file name to `openam.war` when deploying in Tomcat, so that the deployment URI is `/openam`.

Note

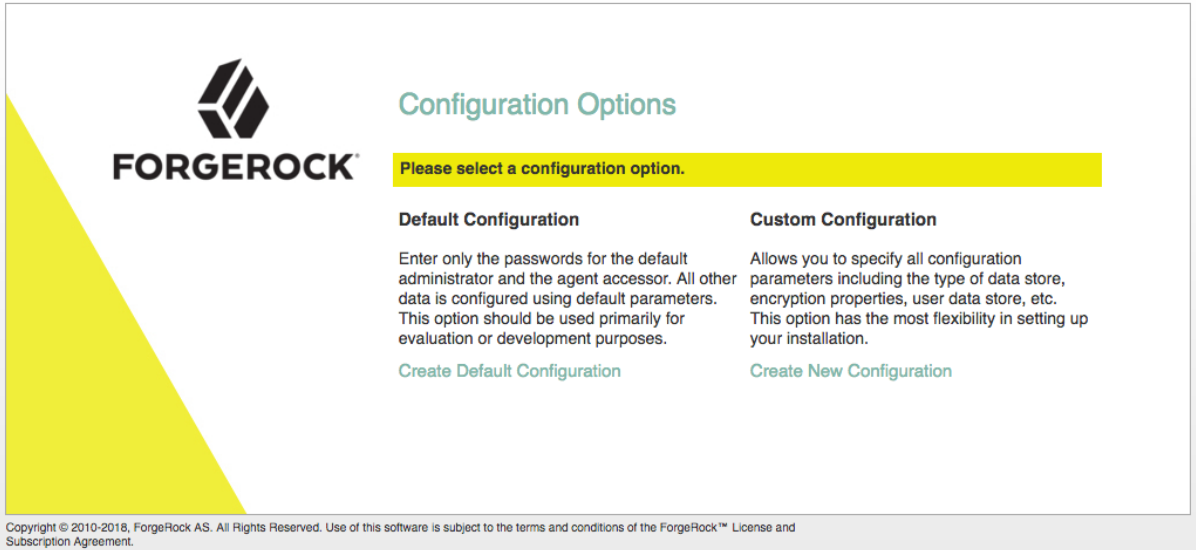
Change the file name to something other than `openam.war` when deploying so that the deployment URI is not `/openam`. In a production environment, your deployment URI should not disclose the kind of software it is hosting.

Important

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

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

- Navigate to the initial configuration screen. For example, <https://openam.example.com:8443/openam>.



AM is now ready for installation.

- Proceed to configuring external data stores using the files created during deployment. See "*Preparing External Stores*".

Chapter 6

Preparing External Stores

AM requires at least one DS server to store the different data it requires to work, such as AM's own configuration data, information about your users, devices, and things, as well as data pertaining to authenticated identities.

By default, AM stores all the data after the installation process in the *configuration store*, but you can configure different DS instances to keep data separated. This can be useful in high-load deployments; for example, when data tuning requirements differ.

Tip

If you want to install a single AM instance for a quick test or for demo purposes, AM provides an embedded DS server. See the Evaluation Guide.

The following table shows the different DS stores that AM supports:

Store Name	Type of Data	Required During Installation?
Configuration store	Stores the properties and settings used by the AM instance.	Yes
Identity or user store	Stores identity profiles; that is, information about the users, devices, or things that will be authenticating to your systems. You can also configure AM to access existing directory servers to obtain identity profiles.	No, but you can configure one during the install process ForgeRock recommends that you configure an external identity store, or that you configure AM to access an external identity store.
Policy store	Stores policy-related data, such as policies, policy sets, and resource types.	No
Application store	Stores application-related data, such as web and Java agent configurations, federation entities and configuration, and OAuth 2.0 client definitions.	No
CTS token store	Stores information about sessions, SAML v2.0 assertions, OAuth 2.0 tokens, and session blacklists and whitelists.	No

Store Name	Type of Data	Required During Installation?
UMA store	Stores information about UMA resources, labels, audit messages, and pending requests.	No

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

Supported Data Stores

Directory Server	Versions	Configuration	Apps / Policies	CTS	Identities	UMA
Embedded ForgeRock Directory Services ^a	7.1.4	✓	✓	✓	✓	✓
External ForgeRock Directory Services	Any ForgeRock-supported version	✓	✓	✓	✓	✓
File system-based	N/A	✓				
Oracle Unified Directory	11g R2				✓	
Oracle Directory Server Enterprise Edition	11g				✓	
Microsoft Active Directory	2016, 2019				✓	
IBM Tivoli Directory Server	6.4				✓	

^aDemo and test environments only

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

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

To prepare the external stores AM needs during installation, see the following sections:

		
---	---	---

Prepare a Truststore	Prepare Configuration Stores	Prepare Identity Repositories
Create a truststore ready for the certificates required to connect to external stores using LDAPS.	Install and prepare a DS instance to serve as AM's configuration store.	Install and prepare a DS instance to serve as an identity repository for AM. Optionally, prepare an existing identity store for use by AM.

+ *Where do I find more information about the other external stores?*

You can configure all the stores except the configuration store after installing AM. To read more, see:

- "Preparing Policy and Application Stores" in the *Setup Guide*.
- Core Token Service Guide (CTS).
- "Preparing External UMA Data Stores" in the *User-Managed Access (UMA) 2.0 Guide*.

Preparing a Truststore

Directory Services 7 introduces a *secure by default* approach. One aspect of this approach is that all connections to DS 7 instances must be made using secure connection, including LDAPS.

To connect to a DS instance using LDAPS, AM requires access to the self-signed certificate that DS generates.

AM also requires access to CA certificates for making secure connections to other sites, for example calling out to social providers using HTTPS.

To provide these certificates to AM, you use a *truststore* that contains the necessary certificates, and configure AM to use that truststore when starting up.

By default, Apache Tomcat loads the truststore configured for its JVM (for example, `$JAVA_HOME/JRE/lib/security/cacerts`). This file contains multiple CA certificates. Its password, by default, is `changeit`.

As a best practice, create a new truststore with the certificates you need in your environment and configure your container to use it. Do not add the DS certificate to the JVM's truststore because other applications may be using it.

AM 7 only supports a single truststore. As the truststore is also used for outbound HTTPS connections, your truststore also needs to contain the CA certificates of sites that your instance communicates securely with.

For example, to keep all existing CA certificates, copy the `cacerts` keystore file, change its password to a secure one, and import the DS certificate into it.

Then, ensure your web application container; for example, Apache Tomcat, is loading that file as its truststore.

+ What if I am evaluating using an embedded DS?

If you are installing AM for evaluation purposes, AM creates a copy of your JDK's default `lib/security/cacerts` truststore, names it `truststore`, and places it in `/path/to/openam/security/keystores/`.

AM then attempts to add the DS self-signed certificate to that store, with an alias of `ds-ca-cert`.

Important

If the `lib/security/cacerts` truststore does not have the default password of `changeit`, and/or if it does not have at least 644 permissions, then AM installation will fail, as it will not be able to open the truststore to add the DS certificate.

To Create a Truststore for AM

Follow the steps in this procedure to create a copy of the default truststore for AM to use, and configure your container to use the store.

1. Make a copy of your JDK's default truststore; for example, `$JAVA_HOME/lib/security/cacerts`, name it `truststore`, and place it in a directory of your choice.

```
$ cp $JAVA_HOME/lib/security/cacerts /my/directory/truststore
```

Caution

If you place the truststore in the `/path/to/openam` directory before installing AM, the installation process will detect that the directory is not empty and the installation will not continue.

Once AM is installed, you can move the truststore to a different directory. For example, the `/path/to/openam/security/keystores` directory.

2. (Optional) For security reasons, you *should* change the default password for the truststore.

Tip

The default password of the `$JAVA_HOME/lib/security/cacerts` truststore is `changeit`.

Use the `keytool -storepasswd` option to change the default password:

```
$ keytool -storepasswd \  
-keystore /my/directory/truststore  
  
Enter keystore password: changeit  
New keystore password: new-password  
Re-enter new keystore password: new-password
```

3. Export the DS server certificate:


```
$ keytool -exportcert \  
-keystore /path/to/opensj/config/keystore \  
-storepass $(cat /path/to/opensj/config/keystore.pin) \  
-alias ssl-key-pair \  
-rfc \  
-file ds-cert.pem
```

The default DS server certificate only has the hostname you supplied at setup time, and `localhost`, as the value of the `SubjectAlternativeName` attribute; however, certificate hostname validation is strict.

Copy the `ds-cert.pem` file to an accessible location on the AM host.

4. Import the DS server certificate into the new truststore:

```
$ keytool \  
-importcert \  
-file ds-cert.pem \  
-keystore /my/directory/truststore
```

5. To configure the truststore in Apache Tomcat so that AM can access it, append the truststore settings to the `CATALINA_OPTS` variable in the `$CATALINA_BASE/bin/setenv.sh` file.

For example:

```
export CATALINA_OPTS="$CATALINA_OPTS -Djavax.net.ssl.trustStore=/my/directory/truststore\  
-Djavax.net.ssl.trustStorePassword=new-password\  
-Djavax.net.ssl.trustStoreType=jks"
```

Refer to your specific container's documentation for information on configuring truststores.

Once AM is installed, you can move the truststore to a different location. For example, the `/path/to/openam/security/keystores/`. If you do, remember to reconfigure its path in the container.

Preparing Configuration Stores

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

To Install and Configure Directory Services for Configuration Data

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

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

1. To install DS using a setup profile, follow the steps in *DS for AM Configuration Data* in the *Directory Services 7.1 Installation Guide*.
2. Proceed to installation of AM to use the prepared DS directory server as an external configuration store. See "To Configure an Instance".

The default bind DN of the service account to use when installing AM to authenticate to the external configuration store is:

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

3. Share the configuration store certificate with the AM container to prepare for TLS/LDAPS. The configuration store should communicate over secure connections for security reasons.

DS 7 or later is configured to require secure connections by default; therefore, share its certificate with the AM container before continuing.

+ *Sharing the DS Certificate with AM*

1. Export the DS server certificate:

```
$ keytool -exportcert \  
-keystore /path/to/openssl/config/keystore \  
-storepass $(cat /path/to/openssl/config/keystore.pin) \  
-alias ssl-key-pair \  
-rfc \  
-file ds-cert.pem*
```

The default DS server certificate only has the hostname you supplied at setup time, and `localhost`, as the value of the `SubjectAlternativeName` attribute; however, certificate hostname validation is strict. Ensure that the certificate matches the hostname (or the FQDN) of the DS server before continuing.

Copy the `ds-cert.pem` file to an accessible location on the AM host.

2. Import the DS certificate into the AM truststore:

```
$ keytool \  
-importcert \  
-file ds-cert.pem \  
-keystore /path/to/openam/security/keystores/truststore
```

For more information on configuring AM's truststore, see "Preparing a Truststore".

Once the certificate is in place, continue installing AM.

Preparing Identity Repositories

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

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

Note

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

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

To prepare external identity repositories, see the following sections:

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

Installing and Configuring Directory Services for Identity Data

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

To Install and Configure Directory Services for Identity Data

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

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

1. To install DS using a setup profile, follow the steps in *DS for AM Identities* in the *Directory Services 7.1 Installation Guide*.
2. Share the identity store certificate with the AM container to prepare for TLS/LDAPS. Identity stores should communicate over secure connections for security reasons.

DS 7 or later is configured to require secure connections by default; therefore, share its certificate with the AM container before continuing.

+ *Sharing the DS Certificate with AM*

1. Export the DS server certificate:

```
$ keytool -exportcert \  
-keystore /path/to/opensj/config/keystore \  
-storepass $(cat /path/to/opensj/config/keystore.pin) \  
-alias ssl-key-pair \  
-rfc \  
-file ds-cert.pem*
```

The default DS server certificate only has the hostname you supplied at setup time, and `localhost`, as the value of the `SubjectAlternativeName` attribute; however, certificate hostname

validation is strict. Ensure that the certificate matches the hostname (or the FQDN) of the DS server before continuing.

Copy the `ds-cert.pem` file to an accessible location on the AM host.

2. Import the DS certificate into the AM truststore:

```
$ keytool \  
-importcert \  
-file ds-cert.pem \  
-keystore /path/to/openam/security/keystores/truststore
```

For more information on configuring AM's truststore, see "Preparing a Truststore".

3. (Optional) If you did not install DS using a setup profile, perform the following steps to update the permissions in an external Directory Services identity repository.

If you are using a directory server other than Directory Services, apply the relevant LDIF files using the appropriate tools.

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

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

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

```
$ /path/to/opendj/bin/ldapmodify \  
--hostname 'id.example.com' \  
--port 1636 \  
--useSsl \  
--usePkcs12TrustStore /path/to/opendj/config/keystore \  
--trustStorePasswordFile /path/to/opendj/config/keystore.pin \  
--continueOnError \  
--bindDN uid=admin \  
--bindPassword str0ngAdm1nPa55word \  
/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opendj/opendj_userinit.ldif
```

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

- c. (Optional) If you intend to use web authentication, or perform device profiling with the ForgeRock SDK, you might need to update the directory server schema. For a ForgeRock Directory Services repository, you can update the schema by applying the following LDIF files:

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

- `/path/to/openam/WEB-INF/template/ldif/opensj/opensj_deviceprofiles.ldif`

For example:

```
$ /path/to/opensj/bin/ldapmodify \  
--hostname 'id.example.com' \  
--port 1636 \  
--useSsl \  
--usePkcs12TrustStore /path/to/opensj/config/keystore \  
--trustStorePasswordFile /path/to/opensj/config/keystore.pin \  
--continueOnError \  
--bindDN uid=admin \  
--bindPassword str0ngAdm1nPa55word \  
/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opensj/opensj_webauthndevices.ldif\  
/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opensj/opensj_deviceprofiles.ldif
```

For more information on these LDIF files, see "Supported LDIF Files". For directory servers other than ForgeRock Directory Services, adapt the `opensj_*.ldif` files accordingly.

4. Proceed to configuring the identity store in AM. See "*Identity Stores*" in the *Setup Guide*.

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

Configuring Existing Directory Servers for Identity Data

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

For a list of supported external directory servers, see "Supported Data Stores".

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

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

Creating a Directory Server User for AM Connections

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

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

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

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

Schema Update Privileges

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

Directory Read and Write Access Privileges

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

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

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

To Create a Bind Account in Active Directory for AM Connections

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

1. Create a new user account in the Active Directory domain. For example, in Windows 2019:
 - a. In the Active Directory Users and Computers tool, right-click Users in the domain, and select New > User.
 - b. Provide descriptive values for the new user, and a logon name such as **AM-Bind-Account**.
 - c. Click Next.
 - d. Enter a strong password for the bind account, disable the *User must change password at next logon* option, and then click Next.
 - e. Review the details of the new account, and then click Finish.
2. Give the new bind account access to the directory data:
 - a. In the Active Directory Users and Computers tool, right-click the domain that contains your users, and then select Delegate Control.
 - b. Add the bind account you created in the previous step, and then click Next.
 - c. Select the tasks you want to allow the AM bind account to perform from the list.

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

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

You can now set up the necessary schema in the directory server.

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

Updating the Schema in an External Identity Repository

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

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

To Prepare an External Identity Repository With Manual Schema Updates

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The command has completed successfully
```

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

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

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

4. Proceed to configuring the identity store in AM. See "Identity Stores" in the *Setup Guide*.

Important

If you updated the schema to make use of web authentication, when configuring the external identity store in a realm, on the User Configuration tab, ensure `webauthnDeviceProfilesContainer` is in the LDAP User Object Class property. If not, add the value, and then save your changes.

To Prepare an External Identity Repository With Automatic Schema Updates

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

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

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

Important

Enable the Load Schema option before clicking Save Changes, and AM will apply the necessary schema to the directory server.

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

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

Supported LDIF Files

AM installation deploys several LDIF files that can be used to create the schemas required by AM. LDIF files are available for Microsoft Active Directory, Microsoft Active Directory Lightweight Directory Services, Oracle Directory Server Enterprise Edition, ForgeRock Directory Services, Oracle Unified Directory, and IBM Tivoli Directory Server.

The following tables provide descriptions for each LDIF file:

Microsoft Active Directory LDIF Files

LDIF File	Description
ad_config_schema.ldif	LDIF for the configuration schema.
ad_dashboard.ldif	LDIF to support the dashboard service.
ad_deviceprint.ldif	LDIF to support the device print service.

LDIF File	Description
ad_kba.ldif	LDIF to support the User Self-Service's knowledge-based questions and answers service.
ad_oathdevices.ldif	LDIF to support registered devices for the OATH authentication service.
ad_pushdevices.ldif	LDIF to support registered devices for the PUSH notification service.
ad_user_schema.ldif	LDIF for the user schema.
ad_webauthndevices.ldif	LDIF to support registered devices for the Web Authentication (WebAuthn) authentication service.

Microsoft Active Directory Lightweight Directory Services LDIF Files

LDIF File	Description
adam_dashboard.ldif	LDIF to support the dashboard service.
adam_deviceprint.ldif	LDIF to support the device print service.
adam_kba.ldif	LDIF to support the User Self-Service's knowledge-based questions and answers.
adam_oathdevices.ldif	LDIF to support registered devices for the OATH authentication service.
adam_pushdevices.ldif	LDIF to support registered devices for the PUSH notification service.
adam_user_schema.ldif	LDIF for the user schema.
adam_webauthndevices.ldif	LDIF to support registered devices for the Web Authentication (WebAuthn) authentication service.

Oracle Directory Server Enterprise Edition LDIF Files

LDIF File	Description
amsdk_plugin	Folder containing the AM SDK LDIF files: amsdk_init_template.ldif and amsdk_sunone_schema2.ldif.
odsee_config_index.ldif	LDIF for the ODSEE configuration indexes.
odsee_config_schema.ldif	LDIF for the ODSEE configuration schema.
odsee_dashboard.ldif	LDIF to support the dashboard service.
odsee_deviceprint.ldif	LDIF to support the device print service.
odsee_kba.ldif	LDIF to support the User Self-Service's knowledge-based questions and answers.
odsee_oathdevices.ldif	LDIF to support registered devices for the OATH authentication service.

LDIF File	Description
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.
odsee_webauthndevices.ldif	LDIF to support registered devices for the Web Authentication (WebAuthn) authentication service.

DS LDIF Files

LDIF File	Description
oath_2fa.ldif	LDIF for the OATH two-factor authentication service.
opendj_aci_lift_user_password_restriction.ldif	LDIF to add an ACI entry to the root suffix to allow users to modify the user password attribute.
opendj_aci_remove_blanket_deny_all.ldif	LDIF to lift any user password restrictions for upgrade.
opendj_add_kba_attempts.ldif	LDIF to upgrade a user data store from a version earlier than AM 6 to support account lockout when the user fails to answer their security questions a number of times.
opendj_config_schema.ldif	LDIF for the DS configuration schema.
opendj_dashboard.ldif	LDIF to support the dashboard service.
opendj_deviceprint.ldif	LDIF to support the device print service.
opendj_deviceprofiles.ldif	LDIF to support storage of device information, collected by the SDK device authentication nodes. Apply this LDIF if you intend to use the ForgeRock SDK for device profiling.
opendj_embinit.ldif	LDIF for the DS user management and SMS/configuration datastore schema for evaluation (embedded DS) deployments.
opendj_kba.ldif	LDIF to support the User Self-Service's knowledge-based questions and answers.
opendj_oathdevices.ldif	LDIF to support registered devices for the OATH authentication service.
opendj_pushdevices.ldif	LDIF to support registered devices for the PUSH notification service.
opendj_remove_config_schema.ldif	LDIF to remove the configuration schema.
opendj_remove_user_schema.ldif	LDIF to remove the user schema.
opendj_retry_limit_node_count.ldif	LDIF to upgrade the identity store to support persisting failed login attempts to the user's profile when using the "Retry Limit Decision Node" in the <i>Authentication and Single Sign-On Guide</i> .

LDIF File	Description
	There are no equivalent files for other supported directory servers. Adapt the contents of the <code>opendj_retry_limit_node_count.ldif</code> file to work with your directory server.
<code>opendj_uma_audit.ldif</code>	LDIF to add auditing capabilities for the UMA service.
<code>opendj_uma_labels_schema.ldif</code>	LDIF to add a schema for the UMA service labels.
<code>opendj_uma_pending_requests.ldif</code>	LDIF to add pending requests for the UMA service.
<code>opendj_uma_resource_set_labels.ldif</code>	LDIF to support labels for UMA resources.
<code>opendj_uma_resource_sets.ldif</code>	LDIF to support UMA resources.
<code>opendj_update_aci_kba_attempts.ldif</code>	LDIF to upgrade a user data store from a version earlier than AM 6 to support account lockout when the user fails to answer their security questions a number of times.
<code>opendj_user_index.ldif</code>	LDIF for the user repository indexes.
<code>opendj_user_schema.ldif</code>	LDIF for the user repository schema.
<code>opendj_userinit.ldif</code>	LDIF for the setting up user session initialization.
<code>opendj_webauthndevices.ldif</code>	LDIF to support registered devices for the Web Authentication (WebAuthn) authentication service.
<code>push_2fa.ldif</code>	LDIF for the push two-factor authentication service. Not required if you installed DS 7.1 or later by using the `am-identity-store` setup profile.

Tivoli LDIF Files

LDIF File	Description
<code>tivoli_dashboard.ldif</code>	LDIF to support the dashboard service.
<code>tivoli_deviceprint.ldif</code>	LDIF to support the device print service.
<code>tivoli_kba.ldif</code>	LDIF to support the User Self-Service's knowledge-based questions and answers.
<code>tivoli_oathdevices.ldif</code>	LDIF to support registered devices for the OATH authentication service.
<code>tivoli_pushdevices.ldif</code>	LDIF to support registered devices for the PUSH notification service.
<code>tivoli_user_schema.ldif</code>	LDIF for the user repository schema.
<code>tivoli_webauthndevices.ldif</code>	LDIF to support registered devices for the Web Authentication (WebAuthn) authentication service.

Chapter 7

Installing Instances

Production or production-like environments require an external configuration store and an external CTS token store, both of which are DS servers. The configuration store will act as the CTS token store until you configure an external one.

For more information, see "*Preparing External Stores*".

+ *Installing Multiple Instances for High Availability*

Install multiple instances to maintain service availability. If one instance is down for any reason, another instance can respond instead. This means that you need some type of component, such as a load balancer or a proxy server, between incoming traffic and AM to route around instances that are down.

AM uses a *site* for this purpose. In an AM site, multiple AM instances are configured in the same way, and accessed through a load balancer layer.

The load balancer can be implemented in hardware or software, but it is separate and independent from AM. When installed properly, a site configuration improves service availability, as the load balancer routes around AM instances that are down, sending traffic to other servers in the site.

For high-level deployment information, see the *Deployment Planning Guide*.

The following table provides information to help you install AM instances:

Task or Requirement	Resources
Configure a single instance on a production or pre-production environment. This can be the first instance of a site deployment.	"To Configure an Instance"
Configure a site and add a instance to an AM site	"Configuring Sites and Adding Servers to Sites"
Configure an instance silently	"Installing Silently"

Tip

If at any point you need to scrap the AM configuration to start again, see "Deleting and Redeploying AM".

Installing an Instance

You can customize several AM parameters during installation, such as the cookie domain and the settings of the configuration store.

Perform the steps in the following procedure to install a single AM instance, or to install the first instance on a site.

- Installing the first instance creates the required configuration that the site will share.

You can specify the site configuration when you install the first instance or configure the site when the first instance is running.

- By default, the cookie domain is set to the full URL of the first instance; for example, `server.west.example.com`.

You can change the cookie domain when you're installing the first instance or later.

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

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

- When you are protecting AM with a load balancer or proxy service, configure your container so that AM can trust the load balancer or proxy service.
- The container for each instance in the site must trust any certificate authorities (CA) used to sign certificates used by other instances in the site in order to communicate using SSL.
- AM authentication can depend on information about the user to authenticate, such as the IP address where the request originated. When AM is accessed through a load balancer or proxy layer, pass this information along using request headers. Also, configure AM to consume and to forward the headers as necessary. Refer to "Handling HTTP Request Headers" in the *Setup Guide* for details.

To Configure an Instance

1. Navigate to your deployment URL. For example, `https://openam.example.com:8443/openam`.
2. In the initial configuration screen, click Create New Configuration under Custom Configuration.
3. Read the license agreement. If you agree to the license, click "I agree to the license agreement", and then click Continue.

- On the Default User Password page, provide a password with at least eight characters for the AM Administrator, `amAdmin`.

ForgeRock Access Management Configurator
✕

Custom Configuration Option

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

Step 1: General 🔔

Enter the password for the default user, `amAdmin`. The password must be at least 8 characters long. 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

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

The screenshot shows the 'ForgeRock Access Management Configurator' window with the 'Custom Configuration Option' tab selected. The left sidebar lists steps: 1. General, 2. Server Settings (selected), 3. Configuration Store, 4. User Store, 5. Site Configuration, and 6. 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' (https://openam.example.com:8443), 'Cookie Domain' (openam.example.com), 'Platform Locale' (en_US), and 'Configuration Directory' (/home/forgerock/openam). Each field has an 'OK' checkbox. A red asterisk indicates required fields. At the bottom are 'Previous', 'Next', and 'Cancel' buttons.

Server URL

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

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

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

Cookie Domain

The domain for which created cookies will be valid; for example `example.com`.

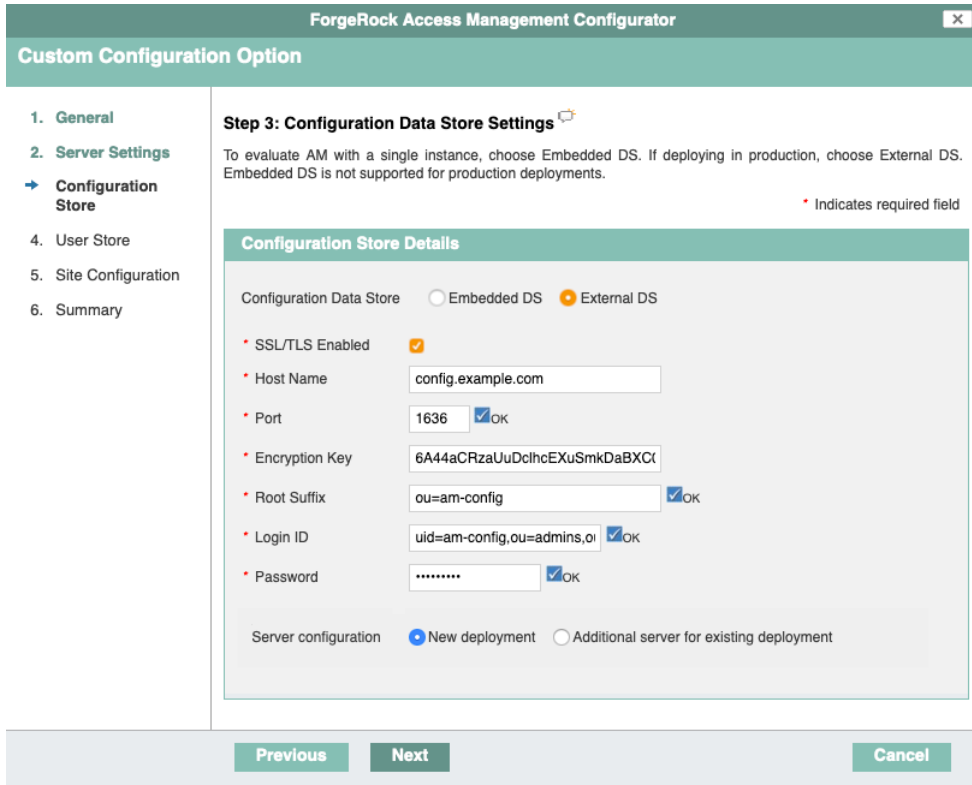
Platform Locale

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

Configuration Directory

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

- In the Configuration Data Store screen, you make choices related to AM configuration data.



The screenshot shows the 'ForgeRock Access Management Configurator' window with the 'Custom Configuration Option' tab selected. The left sidebar lists steps: 1. General, 2. Server Settings, 3. Configuration Store (highlighted), 4. User Store, 5. Site Configuration, and 6. Summary. The main content area is titled 'Step 3: Configuration Data Store Settings' and includes a note: 'To evaluate AM with a single instance, choose Embedded DS. If deploying in production, choose External DS. Embedded DS is not supported for production deployments.' Below this is a 'Configuration Store Details' section with the following settings:

- Configuration Data Store: Embedded DS, External DS
- * SSL/TLS Enabled:
- * Host Name: config.example.com
- * Port: 1636 OK
- * Encryption Key: 6A44aCRzaUuDclhcEXuSmkDaBXCf
- * Root Suffix: ou=am-config OK
- * Login ID: uid=am-config,ou=admins,ou= OK
- * Password: OK
- Server configuration: New deployment, Additional server for existing deployment

At the bottom of the window are 'Previous', 'Next', and 'Cancel' buttons.

Configuration Store

- Embedded DS

The configurator process spins an embedded DS instance to store AM configuration data, the default users, and the CTS store. Use on **demo or test environments only**.

Instances using the embedded DS server cannot be added to a site.

If you choose the embedded DS, you can leave the rest of the values by default.

- External DS

The configurator process stores AM configuration data in an existing DS server. You must have prepared the server as explained in "Preparing Configuration Stores".

Choose this option if you intend to add this instance to an existing deployment.

An external DS instance is **required** in non-evaluation deployments.

If you choose the external DS, you must configure the rest of the fields.

SSL/TLS Enabled

Whether AM must use LDAPS to communicate with the configuration store.

For security reasons, the configuration store should communicate with AM using LDAPS.

DS 7 is configured for LDAPS by default. If you are using this version, enable this option and share the DS certificate with the container where AM is running before continuing. For more information, refer to "Securing Directory Server Communication" in the *Security Guide*.

If you are using a different version, you can configure DS and AM to use LDAPS after the installation.

Host Name

The FQDN of the external DS.

Port

The LDAP or LDAPS port of the external DS. The default values are:

- LDAP: 1389
- LDAPS: 1636

Encryption Key

A randomly-generated key that AM uses for different purposes. All the servers in the site must have the same encryption key.

The installer creates a random key automatically; you can leave the value by default.

Root Suffix

The root suffix of the external DS store.

The default base DN of an external DS store when you configure it with the `am_config` profile is `ou=am-config`.

Login ID

The bind DN that AM should use to connect to the external DS store.

The default bind DN of an external DS store when you configure it with the `am_config` profile is `uid=am-config,ou=admins,ou=am-config`.

Do not use `cn=admin` as the bind account.

Password

The password of the bind DN.

Server configuration

Note that this option only appears when you specify an external configuration store.

- New deployment

Specifies that the installation is a new deployment, with its own configuration and identity stores.

If you choose this option, the next step is to configure the identity store.

- Additional server for existing deployment

Specifies that the installation is an additional server for an existing deployment and will use the existing configuration and identity stores.

If you choose this option, you don't need to configure the identity store. The installation uses the same stores as those of the existing deployment.

For more information, refer to "To Add a Server to a Site".

7. (Optional) If you specified New deployment in the previous step, the User Store screen appears as the next step. Use the page to configure where AM looks for user identities.

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

ForgeRock Access Management Configurator
✕

Custom Configuration Option

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

Step 4: User Data Store Settings

You can store user data in the embedded configuration data store for evaluation purposes. For production deployments you will need to use an external user data store. Please note that the Policy Service and LDAP Authentication Module are configured to use the Directory Administrator DN and Password provided here.

Embedded User Data Store (DS)
 External User Data Store

* Indicates required field

User Store Details

* User Data Store Type

ForgeRock Directory Services (DS) Oracle Directory Server Enterprise Edition
 AD with Domain Name Active Directory with Host and Port
 IBM Tivoli Directory Server Active Directory Application Mode
 ForgeRock DS For IAM

* SSL/TLS Enabled

* Directory Name

* Port

* Root Suffix OK

* Login ID OK

* Password OK

Previous
Next
Cancel

User Data Store Type

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

SSL/TLS Enabled

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

Directory Name

FQDN for the host housing the directory service.

Port

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

Root Suffix

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

Login ID

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

Password

Password for the directory administrator user.

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

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

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

ForgeRock Access Management Configurator
✕

Custom Configuration Option

1. General
2. Server Settings
3. Configuration Store
4. User Store
- ➔ Site Configuration
6. 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 AM, and no site configurations currently exist. To create a new site configuration, provide the following information

* Site Name OK

* Load Balancer URL OK

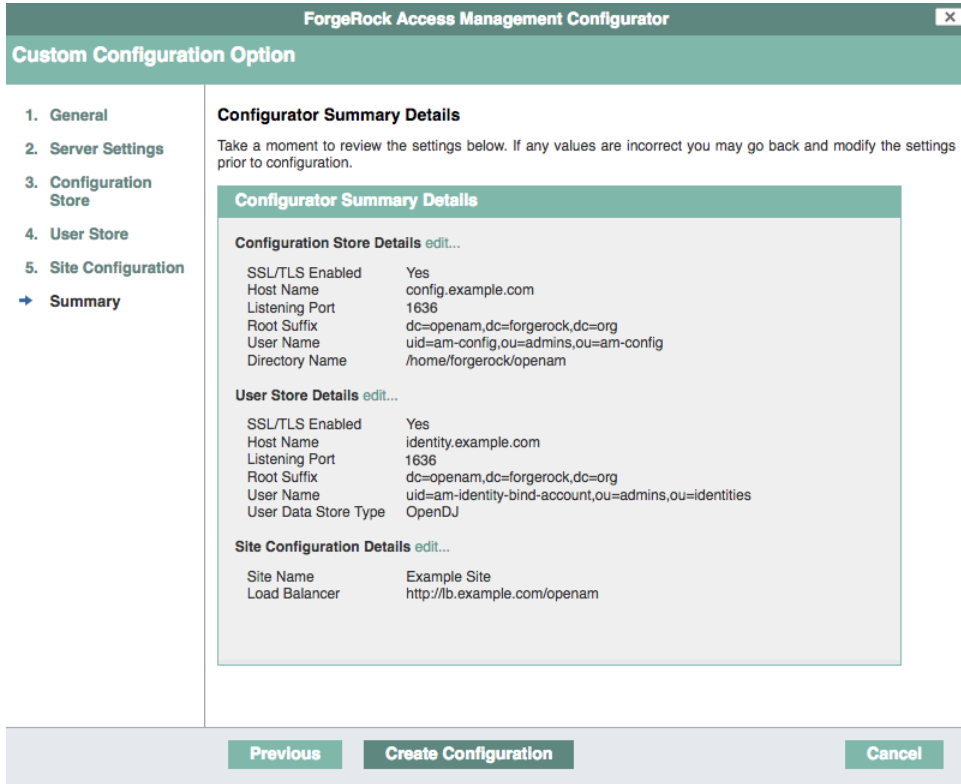
Previous

Next

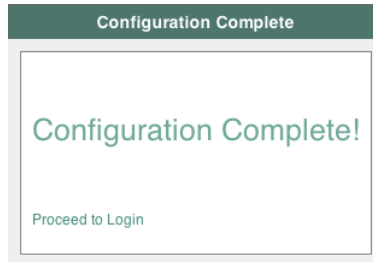
Cancel

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

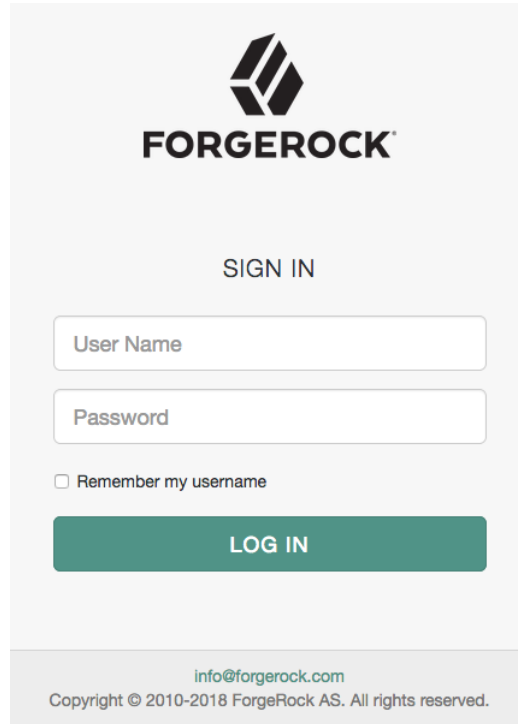
9. Check the summary screen, and if necessary, click Previous to return to earlier screens to fix any configuration errors as needed.



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 AM shows the Proceed to Login prompt.

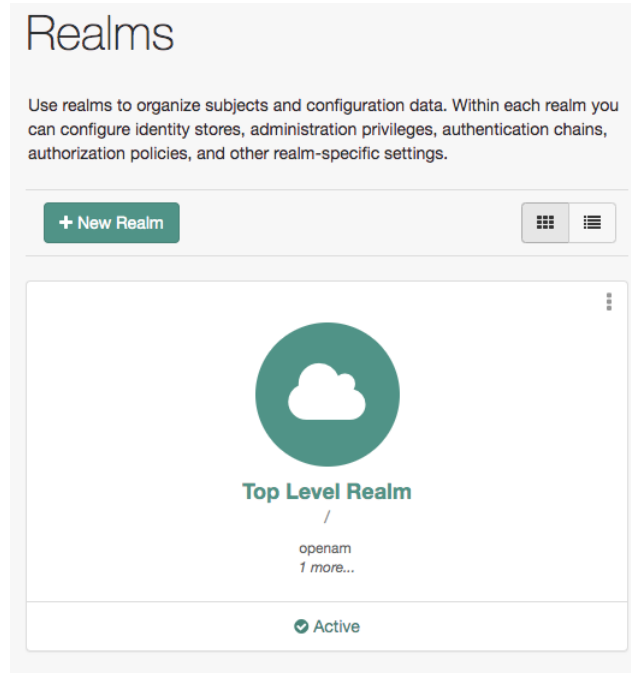


10. When the configuration completes, click Proceed to Login, and then log in as the AM administrator, `amAdmin`.



The image shows a sign-in form for ForgeRock. At the top is the ForgeRock logo, which consists of a stylized 'F' made of three vertical bars of varying heights, followed by the word 'FORGEROCK' in a bold, sans-serif font. Below the logo is the text 'SIGN IN'. There are two input fields: 'User Name' and 'Password'. Below the 'Password' field is a checkbox labeled 'Remember my username'. At the bottom of the form is a green button with the text 'LOG IN'. Below the form is a footer containing the email address 'info@forgerock.com' and the copyright notice 'Copyright © 2010-2018 ForgeRock AS. All rights reserved.'

After logging in, AM redirects you to the AM Realms page.



You can also access the AM console by going to the console URL; for example <https://openam.example.com:8443/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. The AM install wizard uses three libraries that should be removed after installation for security reasons.

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

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

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

13. Review the suggested next steps after installing AM. See "*Next Steps*".

Configuring Sites and Adding Servers to Sites

Configuring a site is a three-step process:

1. Install the first server in the site. This will create the configuration that the site will share.

See "Installing an Instance".

2. Add the first server to a site, if you did not already while installing it.

See "To Configure a Site With the First Server".

3. Add more servers to the site.

See "To Add a Server to a Site".

To Configure a Site With the First Server

The following steps show how to set up a site when AM is running:

1. Review AM's load balancing requirements in "*Load Balancers*" in the *Setup Guide*.
2. In the AM console, go to Deployment > Sites.
3. Click Add a Site to start configuring the new site.
4. On the New Site page enter the site name without any spaces. For example, the site name must be in the format `ExampleSite`, rather than `Example Site`.

Set the Primary URL to the load balancer URL that is the entry point for the site, such as `https://lb.example.com/openam`.

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

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

5. Click Save to keep the site configuration.
6. Configure the cookie domain of your site as required. For more information, see "Configuring the Cookie Domain" in the *Security Guide*.
7. Navigate to Deployment > Servers > *Server Name* > General.
8. Set the Parent Site drop-down to the name of the site you just created, and save your changes.

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

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

To Add a Server to a Site

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

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

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

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

Note

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

5. In the configuration store screen, ensure that you select the External DS option, and configure the same DS instance that is already working as the configuration store for the rest of the instances in the site, including the same encryption key:

ForgeRock Access Management Configurator
✕

Custom Configuration Option

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

Step 3: Configuration Data Store Settings 🗨️

To evaluate AM with a single instance, choose Embedded DS. If deploying in production, choose External DS. Embedded DS is not supported for production deployments.

* Indicates required field

Configuration Store Details

Configuration Data Store Embedded DS External DS

* SSL/TLS Enabled

* Host Name

* Port OK

* Encryption Key

* Root Suffix OK

* Login ID OK

* Password OK

Server configuration New deployment Additional server for existing deployment

Please ensure that the encryption key value above is the same as that for the other servers already in this deployment.

Previous
Next
Cancel

Ensure that you also select the Additional server for existing deployment option.

Instances using the embedded DS cannot be part of a site.

6. In the site configuration screen, select Yes, and enter the same Site Name and Load Balancer URL values as the existing servers in the site.

Note

Spaces are not allowed in the site name.

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

7. In the summary screen, verify the settings you chose, and then click Create Configuration.
8. When the configuration process finishes, **stop** the newly-installed AM instance or the container where it runs, and do not try to access it.

9. Compare the `/path/to/openam/config/boot.json` bootstrap file with that of a running instance. You must ensure that the newly installed instance's bootstrap file is appropriate for your environment.

+ *The boot.json File Does Not Exist in the New Instance*

Depending on the configuration of the AM keystore in the site, the installation process may not create the bootstrap file.

If this is your case, copy the bootstrap file from another instance and continue with the procedure.

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

If you are overriding the start up settings:

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

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

- a. Back up the new instance's default keystore and password files in the following locations:
 - `/path/to/openam/security/kestores/`
 - `/path/to/openam/security/secrets/default/`
- b. Ensure that the existing keystores in the site are available in the same location to the new instance. This may mean copying the keystores and their password files, mounting a volume, or others.
- c. Ensure that the keystore files configured in the `/path/to/openam/config/boot.json` file are available to the instance.

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

- a. In the AM console of an existing instance in the site, go to Configure > Secret Stores.
- b. Review the list of secret stores configured globally and make sure to provide the relevant stores to the new instance. For example:
 - For keystore-type secret stores, copy the keystores to the same path on the new instance.
 - For filesystem-type secret stores, copy the contents of the directories to the same path or make the filesystem available on the same mount point on the new instance.
 - For HSM-type stores, ensure the new instance can access it.

- For secrets configured as environment variables accessible by the container where AM runs, ensure they are also accessible by the container of the new instance.
- c. Navigate to Realms > *Realm Name* > Secret Stores.
 - d. Review the list of secret stores configured per realm and make sure to provide the relevant stores to the new instance.
12. Restart the new instance.

The instance is now configured for the site.
 13. Review AM's load balancing requirements in "*Load Balancers*" in the *Setup Guide*.
 14. Ensure that the cookie domain configuration is appropriate for your site. For more information, see "*Configuring the Cookie Domain*" in the *Security Guide*.

Installing Silently

AM provides configuration and upgrade tools for installing and maintaining your server. The `AM-SSOConfiguratorTools-5.1.3.19.zip` file contains libraries, legal notices, and supported binaries for these configuration tools. Also, you can view example configuration and upgrade properties files that can be used as a template for your deployments.

Use the configurator tool, `openam-configurator-tool-14.1.3.19.jar`, to silently install AM. The AM server must be deployed and running, but not yet configured, when you use the tool.

Perform the following tasks to install AM silently:

1. Install the configuration tool. See "*To Set Up the Configuration Tools*".
2. Use the configuration tool to install AM using a property file. See "*To Install AM Silently*".

To Set Up the Configuration Tools

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

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

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

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

3. Unpack the tools from where you unzipped AM:

```
$ cd /path/to/openam-tools/config
$ unzip ~/Downloads/openam/AM-SSOConfiguratorTools-5.1.3.19.zip
Archive:  ~/Downloads/openam/AM-SSOConfiguratorTools-5.1.3.19.zip
creating:  legal-notices/
inflating: legal-notices/LICENSE.DOM-software.html
inflating: legal-notices/NOTICE.resolver.txt
inflating: legal-notices/LICENSE.DOM-documentation.html
... (more output) ...
extracting: lib/xml-apis-2.11.0.jar
extracting: openam-configurator-tool-14.1.3.19.jar
extracting: lib/servlet-api-2.5.jar
```

To Install AM Silently

Perform the following steps:

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

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

2. The configurator tool relies on a property file to specify the configuration for the AM server. For property file options, see "configurator.jar" in the *Reference*.

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

```
$ cd /path/to/openam-tools/config
$ cp sampleconfiguration config.properties
$ grep -v "^#" config.properties | grep -v "^$"
SERVER_URL=https://openam.example.com:8443
DEPLOYMENT_URI=/openam
BASE_DIR=/home/openam/
locale=en_US
PLATFORM_LOCALE=en_US
AM_ENC_KEY=
ADMIN_PWD=password
COOKIE_DOMAIN=example.com
ACCEPT_LICENSES=true
DATA_STORE=external
DIRECTORY_SSL=SIMPLE
DIRECTORY_SERVER=config.example.com
DIRECTORY_PORT=50389
DIRECTORY_ADMIN_PORT=4444
DIRECTORY_JMX_PORT=1689
ROOT_SUFFIX=dc=openam,dc=forgerock,dc=org
DS_DIRMGRDN=uid=admin
DS_DIRMGRPASSWD=password
```

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

- If you include the `ACCEPT_LICENSES=true` property, AM auto-accepts the software licensing agreement and suppresses the display of the license acceptance screen during silent installation.

- When installing AM to support HTTPS, make sure the `SERVER_URL` property specifies a URL with HTTPS, and set the `DIRECTORY_SSL` property to `SIMPLE`.
3. Run the AM configurator tool, `openam-configurator-tool-14.1.3.19.jar`:

```
java -jar openam-configurator-tool-14.1.3.19.jar --file config.properties
```

If required, you can specify additional runtime options on the command line:

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

Output similar to the following appears:

```
$ java -jar openam-configurator-tool-14.1.3.19.jar --file config.properties
Checking license acceptance...License terms accepted.
Checking configuration directory /home/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 uid=admin
--ldapPort 50389 --skipPortCheck --rootUserPassword xxxxxxx --jmxPort 1689
--no-prompt --doNotStart --hostname openam.example.com ...
...Success
Installing OpenAM configuration store in /home/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!
```

Deleting and Redeploying AM

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

To Delete a Configuration Before Redeploying

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

The following example shuts down Apache Tomcat:

```
$ /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 the AM configuration files, by default under the `$HOME` of the user running the web application container:

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

3. When installing or reinstalling a standalone AM instance, you must ensure the configuration store used does not contain previous configuration data.

Either install a new, clean instance of DS, or delete the entries under the configured AM suffix (by default `ou=am-config`) of an existing instance.

Note that when adding a server to an existing deployment, you must not delete the configuration from DS, as it is shared by all servers in the deployment. See "To Add a Server to a Site".

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

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

5. Restart the AM web application.

The following example starts the Tomcat container:

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

Chapter 8

Starting Instances

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

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

+ *How Is the Bootstrap File Created?*

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

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

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

+ *When Is the Bootstrap File Updated?*

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

If you change the configuration of the AM keystore, for example, the path to its files, AM will save the changes to the bootstrap file the next time it starts successfully.

This is why, if you want to override AM's startup settings, you need to replace the bootstrap file manually before AM starts.

Overriding Startup Settings

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

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

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

Replacing the Bootstrap File

AM's bootstrap file is located at the path `/path/to/openam/config/boot.json`, where `/path/to/openam` is the AM configuration directory.

+ *When/Where Is the AM Configuration Directory Specified?*

You specify it during AM installation, as follows:

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

To override AM's startup configuration, modify the bootstrap file, `boot.json`, and then overwrite the existing bootstrap file with your modified file *prior to every AM restart*.

You must overwrite the file each time you start AM because after startup, AM overwrites the bootstrap file with the initial startup settings created during AM installation, removing any modifications you might have made to startup settings in the bootstrap file.

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

+ *Bootstrap File Example, With Properties*

```
{
  "instance": "https://openam.example.com:8443/openam",
  "dsameUser": "cn=dsameuser,ou=DSAME Users,dc=openam,dc=forgerock,dc=org",
  "keystores": {
    "default": {
      "keyStorePasswordFile": "/path/to/openam/security/secrets/default/.storepass",
      "keyPasswordFile": "/path/to/openam/security/secrets/default/.keypass",
      "keyStoreType": "JCEKS",
      "keyStoreFile": "/path/to/openam/security/keystores/keystore.jceks"
    }
  },
  "configStoreList": [
    {
      "baseDN": "ou=am-config",
      "dirManagerDN": "uid=am-config,ou=admins,ou=am-config",
      "ldapHost": "opendj.example.com",
      "ldapPort": 1636,
      "ldapProtocol": "ldap"
    }
  ]
}
```

Startup Settings in the Bootstrap File

Property	Description and Derivation
<code>instance</code>	<p>AM 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 AM instance, <i>not</i> an AM site using a load balancer URL.</p> <p>Do not modify this bootstrap file property. If you need to change the AM instance URL, reinstall AM.</p>
<code>dsameUser</code>	<p>Special AM 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 AM 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 AM keystore. Always created initially as <code>/path/to/openam/security/secrets/default/.storepass</code>.</p>

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

Property	Description and Derivation
	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).

Overriding Startup Settings by Using Environment Variables

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

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

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

To Override Startup Settings by Using Environment Variables

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

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

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

Overriding Startup Settings by Using Java Properties

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

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

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

To Override Startup Settings by Using Java Properties

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

```
"keystores" : {  
  "default" : {  
    "keyStorePasswordFile" : "/Users/nabil.maynard/.storepass",  
    "keyPasswordFile" : "/Users/nabil.maynard/.keypass",  
    "keyStoreType" : "JCEKS",  
    "keyStoreFile" : "/Users/nabil.maynard/keystore.jceks"  
  }  
},
```

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

Chapter 9

Setting Up Administration Tools

AM provides a set of administration tools that are now deprecated in favor of Amster. They are part of the AM distributable file.

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

To Set Up Administration Tools

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

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

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

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

4. Unpack the tools:

```
$ cd /path/to/openam-tools/admin
$ unzip ~/Downloads/openam/AM-SSOAdminTools-5.1.3.19.zip
```

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

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

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

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

6. Run the **setup** utility (**setup.bat** on Windows) providing the location, password, and type of the truststore containing the public certificate of the DS configuration store.

Optionally, include the `--acceptLicense` option if you want to auto-accept the license agreement and suppress the license acceptance screen to the user.

+ How Do I Create the Truststore?

Even though you may have other truststores containing the public certificate of the configuration store, ForgeRock recommends that you use a truststore specifically for the **ssoadm** command.

To create it, follow the steps in "To Create a Truststore for AM", but *do not* configure the new truststore in the container. You will configure it in the **ssoadm** command script later.

If the container where AM runs is configured for secure connections and is using self-signed certificates, import that public certificate into the new truststore, too. For more information, see "To Share Self-Signed Certificates".

Note

When using self-signed certificates, for example, 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*.

You will also need to provide the paths to the directories where AM configuration files are located, and where the **ssoadm** debug and log information will be located. For example:

```
$ ./setup --truststore-path /my/ssoadm/truststore --truststore-password changeit --truststore-type JKS
--acceptLicense
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: ForgeRock Access Management version, Build, and date
```

Tip

If the **setup** utility cannot connect to the configuration store, it will show a message similar to the following:

```
Connect Error: No operational connection factories available
```

If you see this message, check that the truststore exists in the specified location, that it contains the configuration store certificate, and that the user running the **setup** utility can change directories to the specified location and open/read the file.

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

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

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

7. Edit the **ssoadm** script and configure the truststore containing the certificate of the configuration store. This truststore may also contain the certificate to connect to AM using SSL, if needed.
 - a. In the script, look for the following lines:

```
....
TRUSTSTORE="-Djavax.net.ssl.trustStore=$truststore_path"
TRUSTSTORE="$TRUSTSTORE -Djavax.net.ssl.trustStorePassword=$truststore_password"
TRUSTSTORE="$TRUSTSTORE -Djavax.net.ssl.trustStoreType=$truststore_type"
....
```

- b. Add the **truststore_path**, **truststore_password**, and **truststore_type** variables above the lines you found:

```
truststore_path=/my/ssoadm/truststore
truststore_password=changeit
truststore_type=JKS

TRUSTSTORE="-Djavax.net.ssl.trustStore=$truststore_path"
TRUSTSTORE="$TRUSTSTORE -Djavax.net.ssl.trustStorePassword=$truststore_password"
TRUSTSTORE="$TRUSTSTORE -Djavax.net.ssl.trustStoreType=$truststore_type"
....
```

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

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

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

9. Check that the **ssoadm** command works properly:
 - a. Create a text file, for example **\$HOME/.pwd.txt**, containing the AM administrative user's password string in cleartext on a single line.
 - b. Make the text file read-only:

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

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

```
$ cd /path/to/openam-tools/admin/openam/bin/
$ ./ssoadm list-servers --adminid uid=amAdmin,ou=People,dc=openam,dc=forgerock,dc=org --password-
file $HOME/.pwd.txt
https://openam.example.com:8443/openam
```

Note

The value for the `--adminid` parameter is the universal ID of an administrative user.

Administrative users are listed in the `com.sun.identity.authentication.super.user` or `com.sun.identity.authentication.special.users` advanced properties, under `Configure > Server Defaults > Advanced`.

The default super-user account is `uid=amAdmin,ou=People,%ROOT_SUFFIX%`. To check your `%ROOT_SUFFIX%` value, go to the `/path/to/openam/config/boot.json` file, and find the value for the `configStoreList/baseDN` property.

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

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

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

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

Important

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

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

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

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

Chapter 10

Next Steps

Congratulations on installing AM!

This list indicates the tasks to consider after installing AM:

Core Administrative Tasks

- Learn about realms, configure them, and connect them to identity stores.
- Configure AM's cookie domain.
- Learn about other types of configuration stores and decide if your environment would benefit from having dedicated application stores.

For more information, see the [Setup Guide](#).

Core Token Service Tasks

- Learn about the Core Token Service and decide if your environment would benefit from having dedicated CTS token stores.

For more information, see the [Core Token Service Guide \(CTS\)](#).

Access Management-Related Tasks

- Learn about authentication trees and nodes and configure them to let your users log in to AM.
- Learn about sessions in AM and configure them for your environment.

For more information, see the [Authentication and Single Sign-On Guide](#)

Security-Related Tasks

- Secure your core AM environment against different threats.
- Configure keys and keystores used for different AM features.
- Change the `amAdmin` user password.
- Learn about delegated privileges and configure delegated realm administrators.
- Configure audit logging services.

For more information, see the [Security Guide](#).

Maintenance-Related Tasks

- Learn how to back up and restore your environment.
- Learn how to monitor your AM instances.

- Learn how to enable debug logging and how to record troubleshooting information.
- Tune AM.

For more information, see the [Maintenance Guide](#).

AM offers authentication and authorization functionality, which you can expand with Internet specifications and drafts, such as OAuth 2.0, and SAML v2.0.

Once you are confident about your base AM configuration, move on to more advanced features, such as protecting web applications, configuring single sign-on (SSO), federating access across applications, and others.

Chapter 11

Uninstall AM

This topic describes how to remove AM software.

To remove a single server from a multi-server deployment, go to **Deployment > Servers > *Server Name*** then click the vertical ellipses and click **Delete**.

For instructions on removing AM agents, see the *ForgeRock Web Agents User Guide*, or the *ForgeRock Java Agent User Guide*.

To Remove an Instance

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

In a test deployment, the following steps remove the AM software and the internal configuration store. If you used an external configuration store, remove AM configuration data after removing all the software.

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

```
$ /etc/init.d/tomcat stop
Password:
Using CATALINA_BASE:   /path/to/tomcat
Using CATALINA_HOME:   /path/to/tomcat
Using CATALINA_TMPDIR: /path/to/tomcat/temp
Using JRE_HOME:        /path/to/jdk/jre
Using CLASSPATH:       /path/to/tomcat/bin/bootstrap.jar:
                        /path/to/tomcat/bin/tomcat-juli.jar
```

2. Unconfigure AM by removing the configuration files in the \$HOME directory of the user running the web application container.

A full uninstall of AM and configuration files consists of removing the following directories:

- The configuration directory, by default `$HOME/openam`. If you did not use the default configuration location, check the value of the Base installation directory property under **Deployment > Servers > *Server Name* > General > System**.
- The hidden directory that holds a file pointing to the configuration directory. For example, if you are using Apache Tomcat as the web container, this file could be `$HOME/.openamcfg/AMConfig_path_to_tomcat_webapps_openam_`.

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

3. Remove the configuration manually from your external directory server. The default base DN for the AM configuration is `dc=openam,dc=forgerock,dc=org`.

Note

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

4. Undeploy the AM web application.

For example, if you are using Apache Tomcat as the web container, remove the `.war` file and expanded web application from the container:

```
$ cd /path/to/tomcat/webapps/  
$ rm -rf openam.war openam/
```

Chapter 12

Troubleshooting

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

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

To Troubleshoot an Installation

Follow these steps to capture debug logs while installing AM on Tomcat:

1. If Tomcat is already started, stop it.
2. Specify the `-Dcom.iplanet.services.debug.level=message` option in the `CATALINA_OPTS` environment variable:

```
$ export CATALINA_OPTS=-Dcom.iplanet.services.debug.level=message
```

There are several ways that you can specify the `CATALINA_OPTS` environment variable. You can set the variable:

- In the `/path/to/tomcat/bin/setenv.sh` file
 - In the login shell of the user who runs Tomcat
3. Run the AM installation. Debug log files containing troubleshooting information appear in the `/path/to/openam/var/debug` directory.
 4. When you have completed AM installation and no longer need to capture debug logs, stop Tomcat, revert the debug logging options, and restart Tomcat.

Glossary

Access control	Control to grant or to deny access to a resource.
Account lockout	The act of making an account temporarily or permanently inactive after successive authentication failures.
Actions	Defined as part of policies, these verbs indicate what authorized identities can do to resources.
Advice	In the context of a policy decision denying access, a hint to the policy enforcement point about remedial action to take that could result in a decision allowing access.
Agent administrator	User having privileges only to read and write agent profile configuration information, typically created to delegate agent profile creation to the user installing a web or Java agent.
Agent authenticator	Entity with read-only access to multiple agent profiles defined in the same realm; allows an agent to read web service profiles.
Application	<p>In general terms, a service exposing protected resources.</p> <p>In the context of AM policies, the application is a template that constrains the policies that govern access to protected resources. An application can have zero or more policies.</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	AM authentication unit that handles one way of obtaining and verifying credentials.
Authorization	The act of determining whether to grant or to deny a principal access to a resource.
Authorization Server	In OAuth 2.0, issues access tokens to the client after authenticating a resource owner and confirming that the owner authorizes the client to access the protected resource. AM can play this role in the OAuth 2.0 authorization framework.
Auto-federation	Arrangement to federate a principal's identity automatically based on a common attribute value shared across the principal's profiles at different providers.
Bulk federation	Batch job permanently federating user profiles between a service provider and an identity provider based on a list of matched user identifiers that exist on both providers.
Circle of trust	Group of providers, including at least one identity provider, who have agreed to trust each other to participate in a SAML v2.0 provider federation.
Client	In OAuth 2.0, requests protected web resources on behalf of the resource owner given the owner's authorization. AM can play this role in the OAuth 2.0 authorization framework.
Client-based OAuth 2.0 tokens	After a successful OAuth 2.0 grant flow, AM returns a token to the client. This differs from CTS-based OAuth 2.0 tokens, where AM returns a <i>reference</i> to token to the client.
Client-based sessions	AM sessions for which AM returns session state to the client after each request, and require it to be passed in with the subsequent

	<p>request. For browser-based clients, AM sets a cookie in the browser that contains the session information.</p> <p>For browser-based clients, AM sets a cookie in the browser that contains the session state. When the browser transmits the cookie back to AM, AM decodes the session state from the cookie.</p>
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> <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.</p>
Configuration datastore	LDAP directory service holding AM configuration data.
Cross-domain single sign-on (CDSSO)	AM capability allowing single sign-on across different DNS domains.
CTS-based OAuth 2.0 tokens	After a successful OAuth 2.0 grant flow, AM returns a <i>reference</i> to the token to the client, rather than the token itself. This differs from client-based OAuth 2.0 tokens, where AM returns the entire token to the client.
CTS-based sessions	AM sessions that reside in the Core Token Service's token store. CTS-based sessions might also be cached in memory on one or more AM servers. AM tracks these sessions in order to handle events like logout and timeout, to permit session constraints, and to notify applications involved in SSO when a session ends.
Delegation	Granting users administrative privileges with AM.
Entitlement	Decision that defines which resource names can and cannot be accessed for a given identity in the context of a particular application, which actions are allowed and which are denied, and any related advice and attributes.
Extended metadata	Federation configuration information specific to AM.
Extensible Access Control Markup Language (XACML)	Standard, XML-based access control policy language, including a processing model for making authorization decisions based on policies.
Federation	Standardized means for aggregating identities, sharing authentication and authorization data information between trusted providers, and

	allowing principals to access services across different providers without authenticating repeatedly.
Fedlet	Service provider application capable of participating in a circle of trust and allowing federation without installing all of AM on the service provider side; AM lets you create Java Fedlets.
Hot swappable	Refers to configuration properties for which changes can take effect without restarting the container where AM runs.
Identity	Set of data that uniquely describes a person or a thing such as a device or an application.
Identity federation	Linking of a principal's identity across multiple providers.
Identity provider (IDP)	Entity that produces assertions about a principal (such as how and when a principal authenticated, or that the principal's profile has a specified attribute value).
Identity repository	Data store holding user profiles and group information; different identity repositories can be defined for different realms.
Java agent	Java web application installed in a web container that acts as a policy enforcement point, filtering requests to other applications in the container with policies based on application resource URLs.
Metadata	Federation configuration information for a provider.
Policy	Set of rules that define who is granted access to a protected resource when, how, and under what conditions.
Policy agent	Java, web, or custom agent that intercepts requests for resources, directs principals to AM for authentication, and enforces policy decisions from AM.
Policy Administration Point (PAP)	Entity that manages and stores policy definitions.
Policy Decision Point (PDP)	Entity that evaluates access rights and then issues authorization decisions.
Policy Enforcement Point (PEP)	Entity that intercepts a request for a resource and then enforces policy decisions from a PDP.
Policy Information Point (PIP)	Entity that provides extra information, such as user profile attributes that a PDP needs in order to make a decision.
Principal	Represents an entity that has been authenticated (such as a user, a device, or an application), and thus is distinguished from other entities.

	When a Subject successfully authenticates, AM associates the Subject with the Principal.
Privilege	In the context of delegated administration, a set of administrative tasks that can be performed by specified identities in a given realm.
Provider federation	Agreement among providers to participate in a circle of trust.
Realm	AM unit for organizing configuration and identity information. Realms can be used for example when different parts of an organization have different applications and identity stores, and when different organizations use the same AM deployment. Administrators can delegate realm administration. The administrator assigns administrative privileges to users, allowing them to perform administrative tasks within the realm.
Resource	Something a user can access over the network such as a web page. Defined as part of policies, these can include wildcards in order to match multiple actual resources.
Resource owner	In OAuth 2.0, entity who can authorize access to protected web resources, such as an end user.
Resource server	In OAuth 2.0, server hosting protected web resources, capable of handling access tokens to respond to requests for such resources.
Response attributes	Defined as part of policies, these allow AM to return additional information in the form of "attributes" with the response to a policy decision.
Role based access control (RBAC)	Access control that is based on whether a user has been granted a set of permissions (a role).
Security Assertion Markup Language (SAML)	Standard, XML-based language for exchanging authentication and authorization data between identity providers and service providers.
Service provider (SP)	Entity that consumes assertions about a principal (and provides a service that the principal is trying to access).
Authentication Session	The interval while the user or entity is authenticating to AM.
Session	The interval that starts after the user has authenticated and ends when the user logs out, or when their session is terminated. For browser-based clients, AM manages user sessions across one or more applications by setting a session cookie. See also CTS-based sessions and Client-based sessions.

Session high availability	Capability that lets any AM server in a clustered deployment access shared, persistent information about users' sessions from the CTS token store. The user does not need to log in again unless the entire deployment goes down.
Session token	Unique identifier issued by AM after successful authentication. For a CTS-based sessions, the session token is used to track a principal's session.
Single log out (SLO)	Capability allowing a principal to end a session once, thereby ending her session across multiple applications.
Single sign-on (SSO)	Capability allowing a principal to authenticate once and gain access to multiple applications without authenticating again.
Site	<p>Group of AM servers configured the same way, accessed through a load balancer layer. The load balancer handles failover to provide service-level availability.</p> <p>The load balancer can also be used to protect AM services.</p>
Standard metadata	Standard federation configuration information that you can share with other access management software.
Stateless Service	<p>Stateless services do not store any data locally to the service. When the service requires data to perform any action, it requests it from a data store. For example, a stateless authentication service stores session state for logged-in users in a database. This way, any server in the deployment can recover the session from the database and service requests for any user.</p> <p>All AM services are stateless unless otherwise specified. See also Client-based sessions and CTS-based sessions.</p>
Subject	<p>Entity that requests access to a resource</p> <p>When an identity successfully authenticates, AM associates the identity with the Principal that distinguishes it from other identities. An identity can be associated with multiple principals.</p>
Identity store	Data storage service holding principals' profiles; underlying storage can be an LDAP directory service or a custom IdRepo implementation.
Web Agent	Native library installed in a web server that acts as a policy enforcement point with policies based on web page URLs.