

OpenAM Installation Guide

MarkCraig
VanessaRichie
Mikejang

,
,,

Copyright © 2011-2013 ForgeRock AS

Abstract

Guide showing you how to install OpenAM. OpenAM provides open source Authentication, Authorization, Entitlement and Federation software.



This work is licensed under the [Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License](http://creativecommons.org/licenses/by-nc-nd/3.0/).

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

ForgeRock™ is the trademark of ForgeRock Inc. or its subsidiaries in the U.S. and in other countries. Trademarks are the property of their respective owners.

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

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

DejaVu Fonts

Bitstream Vera Fonts Copyright

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

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

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

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

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

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

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

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

Arev Fonts Copyright

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

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

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

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

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

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

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

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

Admonition graphics by Yannick Lung. Free for commercial use. Available at [Freeens Cumulus](http://Freeens.Cumulus).

Table of Contents

Preface	v
1. Who Should Use this Guide	v
2. Formatting Conventions	v
3. Accessing Documentation Online	vi
4. Joining the Open Identity Platform Community	vii
1. Preparing For Installation	1
1.1. Preparing a Fully-Qualified Domain Name	1
1.2. Preparing a Java Environment	2
1.3. Setting Maximum File Descriptors	3
1.4. Preparing a Configuration Data Store	3
1.5. Preparing an Identity Repository	7
1.6. Obtaining OpenAM Software	9
1.7. Preparing Apache Tomcat	10
1.8. Preparing GlassFish	11
1.9. Preparing OpenAM & JBoss 4 or 5	13
1.10. Preparing OpenAM & JBoss AS 7 / EAP 6	13
1.11. Preparing Jetty	18
1.12. Preparing Oracle WebLogic	18
1.13. Preparing IBM WebSphere	19
2. Installing OpenAM Core Services	21
3. Installing OpenAM Tools	37
4. Installing OpenAM Distributed Authentication	43
5. Customizing the OpenAM End User Pages	49
5.1. Updating the Classic UI	50
5.2. How OpenAM Looks Up UI Files	52
5.3. Configuring the XUI	55
6. Configuring the Core Token Service (CTS)	59
6.1. CTS Configuration Parameters	60
6.2. Preparing an OpenDJ Directory Service for CTS	62
6.3. CTS Access Control Instructions	64
6.4. CTS and OpenDJ Replication	65
6.5. CTS Deployment Scenario	66
6.6. Managing CTS Tokens	67
6.7. General Recommendations for CTS Configuration	68
7. Setting Up OpenAM Session Failover	71
8. Removing OpenAM Software	75
Index	77

Preface

This guide shows you how to install core OpenAM services for access and federation management. Unless you are planning a throwaway evaluation or test installation, read the *Release Notes* before you get started.

1 Who Should Use this Guide

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

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

You do not need to be an OpenAM wizard to learn something from this guide, though a background in access management and maintaining web application software can help. You do need some background in managing services on your operating systems and in your application servers. You can nevertheless get started with this guide, and then learn more as you go along.

2 Formatting Conventions

Most examples in the documentation are created in GNU/Linux or Mac OS X operating environments. If distinctions are necessary between operating environments, examples are labeled with the operating environment name in

parentheses. To avoid repetition file system directory names are often given only in UNIX format as in /path/to/server, even if the text applies to C:\path\to\server as well.

Absolute path names usually begin with the placeholder /path/to/. This path might translate to /opt/, C:\Program Files\, or somewhere else on your system.

Command line, terminal sessions are formatted as follows:

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

Command output is sometimes formatted for narrower, more readable output even though formatting parameters are not shown in the command. In the following example, the query string parameter `_prettyPrint=true` is omitted and some of the output is replaced with an ellipsis (. . .):

```
$ curl https://bjensen:hifalutin@opendj.example.com:8443/users/newuser
{
  "_rev" : "000000005b337348",
  "_id" : "newuser",
  ...
}
```

Program listings are formatted as follows:

```
class Test {
  public static void main(String [] args) {
    System.out.println("This is a program listing.");
  }
}
```

3 Accessing Documentation Online

Open Identity Platform core documentation, such as this document, aims to be technically accurate and complete with respect to the software documented.

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

- Product managers and software architects review project documentation design with respect to the readers' software lifecycle needs.
- Subject matter experts review proposed documentation changes for technical accuracy and completeness with respect to the corresponding software.

- Quality experts validate implemented documentation changes for technical accuracy, completeness in scope, and usability for the readership.

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

Fully reviewed, published core documentation is available at <https://doc.openidentityplatform.org/>. Use this documentation when working with a Open Identity Platform release.

You can find pre-release draft documentation at the online [community resource center](#). Use this documentation when trying a nightly build.

4 Joining the Open Identity Platform Community

Visit the [community resource center](#) where you can find information about each project, download nightly builds, browse the resource catalog, ask and answer questions on the forums, find community events near you, and of course get the source code as well.

Chapter 1

Preparing For Installation

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



Note

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

1.1 Preparing a Fully-Qualified Domain Name

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

Do not use the `localhost` domain for OpenAM, not even for testing purposes. OpenAM relies on browser cookies, which are returned based on domain name.

Furthermore, use a domain name that contains at least 2 . (dot) characters, such `openam.example.com`.

1.2 Preparing a Java Environment

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

1.2.1 Settings For Sun/Oracle Java Environments

When using a Sun or Oracle Java environment set at least the following options.

`-server`

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

`-XX:MaxPermSize=256m`

Set the permanent generation size to 256 MB.

`-Xmx1024m` (minimum)

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

For additional JVM tuning recommendations, see *Java Virtual Machine Settings*.

1.2.2 Settings For IBM Java Environments

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

`-DamCryptoDescriptor.provider=IBMJCE`

`-DamKeyGenDescriptor.provider=IBMJCE`

Use the IBM Java Cryptography Extensions.

`-Xmx1024m` (minimum)

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

1.3 Setting Maximum File Descriptors

If you use the embedded OpenDJ directory, make sure OpenDJ has enough file descriptors. OpenDJ needs to be able to open many files, especially when handling many client connections. Linux systems in particular often set a limit of 1024 per user, which is too low for OpenDJ.

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

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

```
$ ulimit -n
65536
```

The example above assumes the system has enough file descriptors overall. You can verify the new soft limit the next time you log in as user `openam` with the **`ulimit -n`** command.

You can check the Linux system overall maximum as follows.

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

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

1. As superuser, edit `/etc/sysctl.conf` to set the kernel parameter `fs.file-max` to a higher maximum.
2. Run the **`sysctl -p`** command to reload the settings in `/etc/sysctl.conf`.
3. Read `/proc/sys/fs/file-max` again to confirm that it now corresponds to the new maximum.

1.4 Preparing a Configuration Data Store

OpenAM stores configuration, session, and token data in an LDAP directory service. This data is private to OpenAM. In other words, OpenAM controls this data and other applications should access it, if necessary, only through OpenAM.

OpenAM ships with an embedded OpenDJ directory server that you can install as part of the OpenAM configuration process. You can use the embedded directory

server to simplify evaluation. By default OpenAM installs the embedded directory alongside configuration settings under the \$HOME of the user running OpenAM, and runs the embedded directory in the same memory space as OpenAM. Before deploying OpenAM in production, measure the impact of using the embedded directory not only for relatively static configuration data, but also for volatile session and token data. Your tests should subject OpenAM to the same load patterns you expect in production. If it looks like a better choice to use an external directory service, then use one of the supported external configuration stores listed in the *Release Notes*, such as OpenDJ.

With the embedded OpenDJ directory and the default configuration settings, OpenAM connects as directory super user, bypassing access control evaluation because OpenAM manages the directory as its private store. Be aware that failover and replication can not be controlled when using the embedded store.

OpenAM now supports the use of the Core Token Service (CTS), with tokens that can be stored in the local or external directory store. For more information, see the chapter on [Configuring the Core Token Service](#).

With an external directory service, the directory administrator can require OpenAM to connect with normal application credentials. In that case, the directory administrator must grant OpenAM specific access.



Tip

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

- OpenAM requires specific directory schema definitions for the object classes and attribute types that describe its data. For the configuration store, the directory administrator should let OpenAM update the directory schema at configuration time.

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

To grant the required access with OpenDJ for example, first add a global access control instruction (ACI) permitting the OpenAM user to modify schema definitions as in the following example where the OpenAM entry has DN `uid=openam,ou=admins,dc=example,dc=com`.

Preparing a Configuration Data Store

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

Also give the OpenAM user privileges to modify the schema and write to subentries such as the schema entry. Set the following attributes on the OpenAM user entry.

```
ds-privilege-name: subentry-write
ds-privilege-name: update-schema
```

See the OpenDJ documentation about [Configuring Privileges & Access Control](#) for a more in-depth explanation of how access is configured for OpenDJ.

- When OpenAM connects to an external directory service to store its data, it requires both read and write access.

With OpenDJ for example, add following ACIs to the configuration base Distinguished Name (DN) entry. Adjust them as necessary if the OpenAM user DN differs from uid=openam,ou=admins,dc=example,dc=com.

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

In addition, the directory administrator should index the following attributes used by OpenAM.

Table 1.1. Configuration Data Store Indexes

Attribute	Indexes Required
coreTokenDate01	equality
coreTokenDate02	equality
coreTokenDate03	equality
coreTokenDate04	equality
coreTokenDate05	equality

Preparing a
Configuration Data Store

Attribute	Indexes Required
coreTokenExpirationDate	ordering
coreTokenInteger01	equality
coreTokenInteger02	equality
coreTokenInteger03	equality
coreTokenInteger04	equality
coreTokenInteger05	equality
coreTokenInteger06	equality
coreTokenInteger07	equality
coreTokenInteger08	equality
coreTokenInteger09	equality
coreTokenInteger10	equality
coreTokenString01	equality
coreTokenString02	equality
coreTokenString03	equality
coreTokenString04	equality
coreTokenString05	equality
coreTokenString06	equality
coreTokenString07	equality
coreTokenString08	equality
coreTokenString09	equality
coreTokenString10	equality
coreTokenString11	equality
coreTokenString12	equality
coreTokenString13	equality

Attribute	Indexes Required
coreTokenString14	equality
coreTokenString15	equality
coreTokenUserId	equality
iplanet-am-user-federation-info-key	equality
sun-fm-saml2-nameid-infokey	equality
sunxmlkeyvalue	equality, substring

1.5 Preparing an Identity Repository

OpenAM stores user identity data in one or more identity repositories. In many deployments OpenAM connects to existing LDAP directory services for user identity data. OpenAM is designed therefore to share data in an identity repository with other applications.

OpenAM ships with an embedded OpenDJ directory server that you can install as part of the OpenAM configuration process. In deployments where you will only ever have a few users to manage and do not need to share identity data with other applications, you can use the embedded store as your identity repository and avoid the additional overhead of maintaining a separate directory service. If OpenAM will share identity data with other applications, or if you expect to have lots of users, then connect OpenAM to an external identity repository. See the *Release Notes* for a list of supported external identity repositories.

When OpenAM connects to an external identity repository, the administrator must give OpenAM the following access rights.

- OpenAM requires specific directory schema definitions for the object classes and attribute types that describe its data. The directory administrator can find these definitions in the `ldif` directory found inside the full `.zip` delivery.

If the directory administrator chooses instead to have OpenAM update the directory schema at configuration time, then the directory administrator must grant OpenAM access.

To grant this access right with OpenDJ for example, first add a global ACI permitting the OpenAM user to modify schema definitions as in the following example where the OpenAM entry has DN `uid=openam,ou=admin,dc=example,dc=com`.

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

Also give the OpenAM user privileges to modify the schema and write to subentries such as the schema entry. Set the following attributes on the OpenAM user entry.

```
ds-privilege-name: subentry-write
ds-privilege-name: update-schema
```

- Allow OpenAM to read directory schema.

With OpenDJ for example, keep the default "User-Visible Schema Operational Attributes" global ACI.

- When OpenAM connects to an external identity repository, it requires access to read and potentially to update data.

To grant the access rights with OpenDJ for example, add following ACIs to the configuration base DN entry. Adjust them as necessary if the OpenAM user DN differs from `uid=openam,ou=admins,dc=example,dc=com`.

```
aci: (targetattr="* || aci")(version 3.0;acl "Allow identity modification";
allow (write)(userdn = "ldap:///uid=openam,ou=admins,dc=example,dc=com");)
aci: (targetattr!="userPassword||authPassword")(version 3.0;
acl "Allow identity search"; allow (search, read)(userdn = "ldap:///
uid=openam,ou=admins,dc=example,dc=com");)
aci: (targetcontrol="2.16.840.1.113730.3.4.3")(version 3.0;acl "Allow
persistent search"; allow (search, read)(userdn = "ldap:///
uid=openam,ou=admins,dc=example,dc=com");)
aci: (version 3.0;acl "Add identity"; allow (add)(userdn = "ldap:///
uid=openam,ou=admins,dc=example,dc=com");)
aci: (version 3.0;acl "Delete identity"; allow (delete)(userdn = "ldap:///
uid=openam,ou=admins,dc=example,dc=com");)
```

- Allow the OpenAM user to reset other users' passwords.

To grant this privilege in OpenDJ for example, set the following attribute on the OpenAM user entry.

```
ds-privilege-name: password-reset
```

In addition for external directory services, the directory administrator should index the following attributes used by OpenAM.

Table 1.2. Identity Repository Indexes

Attribute	Indexes Required
iplanet-am-user-federation-info-key	equality
sun-fm-saml2-nameid-infokey	equality

1.6 Obtaining OpenAM Software

Download OpenAM releases from one of the following locations:

- [Enterprise Downloads](#) has the latest stable version of OpenAM, including a .zip file with all of the OpenAM components, the .war file, OpenAM tools, the configurator, policy agents, OpenIG, and documentation. Make sure you review the Software License and Subscription Agreement presented before you download OpenAM files.
- [Builds](#) has the nightly build, including a .zip file with all of the OpenAM components, the .war file, OpenAM tools, the configurator, policy agents, and the .NET Fedlet. Be aware that this is the working version of the trunk and should not be used in a production environment.
- [Archives](#) has old versions of OpenAM and policy agents. It includes the full .zip file with all of the OpenAM components, the server .war file, OpenAM tools, the configurator, policy agents, the WSS policy agents, and the .NET Fedlet for all previous releases.

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

After you download the .zip file, create a new openam folder, and unzip the .zip file to access the content:

```
$ cd ~/Downloads
$ mkdir openam ; cd openam
$ unzip ~/Downloads/
```

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

The OpenAM Java client SDK library

The .zip file containing the Java client SDK command-line examples, and .jar files needed to run the examples

The .war file containing Java client SDK examples in a web application

The IDP discovery .war file, deployed as a service to service providers that must discover which identity provider corresponds to a SAML 2.0 request

For details, see *Deploying the Identity Provider Discovery Service*.

The .zip that contains the lightweight service provider implementations that you can embed in your Java EE or ASP.NET applications to enable it to use federated access management

The deployable .war file

The deployable .war file for distributed authentication

The deployable .war file when you want to deploy OpenAM server without the OpenAM console

The .zip file that contains tools to manage OpenAM from the command line

The .zip file that contains tools to configure OpenAM from the command line

1.7 Preparing Apache Tomcat

OpenAM examples often use Apache Tomcat as the deployment container. Tomcat is installed on `openam.example.com`, and listens on the default ports, with no Java Security Manager enabled. The script `/etc/init.d/tomcat` manages the service at system startup and shutdown. This script assumes you run OpenAM as the user `openam`.

OpenAM core services require a minimum JVM heap size of 1 GB, and a permanent generation size of 256 MB. If you are including the embedded OpenDJ directory, OpenAM requires at least a 2 GB heap, as 50% of that space is allocated to OpenDJ. See [Section 1.2, “Preparing a Java Environment”](#) for details.

```
#!/bin/sh
#
# tomcat
#
# chkconfig: 345 95 5
# description: Manage Tomcat web application container
CATALINA_HOME="/path/to/tomcat"
export CATALINA_HOME
JAVA_HOME=/path/to/jdk
export JAVA_HOME
CATALINA_OPTS="-server -Xmx2048m -XX:MaxPermSize=256m"
export CATALINA_OPTS

case "${1}" in
start)
    /bin/su openam -c "${CATALINA_HOME}/bin/startup.sh"
    exit $?
    ;;
stop)
    /bin/su openam -c "${CATALINA_HOME}/bin/shutdown.sh"
    exit $?
    ;;
*)
    echo "Usage: $0 { start | stop }"
    exit 1
    ;;
esac
```

1.8 Preparing GlassFish

Before you deploy OpenAM, update the JVM options as described in [Section 1.2, “Preparing a Java Environment”](#). The settings are accessible in the administration console under Application Server > JVM Settings > JVM Options for v2, or under Configurations > server-config > JVM Settings > JVM Options for v3.

1.8.1 Preparing GlassFish v2

In addition to setting JVM options, after downloading the OpenAM server .war file, edit the application configuration to make sure that classes from OpenAM libraries are loaded before GlassFish bundled libraries.

1. Extract the OpenAM server .war file content to a working directory.

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

2. Add a WEB-INF/sun-web.xml file to set class-loading delegation to false.

```
$ vi WEB-INF/sun-web.xml
$ cat WEB-INF/sun-web.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Application Server 9.0 Servlet 2.5//EN"
"http://www.sun.com/software/appserver/dtds/sun-web-app_2_5-0.dtd">
<sun-web-app error-url="">
  <class-loader delegate="false"/>
</sun-web-app>
```

3. Pack the updated .war file to deploy.

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

4. Deploy the updated .war file in place of the server .war file delivered with the release.

1.8.2 Preparing GlassFish v3

In addition to setting JVM options, remove the glassfish-full-profile and metro packages to resolve library conflicts before you deploy OpenAM.

1. Stop GlassFish if it is running.

```
$ /path/to/glassfish3/bin/asadmin stop-domain domain1
Waiting for the domain to stop ....
Command stop-domain executed successfully.
```

2. Remove the packages by using the **pkg** command.

```
$ cd /path/to/glassfish3/bin/
./pkg uninstall glassfish-full-profile metro
PHASE                               ACTIONS
Removal Phase                       56/56
```

3. Start GlassFish.

```
$ /path/to/glassfish3/bin/asadmin start-domain domain1
Waiting for domain1 to start ...
Successfully started the domain : domain1
domain Location: /path/to/glassfish3/glassfish/domains/domain1
Log File: /path/to/glassfish3/glassfish/domains/domain1/logs/server.log
Admin Port: 4848
Command start-domain executed successfully.
```

If the domain fails to start the first time you run the command, then run the **asadmin start-domain** command again.

1.9 Preparing OpenAM & JBoss 4 or 5

OpenAM must be able to store its configuration between restarts. If you plan to deploy OpenAM as a single archive file, then unpack the .war, edit WEB-INF/classes/bootstrap.properties to set the configuration.dir property to the location where OpenAM has write access to store its configuration.

```
$ mkdir openam
$ cd openam
$ jar -xf ~/Downloads/openam/
$ vi WEB-INF/classes/bootstrap.properties
$ grep ^config WEB-INF/classes/bootstrap.properties
configuration.dir=/home/jboss-user/openam
```

Also, OpenAM .jar libraries that conflict with JBoss libraries must be loaded first. Add a WEB-INF/jboss-web.xml to ensure this happens. (If you deploy the exploded .war, you also need to add this file.)

```
$ vi WEB-INF/jboss-web.xml
$ cat WEB-INF/jboss-web.xml
```

```
<!DOCTYPE jboss-web PUBLIC
"-//JBoss//DTD Web Application 5.0//EN"
"http://www.jboss.org/j2ee/dtd/jboss-web_5_0.dtd">
<jboss-web>
  <class-loading java2ClassLoadingCompliance='true'>
    <loader-repository>
      jbia.loader:loader=opensso
      <loader-repository-config>java2ParentDelegaton=true</loader-repository-config>
    </loader-repository>
  </class-loading>
</jboss-web>
```

Repack the .war file that you can then deploy.

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

Before you deploy OpenAM, update the JVM options as described in [Section 1.2, “Preparing a Java Environment”](#).

1.10 Preparing OpenAM & JBoss AS 7 / EAP 6

Some preparation is required to deploy OpenAM on JBoss AS 7 / EAP 6.

The following instructions provide guidance for both standalone and domain deployments. OpenAM must be able to store its configuration between restarts. The procedures listed here are workarounds for JBoss AS 7.1.2 / 7.1.3, and the corresponding versions of JBoss EAP (6.0.0, 6.0.1). Workarounds are also needed

for JBoss EAP 6.1.0/6.1.1. To identify the versions of JBoss EAP that have been built from JBoss AS, see the following article on [JBoss Enterprise Application Platform Component Details](#).

Once JBoss has been configured, you can then prepare OpenAM for deployment, by making a few changes to the contents of the OpenAM .war archive.

- [Procedure 1.1, “To Prepare JBoss AS 7.1.0 / 7.1.1”](#)
- [Procedure 1.2, “Alternative Method: To Prepare JBoss EAP 6.0.0 and 6.0.1”](#)
- [Procedure 1.3, “To Prepare JBoss EAP 6.1.0 and 6.1.1”](#)
- [Procedure 1.4, “To Prepare JBoss for OpenAM”](#)
- [Procedure 1.5, “To Prepare OpenAM for JBoss”](#)

Procedure 1.1. To Prepare JBoss AS 7.1.0 / 7.1.1

For JBoss AS 7.1.0 / 7.1.1, you need to make changes to the `module.xml` file in the `/path/to/jboss/modules/sun/jdk/main` directory, as well as changes to a configuration file associated with JBoss standalone or domain modes.

1. Stop JBoss
2. Update the `module.xml` file associated with the container. You can find this file a directory such as `/path/to/jboss/modules/sun/jdk/main`.
3. In the same `module.xml` file, add the Sun x509 security module path (`sun/security/x509`).

The following example shows an excerpt of the revised file for JBoss AS 7.1.0.

```
<path name="com/sun/security/auth"/>
<path name="com/sun/security/auth/login"/>
<path name="com/sun/security/auth/module"/>
<path name="sun/security/x509"/> <!-- path added here -->
<path name="sun/misc"/>
```

4. When using **ssoadm** or the distributed authentication service (DAS), also add the following path to the aforementioned `module.xml` file.

```
<path name="com/sun/org/apache/xerces/internal/dom" />
```

5. Disable modules that conflict with OpenAM REST libraries. All `jaxrs` references need to be removed from the configuration. The file that you modify depends on whether you are running JBoss in standalone or domain mode.

- The following example is based on JBoss 7.1.0 standalone mode. Remember to remove all subsystems and extension tags associated with `urn:jboss:domain:jaxrs:1.0`.

```
$ vi /path/to/jboss/standalone/configuration/standalone.xml
<extension module="org.jboss.as.ejb3"/>
- <extension module="org.jboss.as.jaxrs"/>
....
- <subsystem xmlns="urn:jboss:domain:jaxrs:1.0"/>
<subsystem xmlns="urn:jboss:domain:jca:1.1">
```

- The following example is based on JBoss 7.1.0 for a managed domain.

```
$ vi /path/to/jboss7/domain/configuration/domain.xml
<extension module="org.jboss.as.ejb3"/>
- <extension module="org.jboss.as.jaxrs"/>
....
- <subsystem xmlns="urn:jboss:domain:jaxrs:1.0"/>
<subsystem xmlns="urn:jboss:domain:jca:1.1">
```

6. In either the `standalone.xml` or `domain.xml` files, you will also need to delete `org.jboss.as.webservices` references. Depending on the file, this includes one or more groups of subsystem lines such as:

```
<subsystem xmlns="urn:jboss:domain:webservices:1.1"/>
....
</subsystem>
```

7. You are now ready to prepare OpenAM as described in [Procedure 1.5, "To Prepare OpenAM for JBoss"](#).

Procedure 1.2. Alternative Method: To Prepare JBoss EAP 6.0.0 and 6.0.1

JBoss EAP 6.0.0 and 6.0.1 are built from JBoss AS 7.1.2 and 7.1.3, respectively. The same techniques described in the [Procedure 1.1, "To Prepare JBoss AS 7.1.0 / 7.1.1"](#) section work here as well. One alternative method is available, as described in this section.

1. Stop JBoss.
2. Update the `openam.war` before deploying OpenAM.
 1. Create a temporary directory and expand the `openam.war`.

```
$ mkdir /tmp/openam ; cd /tmp/openam
$ jar xvf /path/to/
```

2. Create a new `jboss-deployment-structure.xml` file in the `WEB-INF` subdirectory so that it appears as follows, and save the change.

```
$ vi WEB-INF/jboss-deployment-structure.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-deployment-structure xmlns="urn:jboss:deployment-structure:1.2">
  <deployment>
    <exclusions>
      <module name="sun.jdk" />
    </exclusions>
    <exclude-subsystems>
      <subsystem name="jaxrs" />
      <subsystem name="webservices" />
    </exclude-subsystems>
    <dependencies>
      <module name="sun.jdk" >
        <imports>
          <exclude-set>
<path name="com/sun/org/apache/xml/internal/security/transforms/implementations"/>
          </exclude-set>
        </imports>
      </module>
      <system>
        <paths>
          <path name="sun/security/x509" />
          <path name="com/sun/org/apache/xpath/internal" />
          <path name="com/sun/org/apache/xerces/internal/dom" />
          <path name="com/sun/org/apache/xml/internal/utils" />
        </paths>
      </system>
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

3. Rebuild the openam.war file.

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

3. You will want to make at least one more change to the openam.war file before deployment, as described in [Procedure 1.5, "To Prepare OpenAM for JBoss"](#).
4. You do not need to make any of the other changes to XML files described in this section. As JBoss EAP 6.0.0 and 6.0.1 was built from JBoss AS 7.1.2 and AS 7.1.3, respectively, this procedure may also work on those versions of JBoss.

Procedure 1.3. To Prepare JBoss EAP 6.1.0 and 6.1.1

1. For JBoss EAP 6.1.0 / 6.1.1, follow [Step 5](#) and [Step 6](#) from [Procedure 1.1, "To Prepare JBoss AS 7.1.0 / 7.1.1"](#).
2. However, you still need to review [Procedure 1.4, "To Prepare JBoss for OpenAM"](#) and [Procedure 1.5, "To Prepare OpenAM for JBoss"](#) to make sure the JVM and directories are configured appropriately.

Procedure 1.4. To Prepare JBoss for OpenAM

The default JBoss settings for JVM do not give sufficient memory to OpenAM. This procedure documents one method that you can use to modify JBoss. Other methods described in [JBoss Main Documentation Page](#).

1. Stop JBoss.
2. Open an appropriate JBoss configuration file. This procedure describes the use of the `standalone.conf` file in the `/path/to/jboss/bin` directory for JBoss in standalone mode.
3. Check the JVM settings associated with `JAVA_OPTS`. For JBoss AS 7.1.0 and AS 7.1.1, you should change the JVM heap size to `-Xmx1024m`. The default JVM heap size and permanent generation settings for later versions of JBoss may already exceed recommended values (`-Xmx1024m`, `-XX:MaxPermSize=256m`). If you are using the embedded version of OpenDJ, the minimum heap size may be higher. For details on the JVM options to use, see [Section 1.2, "Preparing a Java Environment"](#).
4. Set the following JVM `JAVA_OPTS` setting in the same file.

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

Make sure that headers include the Expires attribute rather than only Max-Age, as some versions of Internet Explorer do not support Max-Age.

5. Now deploy the `openam.war` file into the appropriate JBoss deployment directory. The directory varies depending on whether you are running in standalone or domain mode.
6. You do not need to make any of the other changes to XML files described in this section. As JBoss EAP 6.0.0 and 6.0.1 was built from JBoss AS 7.1.2 and AS 7.1.3, respectively, this procedure may also work on those versions of JBoss.

Procedure 1.5. To Prepare OpenAM for JBoss

To take full advantage of JBoss with OpenAM, you should make a couple of changes to the OpenAM war file. One problem is that JBoss will deploy applications from different temporary directories every time you restart the container, which would require reconfiguring OpenAM. To avoid this issue, take the following steps:

1. If you have not already done so, create a temporary directory and expand the `openam.war`.

```
$ cd /tmp
$ mkdir /tmp/openam ; cd /tmp/openam
$ jar xvf ~/Downloads/
```

2. Update the # configuration.dir= line in the bootstrap.properties file so that it appears as follows, and save the change.

```
$ vi WEB-INF/classes/bootstrap.properties
# 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=$HOME/openamJboss
```

3. Rebuild the openam.war file.

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

1.11 Preparing Jetty

When you deploy OpenAM, make sure you start Jetty with enough memory.

```
$ cd /path/to/jetty
$ java -server -Xmx1024m -XX:MaxPermSize=256m -jar start.jar
```

If you are using the embedded version of OpenDJ, the required JVM memory may be higher. For details on the JVM options to use, see [Section 1.2, “Preparing a Java Environment”](#).

1.12 Preparing Oracle WebLogic

Before you deploy OpenAM, update the JVM options as described in [Section 1.2, “Preparing a Java Environment”](#).

In addition, edit the WebLogic domain configuration to allow basic authentication credentials to be passed back to OpenAM.

By default WebLogic attempts to resolve authentication credentials itself. When you change the WebLogic domain configuration, you make sure that OpenAM OAuth 2.0 providers receive basic authentication credentials for OAuth 2.0 grants that rely on basic authentication.

1. Stop WebLogic server.

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

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

3. Start WebLogic server.

When deploying OpenAM on WebLogic 11g (version 10.3.x), use the SOAP with Attachments API for Java (SAAJ) implementation from the Java Runtime Environment, rather than the WebLogic implementation. The WebLogic implementation can cause OpenAM to throw exceptions with the message `java.lang.UnsupportedOperationException: This class does not support SAAJ 1.1`, and to fail to authenticate users in some cases. No change is necessary when deploying OpenAM on WebLogic 12c.

To use the Sun/Oracle Java SAAJ implementation, edit the WebLogic start up script for the domain where OpenAM runs, such as `/path/to/weblogic/user_projects/domains/wlsdomain/bin/startWebLogic.sh`. Change the following line:

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

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

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

Restart WebLogic for the change to take effect.

1.13 Preparing IBM WebSphere

Before you deploy OpenAM, use the Administrator console to update JVM options as described in [Section 1.2, “Preparing a Java Environment”](#).

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

1. In WebSphere administration console, browse to `Application > Application Type > WebSphere enterprise applications > OpenAM Name > Class loading and update detection`.

2. Set Class loader order > Classes loaded with local class loader first (parent last).
3. Set WAR class loader policy > Single class loader for application.
4. Save your work.

Furthermore when using IBM Java, add the JAXP Reference Implementation .jar into the OpenAM .war file before deploying the .war into WebSphere as this required library is missing otherwise.

1. Unpack the OpenAM .war file.

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

2. Add the JAXP Reference Implementation .jar in WEB-INF/lib/.

```
$ wget http://repo1.maven.org/maven2/com/sun/xml/parsers/jaxp-ri/1.4.5/jaxp-ri-1.4.5.jar  
$ mv jaxp-ri-1.4.5.jar WEB-INF/lib/
```

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

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

4. Deploy the new .war file.

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

Chapter 2

Installing OpenAM Core Services

This chapter covers tasks required for a full install of OpenAM server with or without OpenAM Console.

This chapter does not cover installation for enforcing policies on resource servers. To manage access to resources on other servers, you can use OpenIG or OpenAM policy agents.

[OpenIG](#) is a high-performance reverse proxy server with specialized session management and credential replay functionality. It can function as a standards-based policy enforcement point.

OpenAM policy agents provide policy enforcement on supported web servers and Java EE containers, and are tightly integrated with OpenAM. See the *OpenAM Web Policy Agent Installation Guide*, or *OpenAM Java EE Policy Agent Installation Guide* for instructions on installing OpenAM policy agents in supported web servers and Java EE application containers.

Table 2.1. Deciding How To Install OpenAM

If you want to...	Then see...
Install quickly for evaluation using default settings	Procedure 2.1, “To Deploy OpenAM” and Procedure 2.2, “To Configure OpenAM With Defaults”

If you want to...	Then see...
	Alternatively, follow the full example in the <i>Getting Started</i> guide.
Install OpenAM server and console, choosing settings	Procedure 2.1, "To Deploy OpenAM" and Procedure 2.4, "To Configure OpenAM"
Erase the configuration and start over	Procedure 2.3, "To Delete an OpenAM Configuration Before Redeploying"
Add an OpenAM server to a site	Procedure 2.1, "To Deploy OpenAM", and Procedure 2.5, "To Add a Server to a Site"
Install OpenAM server only (no console)	Table 2.2, "Determine Which War File to Deploy", Procedure 2.1, "To Deploy OpenAM", and Procedure 2.6, "To Deploy OpenAM Core Server (No Console)"
Install ssoadm for CLI configuration	<i>Installing OpenAM Tools</i> , or OpenAM ssoadm.jsp in the <i>Administration Guide</i>
Perform a command-line install	<i>To Set Up Configuration Tools</i>
Install OpenAM in your DMZ	<i>Installing OpenAM Distributed Authentication</i>
Skin OpenAM for your organization	<i>Customizing the OpenAM End User Pages</i>
Uninstall OpenAM	<i>Removing OpenAM Software</i>

Select the .war file based on the type of deployment you need, as defined in the following table.

Table 2.2. Determine Which War File to Deploy

If you want to...	Use...
Install an OpenAM server including OpenAM Console	
Install OpenAM server without OpenAM Console	
Install OpenAM distributed authentication UI	

Procedure 2.1. To Deploy OpenAM

The file contains OpenAM server with OpenAM Console. 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 /path/to/tomcat/webapps/openam.war
```

You change the file name to openam.war when deploying in Tomcat so that the deployment URI is /openam.

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

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

Configuration Options

Please select a configuration option.

Default Configuration

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

[Create Default Configuration](#)

Custom Configuration

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

[Create New Configuration](#)

Procedure 2.2. To Configure OpenAM With Defaults

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

The default configuration sets the cookie domain based on the fully qualified domain name of the system. For an FQDN `openam.example.com`, the cookie domain is set to `.example.com`.

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

1. In the initial configuration screen, click Create Default Configuration under Default Configuration.
2. Provide different passwords for the default OpenAM administrator, amadmin, and default Policy Agent users.

The screenshot shows a window titled "OpenAM Configurator" with a sub-header "Default Configuration Option". Below the header is a paragraph of text: "Use this option for a quick setup. Only the super user name and agent user name are required. All other configuration parameters are defaulted for you. The user and agent passwords must be different values." To the right of this text is a legend: "* Indicates required field".

There are two sections of form fields:

- Default User [amAdmin]**
 - * Password: [password field] OK
 - * Confirm Password: [password field]
- Default Policy Agent [UriAccessAgent]**
 - * Password: [password field] OK
 - * Confirm Password: [password field]

At the bottom of the dialog are two buttons: "Create Configuration" and "Cancel".

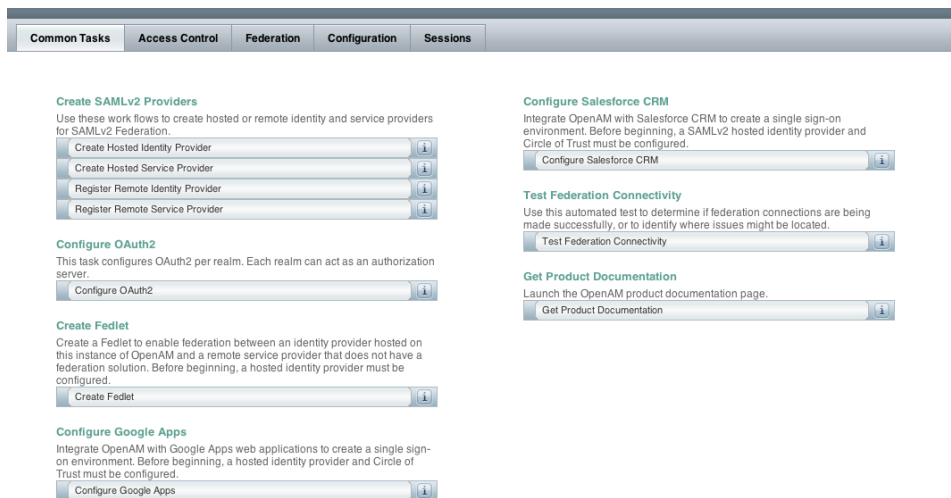
3. When the configuration completes, click Proceed to Login, and then login as the OpenAM administrator with the first of the two passwords you provided.

Sign in to OpenAM

The screenshot shows a login form with the following elements:

- User Name:** [amadmin]
- Password:** [password field]
- LOG IN** button

After successful login, OpenAM redirects you to OpenAM Console.



Procedure 2.3. To Delete an OpenAM Configuration Before Redeploying

If you are unhappy with your configuration and want to start over from the beginning, follow these steps.

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

The following example shuts down Tomcat for example.

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

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

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

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

If you use an external configuration store, also delete the entries under the configured OpenAM suffix (by default).

- Restart the OpenAM web application.

The following example starts the Tomcat container.

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

Procedure 2.4. To Configure OpenAM

- In the initial configuration screen, click Create New Configuration under Custom Configuration.
- Provide a password having at least 8 characters for the OpenAM Administrator, amadmin.

The screenshot shows the 'OpenAM Configurator' window with the 'Custom Configuration Option' dialog. The 'General' step is selected in the left sidebar. The main area displays 'Step 1: General' with instructions: 'Enter the password for the default user, amAdmin. The password must be at least 8 characters in length. If this configuration will be part of an existing deployment, the password you enter must match that of the original deployment.' Below this is a 'Default User Password' section with two input fields: '* Password' and '* Confirm Password'. The 'Password' field has a blue checkmark and 'OK' text next to it. The 'Confirm Password' field is empty. A legend indicates '* Indicates required field'. At the bottom, there are 'Previous', 'Next', and 'Cancel' buttons.

3. Make sure the server settings are valid for your configuration.

OpenAM Configurator

Custom Configuration Option

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

Step 2: Server Settings

Confirm the following settings to use for the server.

* Indicates required field

Server Settings

- * Server URL
- * Cookie Domain
- * Platform Locale
- * Configuration Directory

Previous Next Cancel

Server URL

Provide a valid URL to the base of your OpenAM web container, including a fully qualified domain name (FQDN).

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

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

Cookie Domain

Starts with a dot (.).

Platform Locale

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

Configuration Directory

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

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

The screenshot shows the 'OpenAM Configurator' window with the 'Custom Configuration Option' screen. The left sidebar lists steps: 1. General, 2. Server Settings, 3. Configuration Store (selected), 4. User Store, 5. Site Configuration, 6. Agent Information, and 7. Summary. The main content area is titled 'Step 3: Configuration Data Store Settings' and includes instructions: 'If no other OpenAM instance already exists in the environment, then choose First Instance. If one or more OpenAM instances already exist in the environment, choose Add to Existing Deployment.' There are two radio buttons: 'First Instance' (selected) and 'Add to Existing Deployment?'. A legend indicates that an asterisk (*) denotes a required field. Below this is the 'Configuration Store Details' section with the following fields: 'Configuration Data Store' (radio buttons for 'OpenAM' (selected) and 'OpenDJ or Oracle Directory Server Enterprise Edition'), 'SSL/TLS Enabled' (checkbox, not checked), 'Host Name' (text box with 'localhost'), 'Port' (text box with '50389'), 'Admin Port' (text box with '4444'), 'JMX Port' (text box with '1689'), 'Encryption Key' (text box with 'tn7xlCcolck4UnbDHIZJjhIHvu1UP9H'), and 'Root Suffix' (text box with 'dc=openam,dc=forgerock,dc=org'). At the bottom are 'Previous', 'Next', and 'Cancel' buttons.

You can also add this OpenAM installation to an existing deployment, providing the URL of the site. See [Procedure 2.5, “To Add a Server to a Site”](#) for details.

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

When you create a new OpenAM custom configuration that uses an external LDAP directory server for the configuration data store, you must use a root suffix DN with at least two domain components, such as `dc=example,dc=com`.

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

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

The screenshot shows the 'OpenAM Configurator' window with the 'Custom Configuration Option' tab selected. The left sidebar contains a navigation menu with seven items: 1. General, 2. Server Settings, 3. Configuration Store, 4. User Store (highlighted with a blue arrow), 5. Site Configuration, 6. Agent Information, and 7. Summary.

The main content area is titled 'Step 4: User Data Store Settings' and includes a help icon. Below the title is a paragraph of text: 'You can use the data store that comes with the OpenAM configuration data store, or you can use a different user data store. A good practice for setting up production environments is to use an external user data store, one that is different than the OpenAM user data store. Please note that Policy Service and LDAP Authentication Module shall be configured to use the Directory Administrator DN and Password provided here.'

There are two radio button options: 'OpenAM User Data Store' (unselected) and 'Other User Data Store' (selected). A red asterisk indicates a required field.

The 'User Store Details' section contains the following fields:

- * User Data Store Type: Radio buttons for Oracle Directory Server Enterprise Edition, Active Directory with Host and Port, Active Directory Application Mode, OpenDJ (selected), AD with Domain Name, and IBM Tivoli Directory Server.
- * SSL/TLS Enabled: A checkbox that is currently unchecked.
- * Directory Name: A text input field containing 'opendj.example.com'.
- * Port: A text input field containing '1389' with an 'OK' button to its right.
- * Root Suffix: A text input field containing 'dc=openam,dc=forgerock,dc=org'.
- * Login ID: A text input field containing 'cn=Directory Manager'.
- * Password: A masked text input field (dots) with an 'OK' button to its right.

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

User Data Store Type

If you have a directory service already provisioned 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 on which the directory listens for StartTLS or SSL connections. When using this option you also need to make sure the trust store used by the JVM running OpenAM has the necessary certificates installed.

Directory Name

FQDN for the host housing the directory service

Port

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

Root Suffix

Base distinguished name (DN) where user data are stored

Login ID

Directory administrator user DN. The administrator must be capable of updating schema and user data.

Password

Password for the directory administrator user

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

If you have a site configuration with a load balancer, you can enable session high availability persistence and failover. OpenAM then stores sessions across server restarts, so that users do not have to login again.

If you then add additional servers to this OpenAM site, OpenAM performs *session failover*, storing session data in a directory service that is shared by different OpenAM servers. The shared storage means that if an OpenAM server fails, other OpenAM servers in the site 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. If session failover is important for your deployment, also follow the instructions in [Setting Up OpenAM Session Failover](#).

OpenAM Configurator

Custom Configuration Option

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

Step 5: Site Configuration

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

No
 Yes

* Indicates required field

Site Configuration Details

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

* Site Name OK

* Load Balancer URL OK

Enable Session HA Persistence and Failover OK

Previous Next Cancel


It is possible to set up a site after initial installation and configuration. Doing so is described in the chapter on [Setting Up OpenAM Session Failover](#).

7. In the Agent Information screen, provide a password having at least 8 characters to be used by policy agents to connect to OpenAM.

OpenAM Configurator

Custom Configuration Option

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

Step 6: Default Policy Agent User 

These settings are used by OpenAM policy agents for retrieving policy agent properties.

* Indicates required field

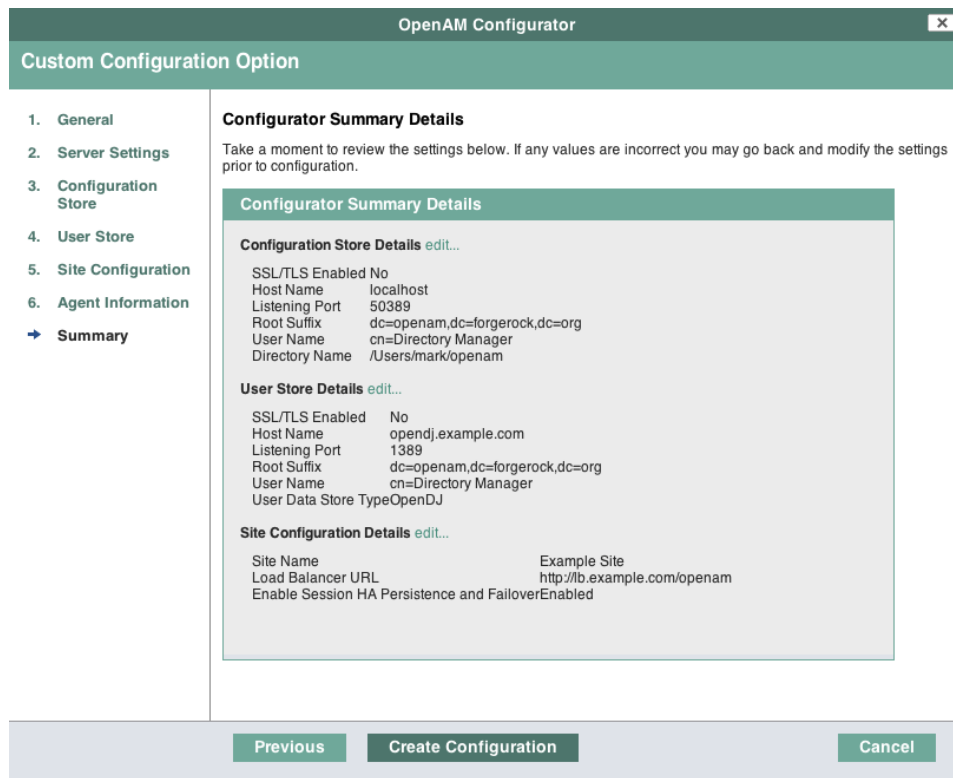
Policy Agent User

Default Policy Agent [UriAccessAgent]

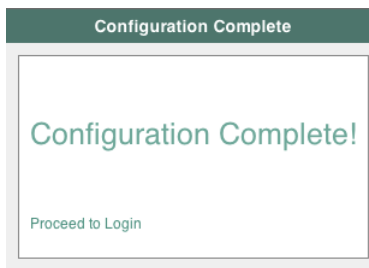
* Password OK

* Confirm Password

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



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



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

Sign in to OpenAM

User Name:

amadmin

Password:

.....

LOG IN

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

The screenshot shows the OpenAM Console interface. At the top, there is a navigation bar with tabs for 'Common Tasks', 'Access Control', 'Federation', 'Configuration', and 'Sessions'. The 'Configuration' tab is selected. Below the navigation bar, there are several sections of configuration tasks, each with a description and a button to perform the task:

- Create SAMLv2 Providers**: Use these work flows to create hosted or remote identity and service providers for SAMLv2 Federation. Tasks include 'Create Hosted Identity Provider', 'Create Hosted Service Provider', 'Register Remote Identity Provider', and 'Register Remote Service Provider'.
- Configure OAuth2**: This task configures OAuth2 per realm. Each realm can act as an authorization server. Task: 'Configure OAuth2'.
- Create Fedlet**: Create a Fedlet to enable federation between an identity provider hosted on this instance of OpenAM and a remote service provider that does not have a federation solution. Task: 'Create Fedlet'.
- Configure Google Apps**: Integrate OpenAM with Google Apps web applications to create a single sign-on environment. Task: 'Configure Google Apps'.
- Configure Salesforce CRM**: Integrate OpenAM with Salesforce CRM to create a single sign-on environment. Task: 'Configure Salesforce CRM'.
- Test Federation Connectivity**: Use this automated test to determine if federation connections are being made successfully. Task: 'Test Federation Connectivity'.
- Get Product Documentation**: Launch the OpenAM product documentation page. Task: 'Get Product Documentation'.

You can also access OpenAM Console by browsing to the Console URL, such as `http://openam.example.com:8080/openam/console`.

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

Procedure 2.5. To Add a Server to a Site

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

Follow these steps to configure a server to belong to an existing site.

1. In the initial configuration screen, under Custom Configuration click Create New Configuration.
2. In the first screen, enter the same password entered for the OpenAM Administrator, `amadmin`, when you configured the first server in the site.
3. Configure server settings as required.

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

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

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

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

5. In the site configuration screen, select Yes and enter the same site configuration details as you did for the first server in the site.

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

6. In the summary screen, verify the settings you chose, and then click Create Configuration.
7. When the configuration process finishes, click Proceed to Login, and then login as the OpenAM administrator to access OpenAM Console.

Procedure 2.6. To Deploy OpenAM Core Server (No Console)

You can deploy OpenAM server without OpenAM console by performing the following steps.

1. Deploy the file in your container.

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

```
$ cp /path/to/tomcat/webapps/coreonly.war
```

2. Browse to the configuration application, such as `http://openam.example.com:8080/coreonly/`, and configure OpenAM core services as in [Procedure 2.4, “To Configure OpenAM”](#).
3. After configuration, restrict permissions to the configuration directory, such as `$HOME/coreonly/` where `$HOME` corresponds to the user who runs the web container. Prevent other users from accessing files in the configuration directory.

Chapter 3

Installing OpenAM Tools

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

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

See [Procedure 3.1, “To Set Up Administration Tools”](#).

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

See [Procedure 3.2, “To Set Up Configuration Tools”](#).

Procedure 3.1. To Set Up Administration Tools

1. Make sure OpenAM is installed and running before proceeding.
2. Make sure the JAVA_HOME environment variable is properly set.

```
$ 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/
```

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 path to the directory where OpenAM configuration files are located, and where you want debug and log information to be located.

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

After setup, the tools are located under a directory named after the instance of OpenAM.

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

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

7. (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 "$@"
```

8. Check that **ssoadm** works properly.

```
$ echo password > /tmp/pwd.txt
$ chmod 400 /tmp/pwd.txt
$ cd /path/to/openam-tools/admin/openam/bin/
$ ./ssoadm list-servers -u amadmin -f /tmp/pwd.txt

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

The **ssoadm** commands can also be run from `ssoadm.jsp` in OpenAM, for example at `http://openam.example.com:8080/openam/ssoadm.jsp`, once the page has been enabled as described in the section on OpenAM `ssoadm.jsp` in the *Administration Guide*.

Not all of the sub-commands available through the **ssoadm** command are available on the `ssoadm.jsp` web page.

9. (Optional) If you connect to OpenAM over SSL (HTTPS), the **ssoadm** by default tries to trust the certificate based on the CA certificates in the Java cacerts truststore. This might not work for your deployment.

If the SSL certificate configured for the container where you deployed OpenAM was not signed by a recognized CA then the SSL connection process fails. For example, if you used a self-signed certificate as described in the *Administration Guide* procedure, *To Set Up OpenAM With HTTPS on Tomcat*, then the **ssoadm** command cannot trust that certificate by default. To allow the **ssoadm** command to trust the certificate, edit the **ssoadm (ssoadm.bat** on Windows) script as follows.

Add two additional options to the **java** command in the script to identify the proper trust store and trust store password, depending on how you set up SSL. The following example points to the key store in which Tomcat holds the self-signed certificate that it presents when establishing an HTTPS connection.

```
-D"javax.net.ssl.trustStore=/path/to/tomcat/conf/keystore.jks"
-D"javax.net.ssl.trustStorePassword=changeit"
```

If the **ssoadm** command cannot access the server key store in this way, set up your own trust store and import the server certificate using the Java **keytool** command.

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

To do this, set a `com.iplanet.am.naming.map.site.to.server` system property option of the **java** command 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 OpenAM server against which you set up the **ssoadm** command.

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

For example, if your site is behind `https://lb.example.com:443/openam`, and the OpenAM server against which you set up the **ssoadm** is at `http://openam.example.com:8080/openam`, then add the following property to the **java** command (all on one line without spaces).

```
-D"com.iplanet.am.naming.map.site.to.server=  
  https://lb.example.com:443/openam=http://openam.example.com:8080/openam"
```

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

Procedure 3.2. To Set Up Configuration Tools

1. Make sure the `JAVA_HOME` environment variable is properly set.

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

2. Unpack the tools from where you unzipped OpenAM.

```
$ cd /path/to/openam-tools/config  
$ unzip ~/Downloads/openam/  
Archive: ~/Downloads/openam/  
  inflating: README  
  inflating: sampleconfiguration  
  inflating: sampleupgrade  
  extracting:  
  extracting:  
  inflating: license.txt
```

Set up configuration files based on the `sampleconfiguration` example, and then apply the configuration to a deployed OpenAM `.war` file using the following command.

```
$ java -jar -f config.file
```

The `config.file` is set up by default to use the embedded data store with OpenAM installed on `server1.example.com`. You must edit the file before using it, as described in the *OpenAM Reference* for **configurator.jar**.

Chapter 4

Installing OpenAM Distributed Authentication

You can minimize the exposure of OpenAM to the Internet. It is a relatively standard practice to protect an enterprise network with a pair of firewalls. Systems that require external access are placed between the firewalls in a semi-secure area known as a demilitarized zone (DMZ). You can deploy a small subset of OpenAM as the login interface in a DMZ. That subset is known as the distributed authentication service (DAS). Logins through the DAS are forwarded through the internal firewall to the OpenAM core server. For more information see the OpenAM Administration Guide section on *Protecting Network Access*.

To deploy the DAS securely, select a system in your DMZ. Then take the following general steps:

1. Make sure the cookie domain for the DAS is configured in OpenAM under Configuration > System > Platform.
2. Make sure the realms used have a Realm/DNS alias for the DAS configured in OpenAM under Access Control > *Realm Name* > General.
3. Deploy the file into your web application container.

How you deploy the DAS .war file depends on your web application container. The procedure in this section shows how to deploy on Apache Tomcat.

4. Configure the DAS UI to access OpenAM core services.

-
5. Configure your firewall to allow end user access to the DAS.

Firewall configuration is not described here.



Important

The DAS relies on the classic OpenAM UI. If you customize the end user pages, follow the procedures for the classic UI described in [Customizing the OpenAM End User Pages](#).

Procedure 4.1. To Deploy the DAS on Tomcat

1. Copy the file into the webapps/ directory.

```
cp ~/Downloads/openam/  
/path/to/tomcat/webapps
```

2. Check that you see the initial DAS configuration screen in your browser.

Procedure 4.2. To Configure the DAS

1. Configure the DAS using the agent profile to connect to OpenAM.

Configuring DistAuth Application

Please provide the OpenAM Server Information.

Server Protocol:	<input type="text" value="http"/>
Server Host:	<input type="text" value="openam.example.com"/>
Server Port:	<input type="text" value="8080"/>
Server Deployment URI:	<input type="text" value="/openam"/>
DistAuth Server Protocol:	<input type="text" value="http"/>
DistAuth Server Host:	<input type="text" value="openam.example.com"/>
DistAuth Server Port:	<input type="text" value="8080"/>
DistAuth Server Deployment URI:	<input type="text" value="/das"/>
DistAuth Cookie Name:	<input type="text" value="AMDistAuthCookie"/>
OpenAM LB Cookie Name:	<input type="text" value="amlbcookie"/>
DistAuth LB Cookie Name:	<input type="text" value="DistAuthLBCookieName"/>
DistAuth LB Cookie Value:	<input type="text" value="DistAuthLBCookieValue"/>
Debug directory	<input type="text" value="/home/openam/das/debug"/>
Debug level	<input type="text" value="error"/>
Encryption Key	<input type="text" value="+w9w1grQXph3536Aqa4ZiZ4vNFdOrY5M"/>
Application user name	<input type="text" value="UrlAccessAgent"/>
Application user password	<input type="password" value="*****"/>
Confirm Application user password	<input type="password" value="*****"/>

Most DAS configuration choices require no clarification. Hints for equivocal parameters follow.

Debug Level

Default is error. Other options include error, warning, message, and off.

Encryption Key

Do not change the password encryption key.

Application User Name

The DAS uses this identity, such as UrlAccessAgent, to authenticate to OpenAM.

Application User Password

The DAS uses this password to authenticate to OpenAM.

2. Login through the DAS to access OpenAM services.

For testing, you can login as user demo, password changeit.

demo

First Name:	<input type="text"/>
* Last Name:	<input type="text" value="demo"/>
* Full Name:	<input type="text" value="demo"/>
Password:	<input type="password"/> Edit
Email Address:	<input type="text"/>
Telephone Number:	<input type="text"/>
Home Address:	<input type="text"/>
Password Reset Options:	Edit
Universal ID:	id=demo,ou=user,dc=openam,dc=forgerock,dc=org

When the `/openam/idm/EndUser` page is inside the firewall, and therefore not visible to users outside, redirect the browser after successful login to a page that exists. One way to do this is to use the `goto` parameter in the URL.

```
https://das.example.com/das/UI/Login?goto=absolute-successful-redirect-URL
```

On successful login, your browser stores an `AMDistAuthConfig` cookie that identifies the DAS.

3. Restrict permissions to the configuration for the DAS under the `$HOME/FAMDistAuth` directory of the user who runs the web container where you deployed the service.

The configuration file name ends in `AMDistAuthConfig.properties`.

If you deploy multiple DAS servers, you can configure them to forward requests to each other based on the `AMDistAuthConfig` cookie by setting the `com.sun.identity.distauth.cluster` property in this file. The following example lines are wrapped to fit on the page, but you put the entire property on a single line in the configuration file.

```
com.sun.identity.distauth.cluster=  
http://das.example.com:8080/das/UI/Login,  
http://das2.example.com:8080/das/UI/Login
```

-
4. If your deployment includes multiple OpenAM servers, then edit the DAS configuration file, `$HOME/FAMDistAuth/*AmDistAuthConfig.properties`, to include `X-Forwarded-For` in the list of `openam.retained.http.request.headers`.

Example: `openam.retained.http.request.headers=X-DSAMEVersion,X-Forwarded-For`

This ensures the authoritative OpenAM authentication server gets the client IP address in this header of the forwarded HTTP request. You can also add the header to the list for the `openam.retained.http.headers` property to have OpenAM copy the header to the response.

5. Some application servers such as JBoss 7 mount the content of the deployed `.war` file in a temporary location that can change on restart. To make sure that the DAS can find its bootstrap configuration in this case, specify the path to the bootstrap configuration file as a Java runtime option for the DAS, as in the following example. The property to set is `openam.das.bootstrap.file`.

```
-Dopenam.das.bootstrap.file=/home/openam/FAMDistAuth/AMDistAuthConfig.properties
```

You must make sure that the value of the option corresponds to the path to the correct `AMDistAuthConfig.properties` file.

Chapter 5

Customizing the OpenAM End User Pages

When you deploy OpenAM to protect your web-based applications, users can be redirected to OpenAM pages for login and logout. ForgeRock provides pages localized for English, French, German, Spanish, Japanese, Korean, Simplified Chinese, and Traditional Chinese, but you might require additional language support for your organization.

Also, by default the end user pages have ForgeRock styling and branding. You likely want to change at least the images to reflect your organization. You might want to have different page customizations for different realms as well. This chapter address how to get started customizing OpenAM end user pages for your organizations and supported locales.



Note

There is an evolving alternative UI available for OpenAM, known informally as the XUI. You can enable XUI in OpenAM Console under Configuration > Authentication > Core > Global Attributes, by selecting XUI Interface Enabled and saving your work. See [Section 5.3, “Configuring the XUI”](#) for more.

To customize the classic UI, first you copy the pages to customize to the proper location, and then you customize the files themselves.



Note

Case mismatch can cause failures in the UI lookup for some systems. To ensure lookup success and for consistency, use lowercase names for your customized directories. All of the default directories are already lowercase.

Classic UI images described in this chapter are located in `/path/to/tomcat/webapps/openam/images/`, and CSS in `/path/to/tomcat/webapps/openam/css/`. If you choose to modify images for your deployment, you can maintain the sizes to avoid having to customize page layout extensively.

5.1 Updating the Classic UI

When developing with a web container that deploys OpenAM in a temporary location, such as JBoss or Jetty, restarting the container can overwrite your changes with the deployable `.war` content. For those web containers, you should also prepare a deployable `.war` containing your changes, and redeploy that file to check your work.



Tip

For production deployment you must package your changes in a custom OpenAM deployable `.war` file. To create a deployable `.war`, unpack the OpenAM `.war` file from `~/Downloads/openam` into a staging directory, apply your changes in the staging directory, and use the `jar` command to prepare the deployable `.war`.

The procedures below describe how to update a deployed version of OpenAM, so that you can see your changes without redeploying the application. This approach works for development as long as your web container does not overwrite changes.

- [Procedure 5.1, “To Copy the Pages to Customize For the Top-Level Realm”](#)
- [Procedure 5.2, “To Copy the Pages to Customize For Another Realm”](#)

- [Procedure 5.3, “To Customize Files You Copied”](#)

Procedure 5.1. To Copy the Pages to Customize For the Top-Level Realm

Rather than changing the default pages, customize your own copy.

1. Change to the `config/auth` directory where you deployed OpenAM.

```
$ cd /path/to/tomcat/webapps/openam/config/auth
```

2. Copy the default files and optionally the localized files to `suffix[_locale]/html`, where `suffix` is the value of the RDN of the configuration suffix, such as `openam` if you use the default configuration suffix, and the optional `locale` is, for example, `jp` for Japanese, or `zh_CN` for Simplified Chinese.

The following example copies the files for the Top-Level Realm (`/`) for a custom French locale.

```
$ mkdir -p openam/html
$ cp -r default/* openam/html
$ mkdir -p openam_fr/html
$ cp -r default_fr/* openam_fr/html
```

See [How OpenAM Looks Up UI files](#) for details.

Procedure 5.2. To Copy the Pages to Customize For Another Realm

As for the top-level realm, customize your own copy rather than the default pages.

1. Change to the `config/auth` directory where you deployed OpenAM.

```
$ cd /path/to/tomcat/webapps/openam/config/auth
```

2. Depending on which locale you want to customize, copy the default files and optionally the localized files to `suffix[_locale]/services/realm/html`, where `suffix` is the value of the RDN of the configuration suffix, which is `openam` if you use the default configuration suffix.

The following example copies the files for a custom French locale and a realm named `ventes`.

```
$ mkdir -p openam/html/ventes/html
$ cp -r default/* openam/services/ventes/html
$ mkdir -p openam_fr/services/ventes/html
$ cp -r default_fr/* openam_fr/services/ventes/html
```

Procedure 5.3. To Customize Files You Copied

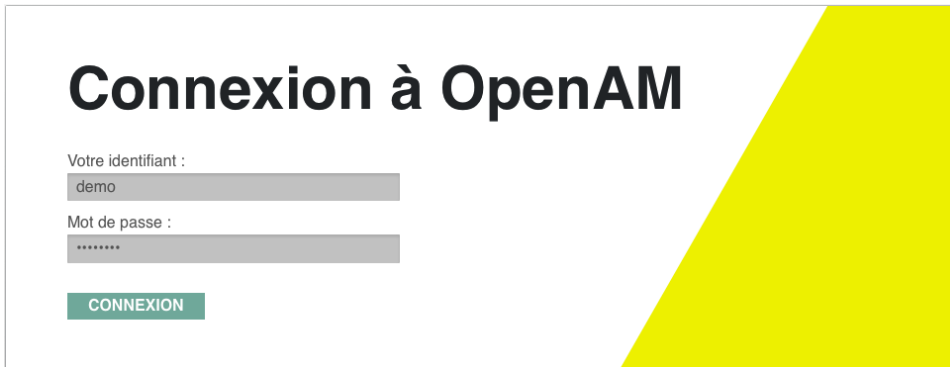
The .jsp files from the default/ directory reference the images used in the OpenAM pages, and retrieve localized text from the .xml files. Thus you customize appearance through the .jsp files, being careful not to change the functionality itself. You customize the localized text through the .xml files.

1. Modify appearance if you must by editing the .jsp, image, and CSS files without changing any of the JSP tags that govern how the pages work.
2. Modify the localized text, using UTF-8 without escaped characters, by changing only the original text strings in the .xml files.

For example, to change the text in the default OpenAM login screen in the top-level realm for the French locale, edit `openam_fr/html/DataStore.xml`.

3. If necessary, restart OpenAM or the web container to test the changes you have made.

The following screen shot shows a customized French login page where the string `Nom d'utilisateur` has been replaced with the string `Votre identifiant`, and the string `Mot de passe` has been replaced with the string `Votre mot de passe` in `openam_fr/html/DataStore.xml`.



5.2 How OpenAM Looks Up UI Files

This section provides a more complete description of how OpenAM looks up UI files.

OpenAM uses the following information to look up the UI files.

Configuration suffix RDN

When you set up the OpenAM to store its configuration in a directory server, you provide the distinguished name of the configuration suffix, by default , therefore, the relative distinguished name attribute value is `openam`.

Client (browser) locale language

The client can specify a locale, which can consist of both a language and a territory, such as `en_GB` for British English. The language in this case is `en`.

Client (browser) locale territory

If the client local is `en_GB`, then the territory in this case is `GB`.

Platform locale language

The platform locale, defined for the platform where OpenAM runs, can also consist of both a language and a territory, such as `hu_HU`. In this example the platform locale language is `hu` for Hungarian.

Platform locale territory

If the platform locale is `hu_HU`, the platform locale territory is `HU` for Hungary.

Realm

Realms can be nested. OpenAM uses the nesting as necessary to look for files specific to a sub-realm before looking in the parent realm.

For all realms below the top level realm, OpenAM adds a `services` directory before the realm to the search path.

Client name

Client names identify the type of client. The default, `html`, is the only client name used unless client detection mode is enabled. When client detection mode is enabled, the client name can be different for mobile clients, for example.

File name

File names are not themselves localized. Thus `Login.jsp` has the same name for all locales, for example.

OpenAM tries first to find the most specific file for the realm and local requested, gradually falling back on less specific alternatives, then on other locales. The first and most specific location as follows.

suffix_client-locale-language_client-locale-territory/services/realm/client-name/file-name

Example 5.1. UI File Lookup

OpenAM looks up `Login.jsp` in the following order for a realm named `realm`, with the browser requesting `en_GB` locale, the platform locale being `hu_HU`, and the configuration suffix named `o=openam`. The client name used in this example is the generic client name `html`.

```
openam_en_GB/services/realm/html/Login.jsp
openam_en_GB/services/realm/Login.jsp
openam_en_GB/services/html/Login.jsp
openam_en_GB/services/Login.jsp
openam_en_GB/html/Login.jsp
openam_en_GB/Login.jsp
openam_en/services/realm/html/Login.jsp
openam_en/services/realm/Login.jsp
openam_en/services/html/Login.jsp
openam_en/services/Login.jsp
openam_en/html/Login.jsp
openam_en/Login.jsp
openam_hu_HU/services/realm/html/Login.jsp
openam_hu_HU/services/realm/Login.jsp
openam_hu_HU/services/html/Login.jsp
openam_hu_HU/services/Login.jsp
openam_hu_HU/html/Login.jsp
openam_hu_HU/Login.jsp
openam_hu/services/realm/html/Login.jsp
openam_hu/services/realm/Login.jsp
openam_hu/services/html/Login.jsp
openam_hu/services/Login.jsp
openam_hu/html/Login.jsp
openam_hu/Login.jsp
openam/services/realm/html/Login.jsp
openam/services/realm/Login.jsp
openam/services/html/Login.jsp
openam/services/Login.jsp
openam/html/Login.jsp
openam/Login.jsp
default_en_GB/services/realm/html/Login.jsp
default_en_GB/services/realm/Login.jsp
default_en_GB/services/html/Login.jsp
default_en_GB/services/Login.jsp
default_en_GB/html/Login.jsp
default_en_GB/Login.jsp
default_en/services/realm/html/Login.jsp
default_en/services/realm/Login.jsp
default_en/services/html/Login.jsp
default_en/services/Login.jsp
default_en/html/Login.jsp
default_en/Login.jsp
default_hu_HU/services/realm/html/Login.jsp
default_hu_HU/services/realm/Login.jsp
default_hu_HU/services/html/Login.jsp
default_hu_HU/services/Login.jsp
default_hu_HU/html/Login.jsp
default_hu_HU/Login.jsp
default_hu/services/realm/html/Login.jsp
default_hu/services/realm/Login.jsp
```

```
default_hu/services/html/Login.jsp
default_hu/services/Login.jsp
default_hu/html/Login.jsp
default_hu/Login.jsp
default/services/realml/html/Login.jsp
default/services/realml/Login.jsp
default/services/html/Login.jsp
default/services/Login.jsp
default/html/Login.jsp
default/Login.jsp
```

5.3 Configuring the XUI

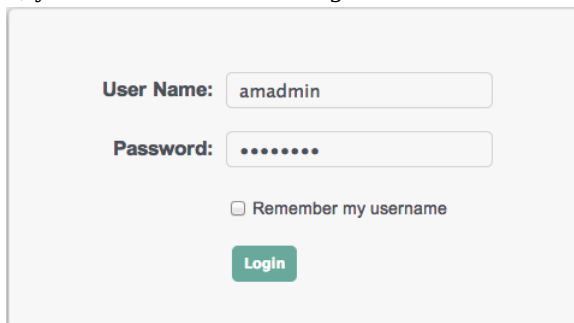
XUI is a new, still evolving UI for OpenAM, based on the Backbone.js JavaScript Model-View-Controller (MVC) framework, Handlebars.js for templating the "View" in the MVC framework, Underscore.js for the JavaScript-related utility functions, and a programmable LESS CSS, working with the OpenAM REST API.

Interface Stability: Internal (not supported)

XUI is not supported for production use.

The main XUI configuration file is based on LESS CSS; for more information, see [LESS, the Dynamic Stylesheet Language](#). If desired, you can incorporate additional LESS CSS features in the XUI, above and beyond what is shown in the themeConfig.json file described in this section.

If you want to test the XUI, the first step is to enable it on the login screen. To do so, login to the OpenAM console as the administrator, and browse to Configuration > Authentication > Core > XUI Interface > Enabled. The next time you start OpenAM, you will see the following screen:



The screenshot shows a login form with the following elements:

- User Name:** amadmin
- Password:** masked with seven dots
- Remember my username
- Login** button

The look and feel of this login screen and user profile page are defined by the main XUI configuration file, themeConfig.json. You can find this file in the /path/to/openam/webapps/XUI directory.

You can customize the settings in the `themeConfig.json` file. For more information on each parameter in this file, see the Reference Guide chapter on *XUI Configuration Parameters*.

If desired, you can create themes for different realms. This assumes that you have already configured a realm named `realm1`. For more information, see *Configuring Realms* in the OpenAM Administration Guide.

Now to create a theme for the second realm, open the `themeConfig.json` file in a text editor. Make a copy of all lines between the left curly bracket `{` after the `themes` parameter, and the corresponding right curly bracket `}` towards the end of the file.

```
{
  "themes": [
    {
      "name": "default",
      "path": "",
      "realms": [".*"],
      "regex": true,
      . . .
      "footer": {
        "mailto": "info@forgerock.com",
        "phone": "+47 2150108"
      }
    }
  ]
}
```

For a new realm named `realm1`, the revised `themeConfig.json` file should look similar to:

```
{
  "themes": [
    {
      "name": "default",
      "path": "",
      "realms": [".*"],
      "regex": true,
      . . .
      "footer": {
        "mailto": "info@forgerock.com",
        "phone": "+47 2150108"
      }
    },
    {
      "name": "realm1",
      "path": "path/to/realm1/",
      "realms": ["realm1.*"],
      "regex": true,

```



```
    . . .  
        "footer": {  
            "mailto": "info@example.com",  
            "phone": "+1 555 555 5555"  
        }  
    }  
]  
}
```

Be careful with the syntax. Do not forget the comma between realms. If in doubt about your JSON syntax, refer to a validation tool such as [The JSON Validator](#).

If you want to keep a parameter used in the default realm, just delete it from the later realm. Except for the following parameters, realm parameters inherit values from the default: name, path, realms, and regex.

When configuring new or revised parameters, keep the following tips in mind:

- The path to the directory with custom realm settings require a trailing forward slash /.
- Logos may require custom height and width parameters.
- Each of the lessVars parameters are based on variables defined in files in the /path/to/webapps/openam/XUI/css/user directory.
- After making changes, use available tools to make sure the file uses correct JSON syntax.
- Each realm after the default requires at least the name, path, realms, and regex parameters.

When testing different options, make sure to clear the browser cache on a regular basis. Otherwise, changes that you have made may not be shown in your browser.

Chapter 6

Configuring the Core Token Service (CTS)

The Core Token Service (CTS) provides persistent and highly available token storage for a several components within OpenAM, including sessions, as well as OAuth 2.0 and SAML 2.0 tokens. The CTS is set up in a generalized token storage format. That format is always used for OAuth 2.0 tokens. If so configured, it is also used to ensure persistence of session and SAML 2.0 tokens.

The CTS relies on OpenDJ to store and replicate tokens. No other directory service is supported for CTS. By default, the CTS uses the same embedded or external directory service as is configured for OpenAM's configuration data store.

CTS tokens may change frequently. Other data stored in an OpenDJ server is considerably more static. The relative performance tuning requirements are quite different. If your deployment is large, that may justify going beyond the default configuration. Nevertheless, it is easier to configure CTS if you can stick with the OpenDJ server embedded in an OpenAM installation.

If you use the OpenDJ service embedded within OpenAM, CTS schema is automatically included. You can choose, however, to manage CTS data in an external instance of OpenDJ.

If you choose to set up CTS in an external instance of OpenDJ, you will have to install OpenDJ separately. For more information, see the [OpenDJ Installation Guide](#).

Once you have installed OpenDJ on an external server, you can set up schema definitions, specify tokens in a valid LDAP format, configure indexes to allow OpenAM to retrieve tokens, and quite possibly Access Control Instructions (ACIs) to give an appropriate user Create, Read, Update, and Delete (CRUD) privileges. But first, you should configure basic parameters for the CTS token data store in the OpenAM console.

6.1 CTS Configuration Parameters

If you want to reconfigure an existing implementation of CTS, be prepared. Any reconfiguration will orphan any tokens that are currently stored. To keep this from happening, disable client access to OpenAM before making any changes. Any changes require a server restart to put them into effect.

To access the main CTS configuration page from the console, select Configuration > Servers and Sites > Default Server Settings > CTS. The options that appear in the screenshot that follows are detailed in the *Reference* document. You can set a root suffix for CTS tokens in either the configuration store or an external token store.

If you select Default Token Store, OpenAM will use the embedded configuration store for CTS tokens.



Note

If desired, you could make these changes from the command line with variations on the **ssoadm update-server-cfg** command, as described in the OpenAM Reference document.

Edit server-default [Save](#) [Reset](#) [Back to Servers and Sites](#)

⌵ CTS Token Store ⌵ External Store Configuration * Indicates required field

CTS Token Store

Default Token Store
 External Token Store

* Root Suffix:

[⌵ Back to top](#)

External Store Configuration

* SSL/TLS Enabled:

* Directory Name:

* Port:

* Login Id:

* Password:

* Max Connections:

* Heartbeat:

[⌵ Back to top](#)

Possible options have been entered in the figure. If the External Token Store is selected, entries are required in all text boxes. The options shown in the figure are:

- Root Suffix

ou=ctsData,dc=openam,dc=example,dc=com

When you configure a new OpenDJ suffix for the CTS, also consider creating a dedicated OpenDJ backend for the suffix. This allows you to manage CTS data separately from less volatile data.

- SSL/TLS Enabled

disabled

- Directory Name

opendj-cts.example.org

- Port

389

- Login Id

`uid=openam,ou=admins,dc=example,dc=com`

This is the DN of a user with administrative access to CTS data. The value here corresponds to the DN used in the examples in [Section 6.3, “CTS Access Control Instructions”](#). You can bypass access control by binding with a root DN such as `cn=Directory Manager`.

- Password

- Max Connections

20 (arbitrary number)

- Heartbeat

10 (default, in seconds)

Navigate to Configuration > Servers and Sites > Default Server Settings > CTS. Any options that you change under this tab are inherited as defaults by individual servers. To confirm, make a change, and then navigate to Configuration > Servers and Sites > [Server Name] > CTS.

6.2 Preparing an OpenDJ Directory Service for CTS

OpenAM stores volatile CTS token data in an instance of OpenDJ. To make that possible, OpenDJ needs the associated configuration store indexes, which allow OpenAM to search CTS token data in an efficient manner.

Different schema files are available in the OpenAM `WEB-INF/template/ldif/sfha` directory. If you install OpenAM with the embedded version of OpenDJ, the schema from the `cts-add-schema.ldif`, `cts-container.ldif`, and `cts-indicies.ldif` files are installed. If you upgrade to OpenAM from a previous version with embedded OpenDJ, the schema from the `99-cts-add-schema-backport.ldif` file is incorporated in your upgrade.

However, if you are configuring an external OpenDJ CTS server, you must add schema manually. You must also configure the indexes in the table shown below. To do so, you can use the **dsconfig** command depicted in the *OpenDJ Administration Guide* chapter on [Configuring a Standard Index](#).

After creating indexes for the external OpenDJ CTS server, rebuild the indexes with the **rebuild-index** command described in the *OpenDJ Administration Guide* chapter on [Rebuilding Indexes](#).

Table 6.1. CTS Data Store Indexes

Attribute	Indexes Required
coreTokenDate01	equality
coreTokenDate02	equality
coreTokenDate03	equality
coreTokenDate04	equality
coreTokenDate05	equality
coreTokenExpirationDate	ordering
coreTokenInteger01	equality
coreTokenInteger02	equality
coreTokenInteger03	equality
coreTokenInteger04	equality
coreTokenInteger05	equality
coreTokenInteger06	equality
coreTokenInteger07	equality
coreTokenInteger08	equality
coreTokenInteger09	equality
coreTokenInteger10	equality
coreTokenString01	equality
coreTokenString02	equality
coreTokenString03	equality
coreTokenString04	equality
coreTokenString05	equality

Attribute	Indexes Required
coreTokenString06	equality
coreTokenString07	equality
coreTokenString08	equality
coreTokenString09	equality
coreTokenString10	equality
coreTokenString11	equality
coreTokenString12	equality
coreTokenString13	equality
coreTokenString14	equality
coreTokenString15	equality
coreTokenUserId	equality

6.3 CTS Access Control Instructions

If you bind to the OpenDJ CTS server as a root DN user, such `cn=Directory Manager`, you can skip this section.

If you bind as a regular administrative user, you must give the user appropriate access to the CTS data. Give the regular administrative user access to add, delete, modify, read, and search CTS data, by setting access control instructions on the Root Suffix entry for CTS data. The user in examples shown here has DN `uid=openam,ou=admins,dc=example,dc=com`.

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


For detailed information on ACIs, with examples showing how you can use the **dsconfig**, as well as various **ldap*** commands to configure them, see the OpenDJ chapter on *Configuring Privileges & Access Control*.

6.4 CTS and OpenDJ Replication

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

The historical information needed to resolve the latest changes is periodically purged to avoid growing to unmanageable sizes. The age at which the information is purged is known as the replication-purge-delay.

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

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

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

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

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

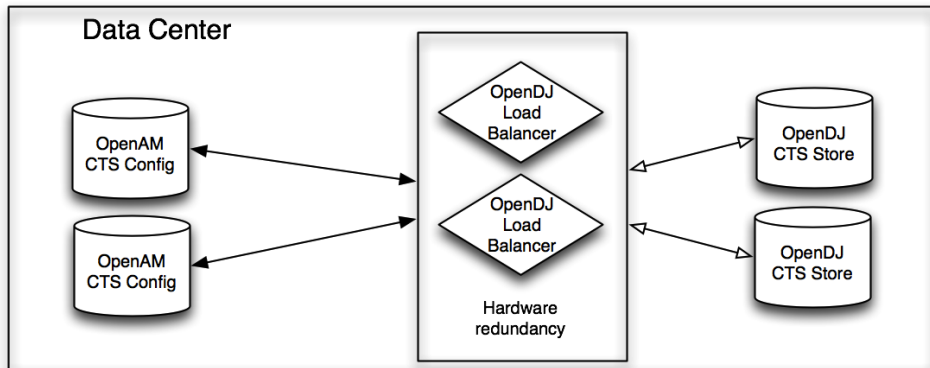
While you may choose to adjust the time periods associated with replication-purge-delay and backups, be sure that backups are performed more frequently. Otherwise, change log records that are required to restore data may be lost.

6.5 CTS Deployment Scenario

When properly configured, CTS can help your deployment avoid single points of failure (SPOF). Session and SAML 2.0 tokens which are normally stored only in the memory of a single server are also written to the CTS as a secondary token store. If the OpenAM instance that owns the session or SAML 2.0 token fails, a second instance of OpenAM can allow access to the session or token. To reduce the impact of any given failure, consider the following options:

- Start your implementation, if possible, with the CTS options available with the OpenDJ instance embedded in OpenAM. You can still set up a different backend on the embedded OpenDJ server. If the embedded OpenDJ server can handle your requirements, it will simplify implementation of CTS.
- Isolate the user, configuration, and session stores from OpenAM in separate external OpenDJ servers.
- Configure multiple directory stores for CTS, set up with load balancer(s).
- Add separate servers for data store replication. For more information on how this is done with OpenDJ, see the OpenDJ documentation on [Stand-alone Replication Servers](#).
- Set up redundancy in the load balancer connections between OpenAM and the external data store.

Deployment is easier if your requirements can be handled by the embedded instance of OpenDJ. But that may not be a viable for all situations. A relatively simplified method for configuring a more complex CTS deployment is depicted here:



For clarity, the diagram does not include options that may be appropriate for a production deployment such as firewalls and OpenAM agents. It also does not include options required for multiple data centers.

6.6 Managing CTS Tokens

There are five properties associated with token encryption, compression, and token cleanup frequency. The three that are associated with encryption and compression are disabled by default. The properties are as follows:

- `com.sun.identity.session.repository.enableEncryption`
Supports encryption of CTS tokens.
- `com.sun.identity.session.repository.enableCompression`
Enables GZip-based compression of CTS tokens.
- `com.sun.identity.session.repository.enableAttributeCompression`
Supports compression over and above the GZip-based compression of CTS tokens.
- `com.sun.identity.session.repository.cleanupRunPeriod`
Specifies a minimum CTS token lifetime. If there is no activity in the specified time period, the token is erased. Default: 300 seconds.
- `com.sun.identity.session.repository.healthCheckRunPeriod`

Sets a period of time when requests are sent to make sure the current instance of OpenAM is running. Default: 60 seconds.

To enable the encryption / compression options, navigate to Configuration > Servers and Sites > Default Server Settings > Advanced. In the Advanced Properties window, you should see these entries in the Property Name column with the corresponding value in the Property Value column. To enable them, change false to true in the Property Value column associated with the desired property, and click Save.



Note

If you are using SFO, or if you are using an external CTS directory, be consistent with these options. If you want to enable compression or encryption, you should enable all three on every instance of OpenAM within a deployment or replication group: `com.sun.identity.session.repository.enableEncryption`, `com.sun.identity.session.repository.enableCompression`, and `com.sun.identity.session.repository.enableAttributeCompression`.

6.7 General Recommendations for CTS Configuration

When configuring CTS, start with the OpenDJ server embedded with an installation of OpenAM. As it already has required CTS indexes included, that simplifies your tasks.

If you are deploying on a single site, and want CTS replication limited to that site, the default configuration store may be sufficient for your particular needs. If your needs go beyond a higher-level performance threshold, you may want to move the CTS token storage to one or more dedicated systems. Alternatively, if you need global replication of session, SAML 2.0, and OAuth 2.0 tokens, that would also justify a move to dedicated systems as it can help to have that extra level of control over how much replication is taking place.

CTS generally cause much more replication traffic than less volatile configuration data. Therefore, in high volume deployments you can move CTS data to dedicated, properly sized directory servers to improve performance. In addition, token compression as discussed in [Section 6.6, “Managing CTS Tokens”](#), is disabled by default. When enabled, token compression can reduce load requirements on the network connection between data stores.

While not recommended for high volume deployments, it is possible to use CTS in production within the default internal OpenDJ configuration store. That assumes a small scale deployment with a relatively simple topology.

The CTS is configured to work with a single OpenDJ directory server. That is a potential SPOF. To address that issue, set up a load balancer between OpenAM and the OpenDJ directory service used for the CTS. Redundant load balancers are preferred. If one instance of OpenDJ fails, the load balancer would redirect CTS requests to another instance of OpenDJ with a copy of the CTS tokens.

Once configured, the OpenDJ directory service replicates CTS data transmitted from OpenAM servers to connected OpenDJ servers. The amount of replication traffic can be significant, especially if replication proceeds over a WAN. You can limit this replication traffic by separating OpenDJ instances into directory and replication servers.

Chapter 7

Setting Up OpenAM Session Failover

This chapter covers setting up SFO when using multiple instances of OpenAM in a site configuration for high availability. The basic idea followed here is that you configure load balancing to be sticky, based on the value of an OpenAM cookie, `amlbcookie`, different for each OpenAM server. Should that server become unavailable, the load balancer fails client requests over to another OpenAM server. The other OpenAM server must then fail over the user session associated with the client.

SFO relies on a shared, highly available Core Token Service (CTS) to store user session data. The service is shared with other OpenAM servers in the same OpenAM site. When the OpenAM server where a user authenticated goes down, other servers in the site read user session information from the CTS, so the user with a valid session does not have to login again. When the original OpenAM server becomes available again, it can also read session information from the CTS, and can carry on serving users with active sessions.

This chapter includes these procedures.

- [Procedure 7.1, “To Configure a Site with a First OpenAM Server”](#)
- [Procedure 7.2, “To Configure Site Load Balancing”](#)
- [Procedure 7.3, “To Configure Session Failover After Installation”](#)

Procedure 7.1. To Configure a Site with a First OpenAM Server

Before you set up SFO, first configure OpenAM in a site configuration with a load balancer as the entry point to the site. The most expedient way to configure the site is to set it up during the initial OpenAM configuration. However, you may already have a working instance before realizing that multiple instances are necessary. The following steps walk you through setting up the site configuration for the first OpenAM server.

Once you have set up a site for the first OpenAM server, see [To Add a Server to a Site](#) for instructions on configuring subsequent servers in the site.

1. Login to OpenAM Console as `amadmin`, and then browse to Configuration > Servers and Sites > Sites.
2. Click New, and on the New Site page enter the site name, and set the Primary URL to the load balancer URL that is the entry point for the site, such as `https://lb.example.com/openam`.

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

3. Click Save to keep the site configuration.
4. Under Configuration > Servers and Sites > Server, click the link to the server configuration.
5. On the server configuration General tab page, set the Parent Site to the name of the site you just created, and then click Save to keep your changes.

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

Procedure 7.2. To Configure Site Load Balancing

If you did not set up the site during initial configuration, first follow the instructions in [Procedure 7.1, "To Configure a Site with a First OpenAM Server"](#), and then follow all the steps below.

1. For each OpenAM server in the site, select Configuration > Servers and Sites > Servers > *Server Name*, and then set Parent Site to the site you created before saving your work.
2. In an OpenAM site, the server that authenticated a user is the server that continues to manage that user's session, unless the server is no longer

available. Therefore, you should use sticky load balancing. To do so, configure your load balancer to inspect the value of the `amlbcookie` so that it can determine which OpenAM server should receive the client request.

As your load balancer depends on the `amlbcookie` value, on each OpenAM server console in the site, select Configuration > Servers and Sites > Servers > *Server Name* > Advanced, makes sure that `com.ipplanet.am.lbcookie.value` is unique. By default the value of the `amlbcookie` is set to the server ID for the OpenAM instance.



Note

When using SSL, the approach requires that you either terminate SSL on the load balancer and re-encrypt traffic from the load balancer to the OpenAM servers.

If you must change `amlbcookie` values to make them unique, then your changes take effect after you restart the OpenAM server. (To check, login to the console and check the cookie value in your browser.)

3. Restart each OpenAM server or the web containers where the OpenAM servers run so that all configuration changes take effect.

Procedure 7.3. To Configure Session Failover After Installation

Session failover requires a site configuration with one or more servers and OpenDJ as a configuration store (embedded or external).

If you did not configure session persistence and availability during initial configuration, first complete the steps in [Procedure 7.2, “To Configure Site Load Balancing”](#), and then follow these steps.

1. In the OpenAM console for one of the servers in the site, under Configuration > Global, click Session.
2. Under Secondary Configuration Instance, click New.

If the server is not part of a site, or if you are not using OpenDJ server, the New button is grayed out.

3. In the Add Sub Configuration page, check that the Name is set to the name of the site.
4. Activate the Session Persistence and High Availability Failover Enabled option.

5. Click Add to save your work.

Chapter 8

Removing OpenAM Software

This chapter shows you how to uninstall OpenAM core software. See the *OpenAM Web Policy Agent Installation Guide* or *OpenAM Java EE Policy Agent Installation Guide* for instructions on removing OpenAM agents.

Procedure 8.1. To Remove OpenAM Core Software

After you have deployed and configured OpenAM core services, you have at least two, perhaps three or four, locations where OpenAM files are stored on your system.

You remove the internal OpenAM configuration store when you follow the procedure below. If you used an external configuration store, you can remove OpenAM configuration data after removing all the software.

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

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

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

For a full install of OpenAM core services, configuration files include the following.

- The configuration directory, by default `$HOME/openam`. If you did not use the default configuration location, then check in the OpenAM console under Configuration > Servers and Sites > *Server Name* > General > System > Base installation directory.
- The hidden file that points to the configuration directory.

For example, if you are using Apache Tomcat as the web container, this file could be `$HOME/.openamcfg/AMConfig_path_to_tomcat_webapps_openam_` OR `$HOME/.openssocfg/AMConfig_path_to_tomcat_webapps_openam_`.

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

or

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



Note

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

If you used an external configuration store you must also remove the configuration manually from your external directory server. The default base DN for the OpenAM configuration is .

3. Undeploy the OpenAM web application.

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

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

Index

C

Core Token Service, 59
Custom end user pages, 49

D

Directory service requirements, 3, 3, 7
Downloading OpenAM, 9

I

Installing
 Behind the firewall, 43
 Full install, 21
 Interactive configuration, 26
 Load Balancer, 71
 No console, 21
 Session failover, 71
 Silent configuration, 40
 Starting over, 25
 Tools (ssoadm, etc.), 37

J

Java requirements, 2

P

Prerequisites, 1

U

Uninstalling, 75
