OpenAM Installation Guide

Version 12.0.0

Mark Craig
David Goldsmith
Gene Hirayama
Mike Jang
Chris Lee
Vanessa Richie

ForgeRock AS
33 New Montgomery St.,
Suite 1500
San Francisco, CA 94105, USA
+1 415-599-1100 (US)
www.forgerock.com

Abstract

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

Table of Contents

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 on GNU/Linux or Mac OS X. Where it is helpful to make a distinction between operating environments, examples for UNIX, GNU/Linux, Mac OS X, and so forth are labeled (UNIX). Mac OS X specific examples can be labeled (Mac OS X). Examples for Microsoft

Windows can be labeled (Windows). To avoid repetition, however, 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.

```
$ curl https://bjensen:hifalutin@opendj.example.com:8443/users/newuser
{
  "_rev" : "000000005b337348",
  "schemas" : [ "urn:scim:schemas:core:1.0" ],
  "contactInformation" : {
    "telephoneNumber" : "+1 408 555 1212",
    "emailAddress" : "newuser@example.com"
  },
  "_id" : "newuser",
  "name" : {
    "familyName" : "New",
    "givenName" : "User"
  },
  "userName" : "newuser@example.com",
  "displayName" : "New User",
  "meta" : {
    "created" : "2014-06-03T09:58:27Z"
  },
  "manager" : [ {
    "_id" : "kvaughan",
    "displayName" : "Kirsten Vaughan"
  } ]
}
```

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

ForgeRock core documentation, such as what you are now reading, 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 users' 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 validity with respect to the software, technical completeness with respect to the scope of the document, and usability for the expected audience.

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

Fully reviewed, published core documentation is available at http://docs.forgerock.org/. Use this documentation when working with a ForgeRock Enterprise release.

In-progress documentation can be found at each project site under the Developer Community projects page. Use this documentation when trying a nightly build.

The ForgeRock Community Wikis and provide additional, user-created information. We encourage you to join the community, so that you can update the Wikis, too.

4 Joining the ForgeRock Community

After you sign up to join the ForgeRock community, you can edit the Community Wikis, and also log bugs and feature requests in the issue tracker.

If you have a question regarding a project but cannot find an answer in the project documentation or Wiki, browse to the Developer Community page for the project, where you can find details on joining the project mailing lists, and find links to mailing list archives. You can also suggest updates to documentation through the ForgeRock docs mailing list.

The Community Wikis describe how to check out and build source code. Should you want to contribute a patch, test, or feature, or want to author part of the core documentation, first have a look on the ForgeRock site at how to get involved.

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`.

Important

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

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

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

1.2 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 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=admins,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 1.1. Identity Repository Indexes

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

1.5 Preparing an External Configuration Data Store

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

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 cannot be controlled when using the embedded store.

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.

OpenAM supports the Core Token Service (CTS), a centralized token repository for OpenAM session tokens, OpenID Connect ID tokens, and SAML 2.0 tokens, which can be stored in a locally embedded or external directory store together or separately from OpenAM configuration data. For more information, see the chapter on *Configuring the Core Token Service*.

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 sections covering directory schema and procedures on how to configure access to directory data.

Procedure 1.1. To Install an External OpenDJ Directory Server

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

1. Prepare your OpenDJ installation, then download the OpenDJ software. See the OpenDJ documentation about Installing OpenDJ From the Command Line.

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

   Example options are as follows:

Table 1.2. Example OpenDJ Setup Parameters

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

2. Change to the OpenDJ directory.

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

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

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

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

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

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

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

```
dn: dc=example,dc=com
objectclass: top
objectclass: domain
dc: example
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");)

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

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

5. Add the initial entries LDIF file using the `ldapmodify` command.

```
$ bin/ldapmodify \
  --port 4444 \
  --bindDN "cn=Directory Manager" \
  --bindPassword pwd \
  —-defaultAdd \
  --useSSL \
  --trustAll \
  —-filename add-config-entries.ldif
```

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

   Note

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

   If you are having difficulty successfully adding the `global-aci` attribute, try doing so without any line breaks.

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

7. At this point, install the OpenAM server if you haven't done so already. For details, see *Installing OpenAM Core Services*.

8. 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.

   Copy the schema files, located at `/path/to/tomcat/webapps/openam/WEB-INF/template/ldif/sfha` to a local file, `cts-add-schema.ldif`.

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

9. Add the schema to the directory server.

```
$ bin/ldapmodify \
  --port 4444 \
  --bindDN "cn=Directory Manager" \
  --bindPassword pwd \
  --useSSL \
  --trustAll \
  --fileName cts-add-schema.ldif
```

10. OpenAM uses certain attributes to search for configuration data. You must set some indexes on these attributes for the configuration store.

Table 1.3. Configuration Data Store Indexes

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

On the OpenDJ directory server, use `dsconfig` to add these indexes to your external configuration store. Repeat for each attribute in Table 1.3, “Configuration Data Store Indexes”.

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

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

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

12. Verify the indexes.

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

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

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/OpenAM-12.0.0.zip
```

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

`ClientSDK-12.0.0.jar`
: The OpenAM Java client SDK library

`ExampleClientSDK-CLI-12.0.0.zip`
: The .zip file containing the Java client SDK command-line examples, and .jar files needed to run the examples

`ExampleClientSDK-WAR-12.0.0.war`
: The .war file containing Java client SDK examples in a web application

`IDPDiscovery-12.0.0.war`
: 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*.

`Fedlet-12.0.0.zip`
: 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

`OpenAM-12.0.0.war`
: The deployable .war file

`OpenAM-DistAuth-12.0.0.war`
: The deployable .war file for distributed authentication

`OpenAM-ServerOnly-12.0.0.war`
: The deployable .war file when you want to deploy OpenAM server without the OpenAM console

`SSOAdminTools-12.0.0.zip`
: The .zip file that contains tools to manage OpenAM from the command line

`SSOConfiguratorTools-12.0.0.zip`
: The .zip file that contains tools to configure OpenAM from the command line

1.7 Enabling CORS Support

Cross-origin resource sharing (CORS) allows requests to be made across domains from user agents. OpenAM supports CORS, but CORS is not configured out of the box.

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

1. Unpack the OpenAM .war file.

```
$ mkdir /tmp/openam
$ cd /tmp/openam/
$ jar -xf ~/Downloads/openam/OpenAM-12.0.0.war
```

2. Edit the deployment descriptor file, `WEB-INF/web.xml`, to add a CORS filter configuration.

   First, add a `<filter-mapping>` element to name the filter and to indicate the URL pattern for the filter. The URL pattern matches the endpoints for which to support CORS. The following example adds CORS support for all OpenAM endpoints.

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

   Next, add a `<filter>` element to configure the filter. The following excerpt describes and demonstrates all of the required and optional configuration parameters.

```
<filter>
    <filter-name>CORSFilter</filter-name>
    <filter-class>org.forgerock.openam.cors.CORSFilter</filter-class>
    <init-param>
        <description>
            Accepted Methods - (Required) -
            A list of HTTP methods for which to accept CORS requests
        </description>
        <param-name>methods</param-name>
        <param-value>POST,PUT</param-value>
    </init-param>
    <init-param>
        <description>
            Accepted Origins - (Required) -
            A list of origins from which to accept CORS requests
        </description>
        <param-name>origins</param-name>
        <param-value>www.example.net,example.org</param-value>
    </init-param>
    <init-param>
        <description>
            Allow Credentials - (Optional) -
            Whether to include the allow Vary (Origin)
            and Access-Control-Allow-Credentials headers
            in the response [default false]
        </description>
        <param-name>allowCredentials</param-name>
        <param-value>true</param-value>
    </init-param>
    <init-param>
```

```
        <description>
            Allowed Headers - (Optional) -
            A list of HTTP headers which if included in the request
            DO NOT make it abort
        </description>
        <param-name>headers</param-name>
        <param-value>headerOne,headerTwo,headerThree</param-value>
    </init-param>
    <init-param>
        <description>
            Expected Hostname - (Optional) -
            The name of the host expected in the request Host header
        </description>
        <param-name>expectedHostname</param-name>
        <param-value>http://openam.example.com</param-value>
    </init-param>
    <init-param>
        <description>
            Exposed Headers - (Optional) -
            The list of headers which the user-agent can expose
            to its CORS client
        </description>
        <param-name>exposeHeaders</param-name>
        <param-value>exposeHeaderOne,exposeHeaderTwo</param-value>
    </init-param>
    <init-param>
        <description>
            Maximum Cache Age - (Optional) -
            The maximum time that the CORS client can cache
            the pre-flight response, in seconds [default 600]
        </description>
        <param-name>maxAge</param-name>
        <param-value>600</param-value>
    </init-param>
</filter>
```

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

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

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

4. Deploy the new .war file.

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

1.8 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.

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.

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

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

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

The following example script, `/etc/init.d/tomcat`, manages the service at system startup and shutdown. This script assumes you run OpenAM as the user `openam`.

```
#!/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
```

1.8.1 Tuning Apache Multi-Processing Modules

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

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

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

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

Important

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

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

1.9 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 1.2. 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 1.6, “To Prepare OpenAM for JBoss”.

Procedure 1.3. 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 1.2, “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/OpenAM-12.0.0.war
    ```

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

```
<?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 1.6, “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 1.4. 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 1.2, “To Prepare JBoss AS 7.1.0 / 7.1.1”.

2. However, you still need to review Procedure 1.5, “To Prepare JBoss for OpenAM” and Procedure 1.6, “To Prepare OpenAM for JBoss” to make sure the JVM and directories are configured appropriately.

Procedure 1.5. 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 1.6. 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/OpenAM-12.0.0.war
```

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

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

configuration.dir=$HOME/openamJboss
```

3. Rebuild the `openam.war` file.

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

1.10 Preparing Oracle WebLogic

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

Next, 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 the OpenAM OAuth 2.0 providers receive basic authentication credentials for OAuth 2.0 grants that rely on basic authentication.

1. Stop the WebLogic server.

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

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

Weblogic uses its own classes if a class exists in both the parent and child classloaders by default. To use OpenAM and its classes on WebLogic 11g, create

a WebLogic deployment descriptor file `weblogic.xml` and place it in the `/WEB-INF` directory. The descriptor file maps resources defined for OpenAM.

1. Create a WebLogic descriptor file, `/WEB-INF/weblogic.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
  <weblogic-web-app xmlns="http://xmlns.oracle.com/weblogic/weblogic-web-app"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://xmlns.oracle.com/weblogic/weblogic-web-app
     http://xmlns.oracle.com/weblogic/weblogic-web-app/1.3/weblogic-web-app.xsd">
        <context-root>/openam</context-root>
        <container-descriptor>
           <prefer-web-inf-classes>true</prefer-web-inf-classes>
        </container-descriptor>
  </weblogic-web-app>
```

2. Start the 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/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 $*
```

When using WebLogic 12.1.1 with Java 6, if you plan to use the **ssoadm** command to configure OpenAM, then make the following change to the start up script, `startWebLogic.sh`, to avoid exceptions and incorrect results. Change the following line:

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

To this:

```
${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.

1.11 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/OpenAM-12.0.0.war
```

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

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” Alternatively, follow the full example in the *Getting Started* guide.

If you want to...	Then see...
Install OpenAM server and console, choosing settings	Procedure 2.1, “To Deploy OpenAM” and Procedure 2.4, “To Custom 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	*Installing OpenAM Tools*, or OpenAM ssoadm.jsp in the *Administration Guide*
Perform a command-line install	*To Set Up Configuration Tools*
Install OpenAM in your DMZ	*Installing OpenAM Distributed Authentication*
Skin OpenAM for your organization	*Customizing the OpenAM End User Pages*
Uninstall OpenAM	*Removing OpenAM Software*

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	`OpenAM-12.0.0.war`
Install OpenAM server without OpenAM Console	`OpenAM-ServerOnly-12.0.0.war`
Install OpenAM distributed authentication UI	`OpenAM-DistAuth-12.0.0.war`

Procedure 2.1. To Deploy OpenAM

The `OpenAM-12.0.0.war` 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 OpenAM-12.0.0.war /path/to/tomcat/webapps/openam.war
```

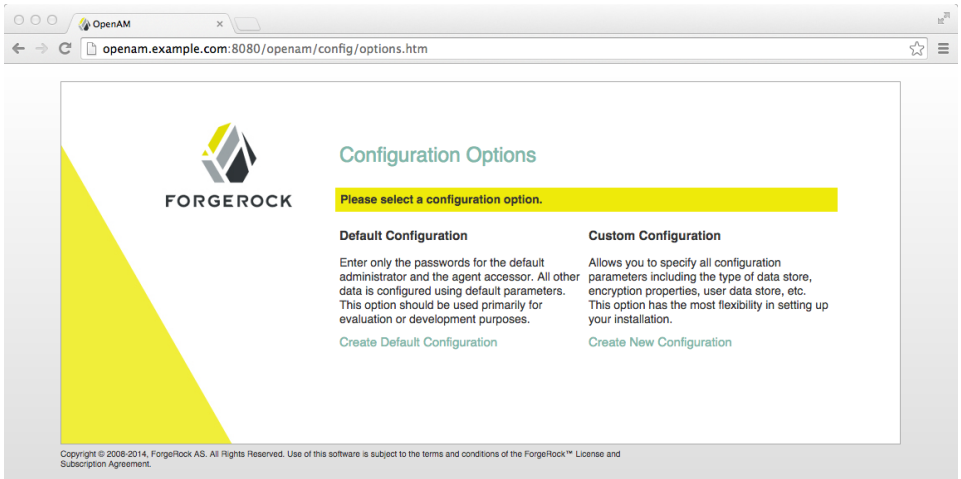
You change the file name to `openam.war` when deploying in Tomcat so that the deployment URI is `/openam`.

Note

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

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`.



Procedure 2.2. To Configure OpenAM With Defaults

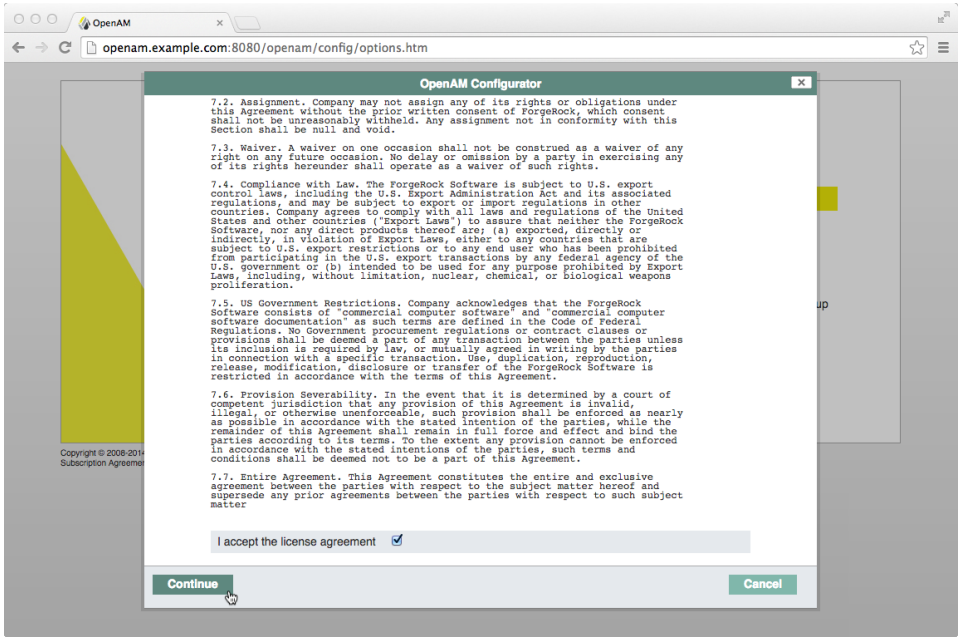
The default configuration option configures the embedded OpenDJ server using default ports—if the ports are already in use, OpenAM uses free ports—as both configuration store and identity store.

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`.

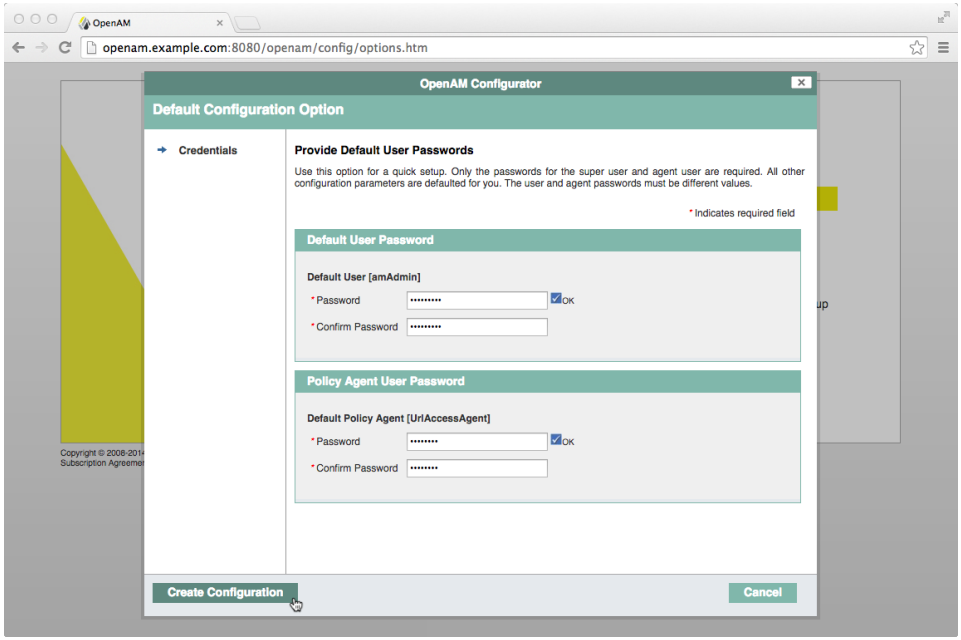
Configuration settings are saved to the home directory of the user running the web application container in a directory named after the deployment URI. In other words if OpenAM is deployed under `/openam`, then the configuration is saved under `$HOME/openam/`.

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

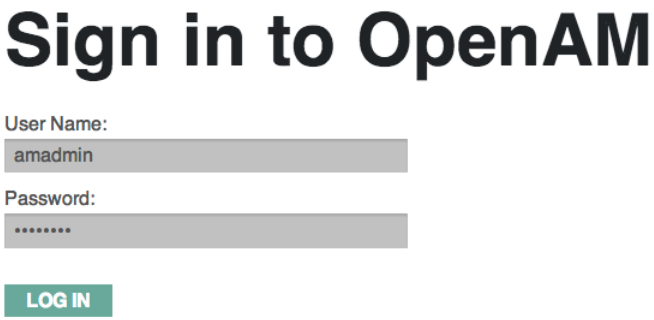
2. Review the software license agreement. If you agree to the license, click "I accept the license agreement", and then click Continue.



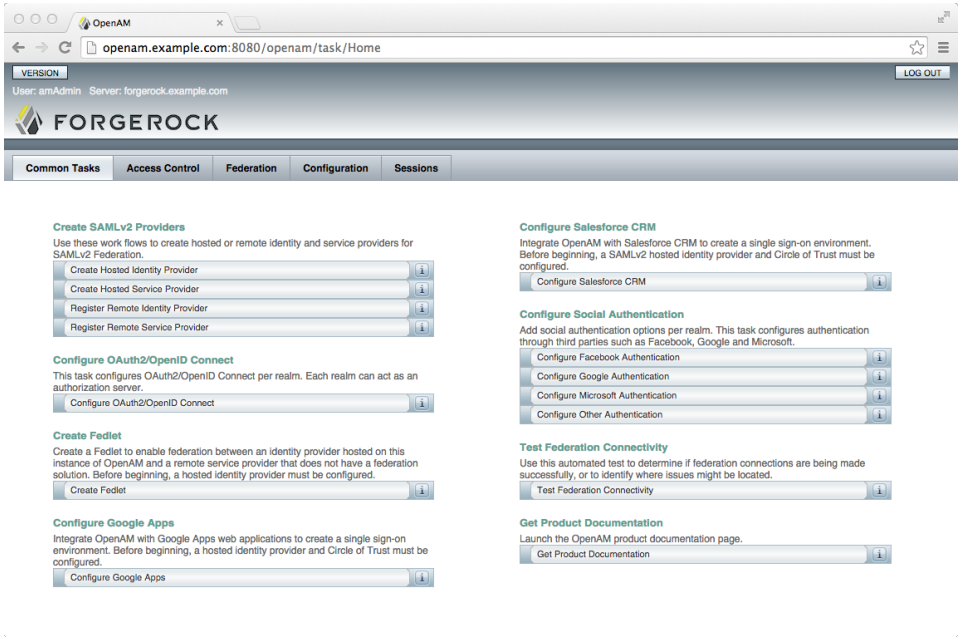
3. Provide different passwords for the default OpenAM administrator, `amadmin`, and default Policy Agent users.



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



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 dc=openam,dc=forgerock,dc=org).

3. Restart the OpenAM web application.

   The following example starts the Tomcat container.

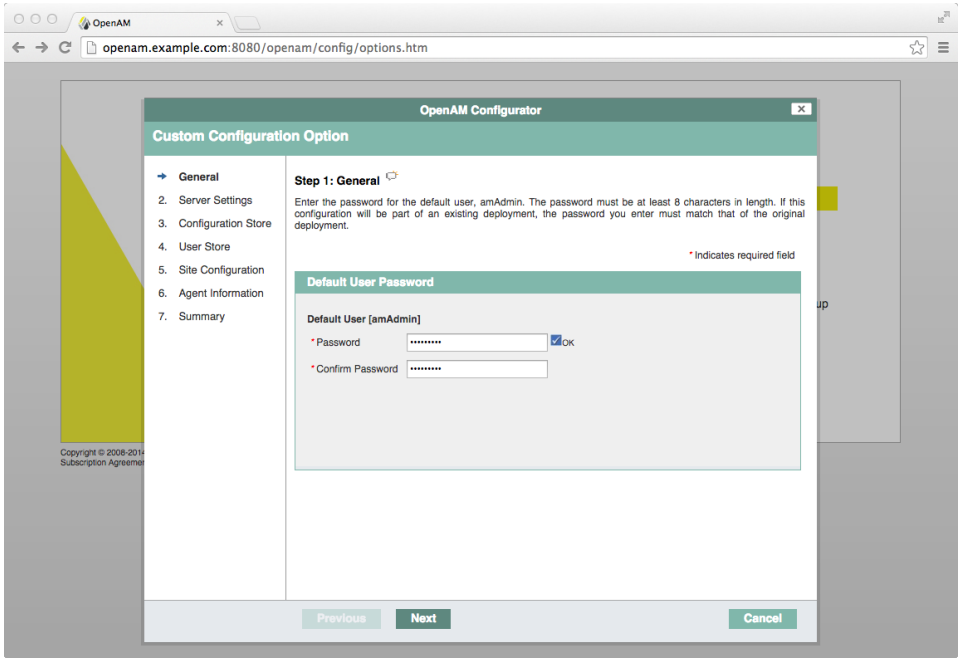
```
$ /path/to/tomcat/bin/startup.sh
Password:
Using CATALINA_BASE:   /path/to/tomcat
Using CATALINA_HOME:   /path/to/tomcat
Using CATALINA_TMPDIR: /path/to/tomcat/temp
Using JRE_HOME:        /path/to/jdk/jre
Using CLASSPATH:
       /path/to/tomcat/bin/bootstrap.jar:/path/to/tomcat/bin/tomcat-juli.jar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

Procedure 2.4. To Custom Configure OpenAM

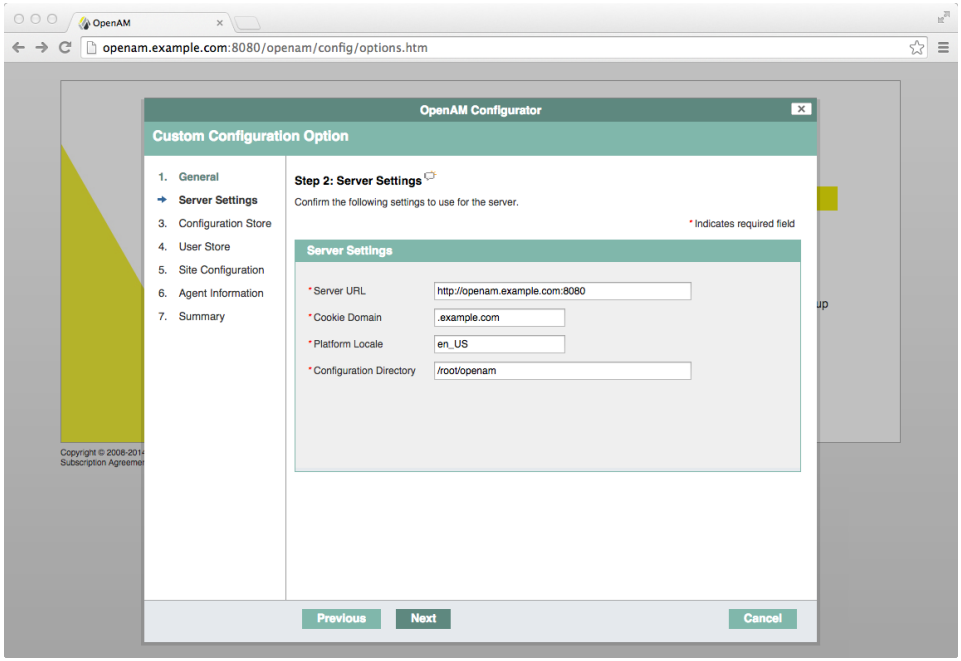
1. In the initial configuration screen, click Create New Configuration under Custom Configuration.

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

3. On the Default User Password page, provide a password having at least 8 characters for the OpenAM Administrator, `amadmin`.



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



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.

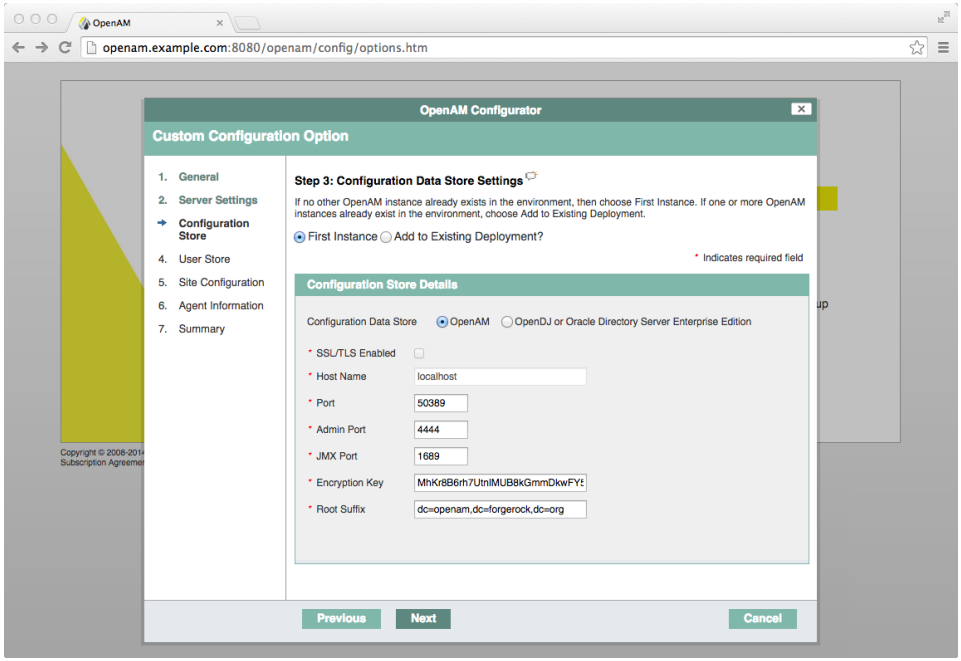
```
127.0.0.1 localhost.localdomain localhost
::1 localhost6.localdomain6 localhost6
127.0.1.1 openam openam.example.com
```

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.

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



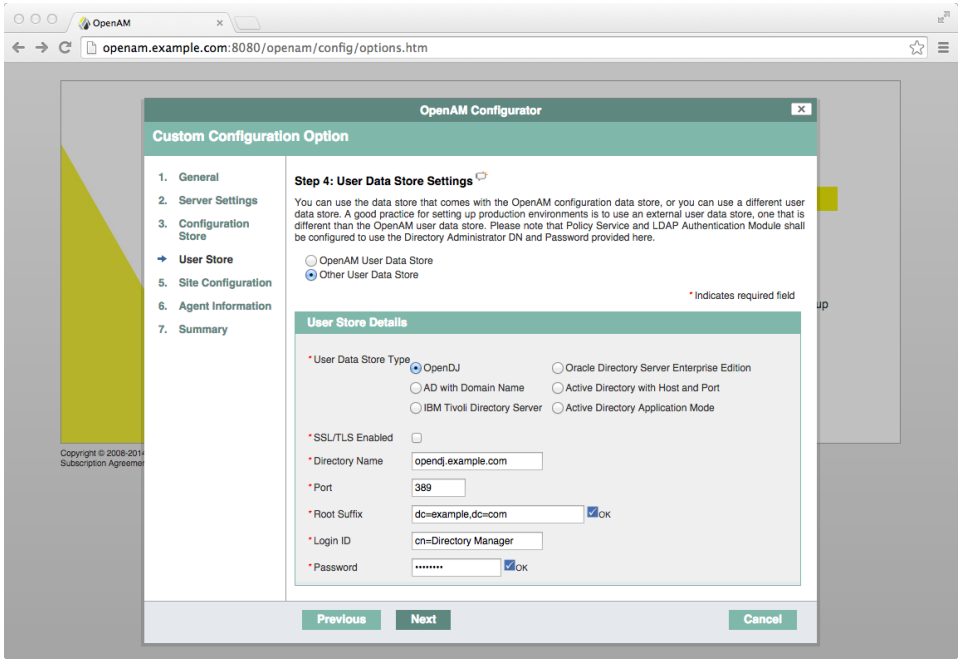
You can also add this OpenAM installation to an existing deployment, providing the URL of the site. See Procedure 2.5, “To Add a Server to a Site” for details.

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`.

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

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



User Data Store Type
: If you have 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

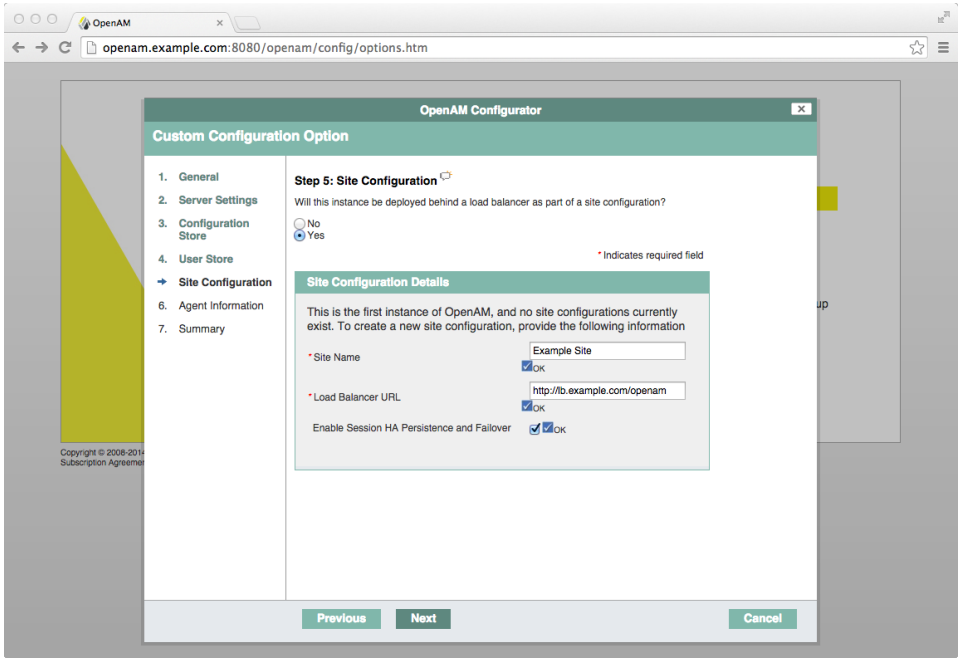
Login ID
: Directory administrator user DN. The administrator must be capable of updating schema and user data.

Password
: Password for the directory administrator user

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

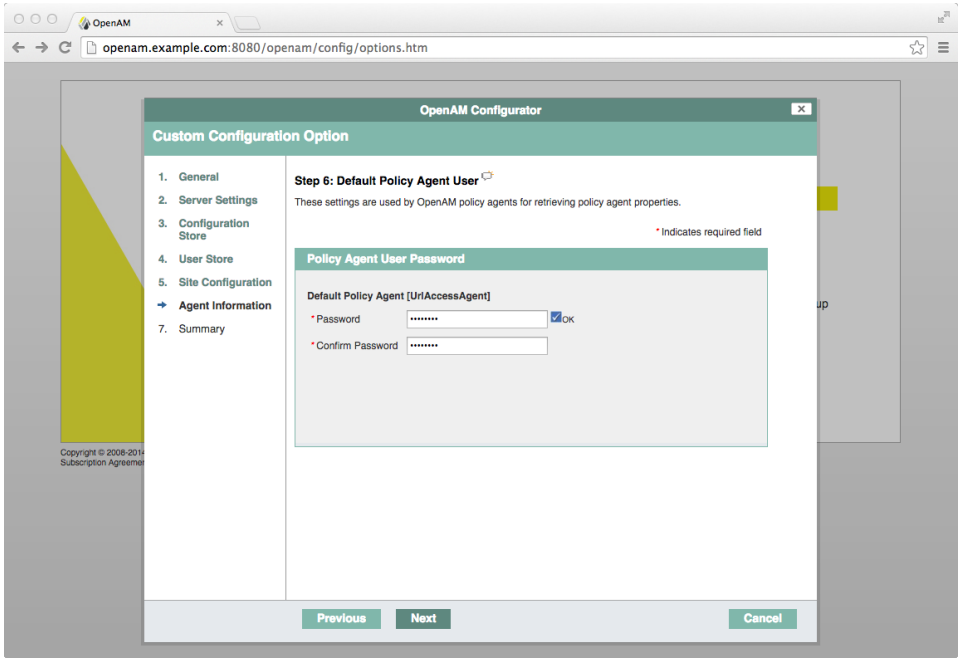
   If you have a site configuration with a load balancer, you can enable session high availability persistence and failover. OpenAM then stores sessions across server restarts, so that users do not have to login again.

   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*.

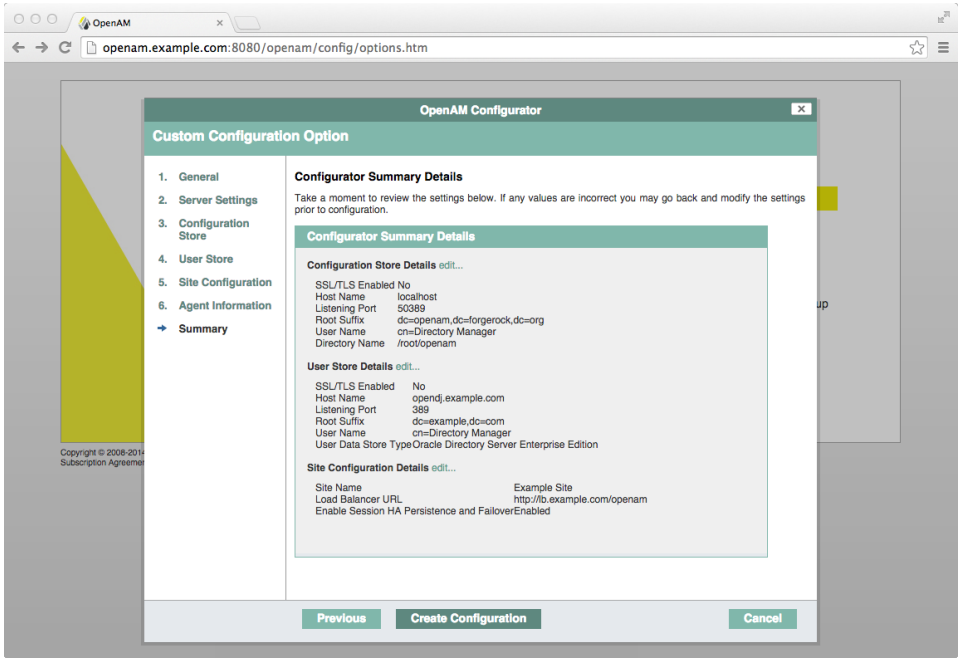


It is possible to set up a site after initial installation and configuration. Doing so is described in the chapter on *Setting Up OpenAM Session Failover*.

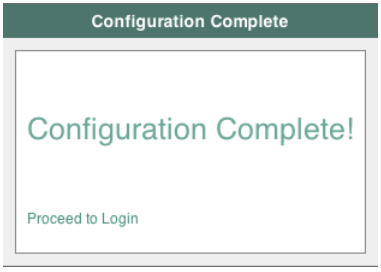
8. In the Agent Information screen, provide a password having at least 8 characters to be used by policy agents to connect to OpenAM.



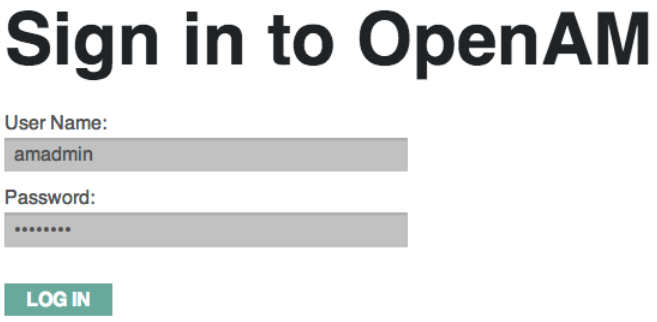
9. Check the summary screen, and if necessary click Previous to return to earlier screens if necessary to fix configuration errors.



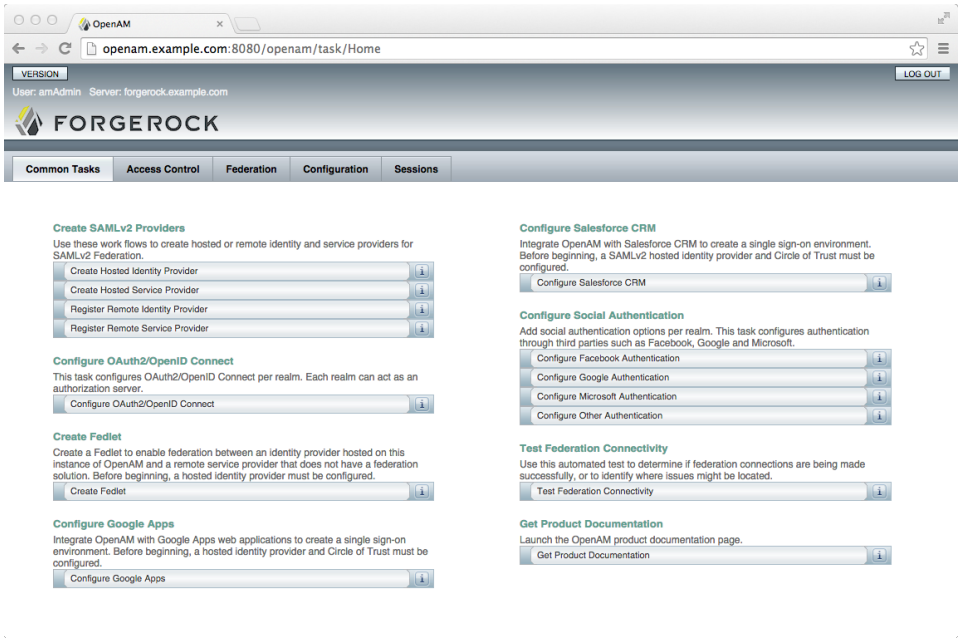
After you click Create Configuration in the summary screen, configuration proceeds, logging progress that you can read in your browser and later in the installation log. The process ends, and OpenAM shows the Proceed to Login prompt.



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



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



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

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

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 `OpenAM-ServerOnly-12.0.0.war` file in your container.

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

   ```
   $ cp OpenAM-ServerOnly-12.0.0.war /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 Custom 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`.

`SSOAdminTools-12.0.0.zip`
: Administration tools: **ampassword**, **ssoadm** and **amverifyarchive**

  See Procedure 3.1, “To Set Up Administration Tools”.

`SSOConfiguratorTools-12.0.0.zip`
: 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

The **ssoadm** administration tool requires access to OpenAM configuration files, and therefore must be installed on the same host as OpenAM core services. The **ssoadm** tool is not designed to run on a host where only the distributed authentication service (DAS) is installed.

The **ssoadm** tool also provides the ability to auto-accept the software license agreement and suppress the license acceptance screen to the user. To do so, you can add the `--acceptLicense` option to the **setup** or **setup.bat** script before you install the tool. If the option is not present, you must scroll through and accept the license interactively.

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/SSOAdminTools-12.0.0.zip
```

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

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

6. If you connect to OpenAM over HTTPS and the certificate for the container is not signed by a Certificate Authority (CA), then you must configure a trust store with OpenAM's certificate for the **ssoadm** tool. Otherwise, the **ssoadm** tool cannot trust OpenAM and cannot complete the handshake phase of setting up a secure connection.

   Once you have set up your trust store, you must update the `setup.sh` (or `setup.bat` on Windows) scripts so that **ssoadm** can reference the trust store. The **ssoadm** by default tries to trust the certificate based on the CA certificates in the Java `cacerts` truststore. The issuer certificate of the configuration data store's server certificate must be included in the truststore.

   To identify the proper trust store, add an additional option to the **java** command in the script. The following example points to the key store in

which Tomcat holds the certificate that it presents when establishing an HTTPS connection.

```
$ JAVA_HOME/bin/java -D"javax.net.ssl.trustStore=/path/to/tomcat/conf/keystore.jks"
```

7. 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.

   You can optionally apply the `--acceptLicense` argument to the end of the script if you want to auto-accept the software license agreement and suppress the license acceptance screen to the user.

```
$ 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 \
                    --acceptLicense
```

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

   After setup, the tools are located under a directory named after the instance of OpenAM. On Windows, these files are .bat scripts.

```
$ ls openam/bin/
```

```
ampassword  amverifyarchive  ssoadm
```

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

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

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

10. Check that **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.

11. If you connect to OpenAM over 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.

    To identify the proper trust store, add an additional option to the **java** command in the script. The following example points to the keystore in which Tomcat holds the self-signed certificate that it presents when establishing

an HTTPS connection. The issuer certificate of the configuration data store's server certificate must be included in the truststore.

```
$ JAVA_HOME/bin/java -D"javax.net.ssl.trustStore=/path/to/tomcat/conf/keystore.jks"
```

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.

12. 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. Create a file system directory to unpack the tools.

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

3. Unpack the tools from where you unzipped OpenAM.

```
$ cd /path/to/openam-tools/config
$ unzip ~/Downloads/openam/SSOConfiguratorTools-12.0.0.zip
Archive:  ~/Downloads/openam/SSOConfiguratorTools-12.0.0.zip
   creating: legal-notices/
  inflating: legal-notices/LICENSE.DOM-software.html
  inflating: legal-notices/NOTICE.resolver.txt
  inflating: legal-notices/LICENSE.DOM-documentation.html
        ... (more output) ...
 extracting: lib/xml-apis-2.11.0.jar
 extracting: openam-configurator-tool-12.0.0.jar
 extracting: lib/servlet-api-2.5.jar
```

4. Configure OpenAM server in a silent, unattended manner by using the openam-configurator-tool-12.0.0.jar tool after you deploy the .war.

   OpenAM server must be deployed and running, but not configured yet, when you use the tool.

   The openam-configurator-tool-12.0.0.jar relies on a properties file to specify the configuration for the OpenAM server. The following example shows the equivalent of a default configuration, which installs OpenAM to run as HTTP.

   If you want implement HTTPS, see the next step.

```
$ cp sampleconfiguration config.properties
$ vi config.properties
$ $ grep -v "^#" config.properties | grep -v "^$"
SERVER_URL=http://openam.example.com:8080
DEPLOYMENT_URI=/openam
BASE_DIR=/home/openam/openam
locale=en_US
PLATFORM_LOCALE=en_US
AM_ENC_KEY=
ADMIN_PWD=password
AMLDAPUSERPASSWD=secret12
COOKIE_DOMAIN=.example.com
ACCEPT_LICENSES=true
DATA_STORE=embedded
DIRECTORY_SSL=SIMPLE
DIRECTORY_SERVER=openam.example.com
DIRECTORY_PORT=50389
DIRECTORY_ADMIN_PORT=4444
DIRECTORY_JMX_PORT=1689
ROOT_SUFFIX=dc=openam,dc=forgerock,dc=org
DS_DIRMGRDN=cn=Directory Manager
```

```
DS_DIRMGRPASSWD=password
```

When the OpenAM server .war file is deployed and running, you can configure it by using the tool with the properties file.

```
$ java -jar openam-configurator-tool-12.0.0.jar --file config.properties
Checking license acceptance...License terms accepted.
Checking configuration directory /home/openam/openam....Success.
Installing OpenAM configuration store...Success RSA/ECB/OAEPWithSHA1AndMGF1...
Extracting OpenDJ, please wait...Complete
Running OpenDJ setupSetup command: --cli --adminConnectorPort 4444
 --baseDN dc=openam,dc=forgerock,dc=org --rootUserDN cn=Directory Manager
 --ldapPort 50389 --skipPortCheck --rootUserPassword xxxxxxx --jmxPort 1689
 --no-prompt --doNotStart --hostname openam.example.com ...
...Success
Installing OpenAM configuration store in /home/openam/openam/... ...Success.
Creating OpenAM suffixImport+task+ ... ...Success
Tag swapping schema files....Success.
Loading Schema opendj_config_schema.ldif...Success.

...

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

...

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

5. To configure HTTPS, you create a properties file and include the `SERVER_URL` property with the HTTPS URL and set the `DIRECTORY_SSL` to `SSL` as follows (the other properties remain the same):

```
$ cp sampleconfiguration config.properties
$ vi config.properties
$ $ grep -v "^#" config.properties | grep -v "^$"
SERVER_URL=https://openam.example.com:1443
DEPLOYMENT_URI=/openam
BASE_DIR=/home/openam/openam
locale=en_US
PLATFORM_LOCALE=en_US
AM_ENC_KEY=
ADMIN_PWD=password
AMLDAPUSERPASSWD=secret12
COOKIE_DOMAIN=.example.com
ACCEPT_LICENSES=true
DATA_STORE=embedded
DIRECTORY_SSL=SSL
DIRECTORY_SERVER=openam.example.com
```

```
DIRECTORY_PORT=50389
DIRECTORY_ADMIN_PORT=4444
DIRECTORY_JMX_PORT=1689
ROOT_SUFFIX=dc=openam,dc=forgerock,dc=org
DS_DIRMGRDN=cn=Directory Manager
DS_DIRMGRPASSWD=password
```

Then, when the OpenAM .war file is deployed and the server is running, configure the server to use HTTPS using the openam-configurator-tool-12.0.0.jar tool with the properties file as follows:

```
java '-Djavax.net.ssl.trustStore=PATH_TO_JKS_TRUSTSTORE' \
      -jar openam-configurator-tool-12.0.0.jar \
      --file config.properties
```

For additional information about the command-line tool, see the reference documentation for **configurator.jar**.

Note

OpenAM supports two methods to auto-accept the software licensing agreement and suppress the display of the licence acceptance screen to the user: using the configuration file or using a command-line option. You can include an optional `ACCEPT_LICENSES=true` property in the openam-configurator-tool-12.0.0.jar configuration file.

You can also use the `--acceptLicense` option with the configurator tool on the command line. The configuration file property has precedence over the command-line option. The default value is `false`, which always displays the license acceptance screen.

```
$ java -jar openam-configurator-tool-12.0.0.jar \
  --file config.properties \
  --acceptLicense
```

Chapter 4

Installing Multiple Servers

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

4.1 Things to Consider When Installing Multiple Servers

When installing multiple servers, consider the following points.

- You generally install multiple servers to provide service availability. If one server is down for any reason, another server can respond instead. This means that you need something between incoming traffic and OpenAM to route around servers that are down.

  OpenAM has the concept of OpenAM *site* for this purpose. In an OpenAM site, multiple OpenAM servers are configured in the same way, and accessed through a load balancer layer.[1] The load balancer can be implemented in hardware or software, but it is separate and independent from OpenAM software. When installed properly, a site configuration improves service availability, as the load balancer routes around OpenAM servers that are down, sending traffic to other servers in the site.

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

[1] Technically it is possible to configure a site with only one OpenAM server.

As an alternative, or in addition, you can use a separate reverse proxy service, or the OpenAM distributed authentication UI. The distributed authentication UI exposes a subset of OpenAM functionality. For instructions on setting up the distributed authentication UI, see *Installing OpenAM Distributed Authentication*.

- When you are protecting OpenAM with a load balancer or proxy service, configure your container so that OpenAM can trust the load balancer or proxy service. OpenAM trusts the distributed authentication UI as the distributed authentication UI uses credentials registered with OpenAM.

- OpenAM authentication can depend on information about the user to authenticate, such as the IP address where the request originated. When OpenAM is accessed through a load balancer or proxy layer, pass this information along using request headers. Also configure OpenAM to consume and to forward the headers as necessary. See Section 4.3, “Handling HTTP Request Headers” for details.

4.2 Configuring OpenAM Sites

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

This section includes the following procedures.

- Procedure 4.1, “To Configure a Site with a First OpenAM Server”

- Procedure 4.2, “To Configure Site Load Balancing”

Procedure 4.1. To Configure a Site with a First OpenAM Server

You might already have configured an OpenAM server before realizing that a site is what you want.

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

1. Login to OpenAM Console as administrator, by default `amadmin`, and then browse to Configuration > Servers and Sites > Sites.

2. Click New to start configuring the new site.

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

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

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

4. Click Save to keep the site configuration.

5. Under Configuration > Servers and Sites > Server, click the link to the server configuration.

6. 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 new site you have configured.

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

Procedure 4.2. To Configure Site Load Balancing

If you did not set up the site during initial configuration, first follow the instructions in Procedure 4.1, "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*, set Parent Site to the site you created, and then Save your work.

2. Make the `amlbcookie` value unique for each OpenAM server.

   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. The load balancer should send subsequent requests to that server to avoid overhead resulting from the server that gets the request being different from the server that authenticated the user.

   When traffic is protected with HTTPS, this approach requires that you terminate the connection on the load balancer. You then either re-encrypt the traffic from the load balancer to OpenAM, or make connections from the load balancer to OpenAM over HTTP.

   a. For each OpenAM server console in the site, browse to Configuration > Servers and Sites > Servers > *Server Name* > Advanced, and set `com.iplanet.am.lbcookie.value` to a unique value.

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

Changes take effect only after you restart the OpenAM server.

b. Restart each OpenAM server where you changed the cookie value.

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

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

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

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

Note

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

When an OpenAM request to read a session arrives at a server that does not hold the user's session, the OpenAM server attempts to locate the session from the Core Token Service session store by default.

OpenAM servers can also use crosstalk to locate remote sessions. With crosstalk, OpenAM servers communicate with each other through a back channel to locate remote sessions. Because crosstalk generates network traffic, locating sessions from the Core Token Service session store is preferred for performance reasons.

Requests to log out or set a session attribute always use crosstalk to ensure the integrity of the update requests.

See *Setting Up OpenAM Session Failover* for information about configuring remote session location options.

4.3 Handling HTTP Request Headers

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

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

If you use the distributed authentication UI, you can retain headers by using the `openam.retained.http.request.headers` setting as described in *Installing OpenAM Distributed Authentication*.

Also configure OpenAM to consume and to forward the headers as necessary. When configuring OpenAM through the console, you set the following properties under Configuration > Servers and Sites > Servers > *`Server Name`* > Advanced.

For example, to configure OpenAM to look for the client IP address in the `X-Forwarded-For` request header, set the advanced configuration property `com.sun.identity.authentication.client.ipAddressHeader` to `X-Forwarded-For`.

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

Chapter 5

Installing OpenAM Distributed Authentication

OpenAM provides a login interface called the *distributed authentication service (DAS)*, deployed within network demilitarized zones to limit OpenAM's exposure to the Internet. Login requests 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. Create a 2.2 Agent profile in OpenAM for the DAS to connect to the server.

   You can create the profile in OpenAM Console under Access Control > *`Realm Name`* > Agents > 2.2 Agents. See the *Administration Guide* section on *Configuring Version 2.2 Policy Agents* for details.

4. Deploy the OpenAM-DistAuth-12.0.0.war 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.

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

6. 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 5.1. To Deploy the DAS on Tomcat

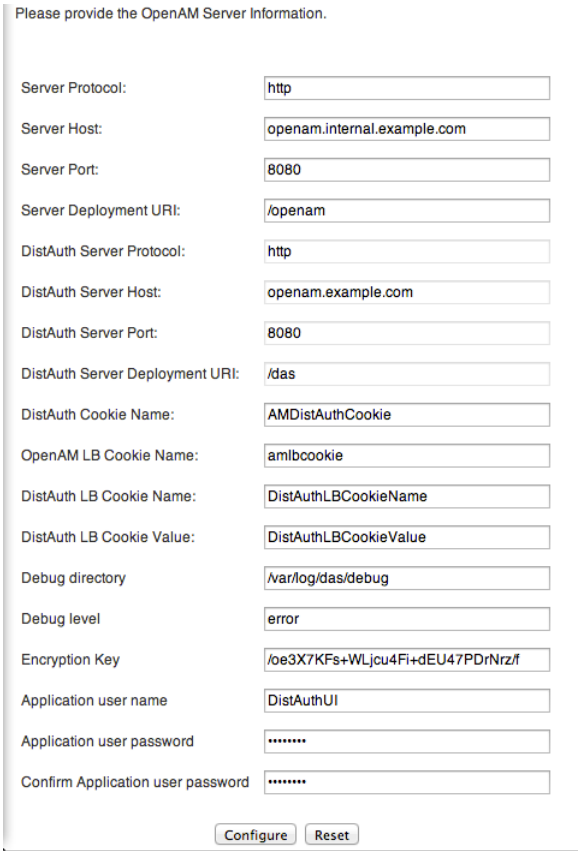
1. Copy the `OpenAM-DistAuth-12.0.0.war` file into the `webapps/` directory.

```
$ cp ~/Downloads/openam/OpenAM-DistAuth-12.0.0.war \
 /path/to/tomcat/webapps
```

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

Procedure 5.2. To Configure the DAS

1. Configure the DAS using the agent profile to connect to OpenAM. The DAS agent profile can only be configured in the top level realm ("/"). If working with sub-realms, you may then define the Realm/DNS alias for the DAS under Access Control > *Realm Name* > General.



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.

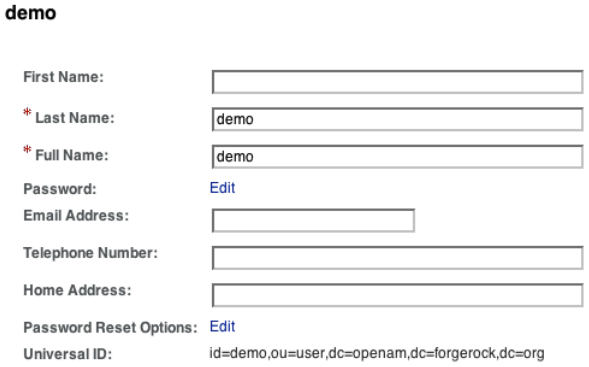
Application User Name
: The DAS uses this 2.2 Agent identity to authenticate to OpenAM.

  You can find the 2.2 Agent profile in OpenAM Console under Access Control > *`Realm Name`* > Agents > 2.2 Agents as described above.

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`.



   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.

   Configuration file names depend on the path where the DAS is deployed, and end in `AMDistAuthConfig.properties`. For example, if the DAS is deployed under `/path/to/tomcat/webapps/das/` then the configuration file name is `$HOME/FAMDistAuth/_path_to_tomcat_webapps_das_AMDistAuthConfig.properties`.

4. 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
```

5. If your deployment includes multiple OpenAM servers, then edit the DAS configuration file mentioned in Step 3 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.

6. 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.

7. If your deployment uses a custom login URI, then edit the DAS configuration file mentioned in Step 3 to add the custom login URI to the whitelist specified by the `org.forgerock.openam.cdc.validLoginURIs` property.

   Example: `org.forgerock.openam.cdc.validLoginURIs=/UI/Login,/customLoginURI`

   For more information about this property, see the *Reference* section on advanced properties, *Servers > Advanced*.

5.1 Configuring Valid goto URL Resources

By default, OpenAM redirects the user to the URL specified in the `goto` and `gotoOnFail` query string parameters supplied to the authentication interface in the login URL.

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

When setting valid `goto` URLs, you can use the "*" wildcard, where "*" matches all characters except "?". The following rules apply:

- To allow only a single host name to be used when redirecting the user after authentication, specify the host name.

  For example, if you specify `http*://secure.example.com`, then you can redirect only to URLs on `secure.example.com`, not URLs on `www.example.com`.

- Start the domain name with `*.` to allow all host names in the domain to be used when redirecting the user after authentication.

  For example, to allow redirects to URLs on any hosts in the domain `secure.example.com`, use `http*://*.secure.example.com`.

  Note that `http*://*.secure.example.com` matches URLs on `www.secure.example.com`, and also URLs on `www.subdomain.secure.example.com`.

  Also note that the `*` wildcard in the protocol, `http*://`, matches URLs starting with either `http://` or `https://`.

- To match URLs using secure connections to `www.example.com` on any port, but no URLs with query string parameters, use `https://www.example.com:*/*`.

Procedure 5.3. To Configure Valid goto URL Resources

You can increase security against possible phishing attacks through open redirect by specifying a list of valid URL resources using the Valid `goto` URL Resource service.

OpenAM only redirects a user if the `goto` and `gotoOnFail` URL matches any of the resources specified in this setting. If no setting is present, it is assumed that the `goto` or `gotoOnFail` URL is valid.

When setting valid `goto` URLs, you can use the "*" wildcard, where "*" matches all characters except "?". For more specific patterns, use resource names with wildcards as described in the procedure, *Configuring Valid goto URL Resources*.

1. Open the OpenAM console. Click Access Control > *`Realm Name`* > Services, and then click Add.

2. Select Validation Service.

3. In the New Value box, enter a valid `goto` URL, and then click Add. You can repeat this step to enter additional URLs.

Chapter 6

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.

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

You configure which set of OpenAM pages the end user sees.

- By default, end users see the XUI pages.

  See Section 6.1, “Configuring the XUI” for details.

- For backwards compatibility, OpenAM bundles the classic UI pages as well. This can be useful when upgrading as it allows you to use customizations developed with earlier versions of OpenAM.

  To enable the classic UI, disable the XUI.

  You can disable XUI globally for an OpenAM server in OpenAM console under Configuration > Authentication > Core > Global Attributes. Clear XUI Interface Enabled, save your work, and log out. When you return to the login page, you see the classic UI.

While customizing the UI, you can set the advanced server property, `org.forgerock.openam.core.resource.lookup.cache.enabled`, to `false` to allow OpenAM immediately to pick up changes to the files as you customize them. This includes both the XML callback files for authentication modules used both by the XUI and also by the classic UI, and the JSP files used by the classic UI. You can set advanced server properties in OpenAM console under Configuration > Servers and Sites > *Server Name* > Advanced. Before using OpenAM in production, set `org.forgerock.openam.core.resource.lookup.cache.enabled` back to the default setting, `true`.

6.1 Configuring the XUI

XUI is the default UI for OpenAM. Compared with the classic UI, XUI does more on the client side. XUI is based on JavaScript that accesses OpenAM REST APIs, and is styled with LESS CSS.

Interface Stability: Evolving

The main XUI configuration file, `XUI/themeConfig.json` under the directory where you unpack OpenAM, serves to customize the look and feel of end user pages. For details, see the *Reference* chapter on *XUI Configuration Parameters*.

You can set different themes for different realms by adding each realm configuration to the array of "themes", with the realm "name", "path", and "realms" fields set appropriately. The following example shows a new theme for `myRealm` added after the default theme.

```
{
     "themes": [
         {
            "name": "default",
            "path": "",
            "realms": [".*"],
            "regex"" true,

         . . .

                  "footer": {
                     "mailto": "info@forgerock.com",
                     "phone": ""
                  }
             }
         },
         {
            "name": "myRealm",
            "path": "path/to/myRealm/",
            "realms": ["myRealm.*"],
            "regex": true,

         . . .

                "footer": {
```

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

Make sure `themeConfig.json` is valid JSON after you apply your changes. If in doubt, use a validator such as http://jsonlint.com.

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 requires 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 `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 might not be shown in your browser.

Much of the text supplied in XUI is stored in `translation.json` files for each locale. To customize the English text, edit `XUI/locales/en/translation.json` under the directory where OpenAM is unpacked for deployment. To prepare a translation for a new locale, copy the English file, `XUI/locales/en/translation.json`, to `XUI/locales/`*`new-locale`*`/` and edit the copy changing only the values not the keys, and taking care not to change the JSON structure or to render it invalid.

6.2 Updating the Classic UI (Legacy)

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

Interface Stability: Deprecated

Classic UI 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.

Classic UI images are located under `images/`, and CSS under `css/` where OpenAM files are unpacked for deployment. If you choose to modify images for your deployment, maintain image size dimensions to avoid having to change page layout.

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 6.1, “To Copy the Pages to Customize For the Top-Level Realm”
- Procedure 6.2, “To Copy the Pages to Customize For Another Realm”
- Procedure 6.3, “To Customize Files You Copied”

Procedure 6.1. 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 `dc=openam,dc=forgerock,dc=org`, 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 Section 6.3, “How OpenAM Looks Up UI Files” for details.

3. You can now either follow the steps in Procedure 6.2, “To Copy the Pages to Customize For Another Realm”, or in Procedure 6.3, “To Customize Files You Copied”.

Procedure 6.2. 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 `dc=openam,dc=forgerock,dc=org`.

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

```
$ mkdir -p openam/services/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
```

3. You can now follow the steps in Procedure 6.3, “To Customize Files You Copied”.

Procedure 6.3. 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.

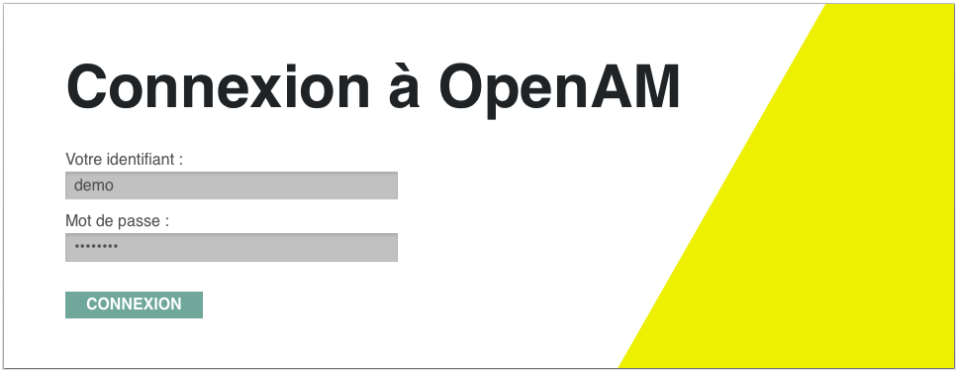
2. Modify the localized text, using UTF-8 without escaped characters, by changing only the original text strings in the .xml files.

   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. After making the changes, restart OpenAM or the web container where it runs.

4. 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. As mentioned in the tip at the outset of this section, build a customized OpenAM .war that includes your tested changes, and use this customized .war to deploy OpenAM.

Procedure 6.4. To Customize UI Elements

To customize classic UI elements such as button text on the login screen, follow these steps.

1. Unpack the core OpenAM library, `openam-core-12.0.0.jar`, that contains the text in Java properties files.

   This library is available under `WEB-INF/lib/` where OpenAM is unpacked for deployment. In the following example OpenAM is deployed on Apache Tomcat.

   ```
   $ mkdir openam-core && cd openam-core
   $ jar xf /path/to/tomcat/webapps/openam/WEB-INF/lib/openam-core-12.0.0.jar
   ```

2. Edit only property values in the appropriate properties files.

3. Prepare a new core OpenAM library with your modifications.

   ```
   $ jar cf ../openam-core-12.0.0.jar *
   ```

4. Replace the existing core OpenAM library with your modified version.

   The following example replaces the library only in a deployed OpenAM server.

   ```
   $ cp openam-core-12.0.0.jar /path/to/tomcat/webapps/openam/WEB-INF/lib/
   ```

   When preparing for production deployment make the modification in the OpenAM war file, `OpenAM-12.0.0.war`, instead.

5. Restart OpenAM or the container in which it runs to load the changes.

6.3 How OpenAM Looks Up UI Files

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

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 except for locale territories. All of the default directories are already lowercase.

Locale settings play an important role in how OpenAM looks up UI files. A locale consists of a language and optionally a territory such as `en` to specify the English language, or `en_GB` to specify British English. Locale settings are determined at authentication time, and are then set in the authentication context. To change locales, the user must reauthenticate. OpenAM allows you and also clients to set locale in the ways described below.

When finding the UI files that best match the user's locale, OpenAM takes two locale settings into account.

1. Requested locale

   OpenAM arrives at the requested locale based on an optional `locale` query string parameter, an optional HTTP `Accept-Language` header from the browser, and the Default Locale set in the configuration for OpenAM.

2. Platform locale

   When OpenAM cannot find a match for the user's requested locale, it tries to use the platform locale, which is the locale for the Java Virtual Machine (JVM) where OpenAM runs.

   If neither the requested locale nor the platform locale result in a match, OpenAM returns the default files that are not localized.

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

Configuration suffix RDN value
: When you set up OpenAM to store its configuration in a directory server, you provide the distinguished name of the configuration suffix, by default `dc=openam,dc=forgerock,dc=org`. Therefore, the default relative distinguished name attribute value is `openam`.

Client locale query string parameter
: The client can request a locale by using the `locale` query string parameter when performing an HTTP GET on the login page.

  For example, a client can specify `locale=fr` to request the French language.

Client (browser) locale language and territory
: The client can specify a locale by using the HTTP `Accept-Language` header. End users set this behavior by choosing languages and territory settings in their web browser preferences.

  The value of this header can include a list of languages with information about how strongly the user prefers each language. OpenAM uses the first language in the list.

Default locale
: You set the default locale in OpenAM when you install OpenAM core services. You can change the default locale either through OpenAM console under Configuration > Servers and Sites > *server-name* > General > System > Default Locale, or by setting the server configuration property, `com.iplanet.am.locale`.

  Default locale only affects the requested locale. Do not confuse the Default Locale setting with the locale that OpenAM uses when it cannot find matching UI files for the requested locale.

  Default: `en_US`

Requested locale
: OpenAM determines the requested locale based on multiple settings.

  If the `locale` query string parameter is set, OpenAM uses this setting as the requested locale.

  Otherwise, if the client set the `Accept-Language` header, OpenAM uses this setting as the requested locale.

  Otherwise OpenAM uses the default locale as the requested locale.

Platform locale language and territory
: The locale for the Java Virtual Machine (JVM) where OpenAM runs is the platform locale. Platform locale is the alternative when OpenAM cannot find files for the requested locale.

  By default the JVM uses the system locale. You can, however, set the JVM platform locale when starting Java by using Java system properties. The following example that sets the platform locale to the Hungarian language in Hungary.

  ```
  java -Duser.language=hu -Duser.region=HU other options
  ```

  See the documentation about your JVM for details.

If OpenAM cannot find matching UI files either for the requested locale or for the platform locale, it returns UI files that are not localized.

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

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

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. For example, `Login.jsp` has the same name in all locales.

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

```
suffix_requested-locale-language_requested-locale-territory/services/realm/client-name/file-name
```

Example 6.1. UI File Lookup

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

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

Chapter 7

Configuring the Core Token Service (CTS)

The Core Token Service (CTS) provides persistent and highly available token storage for OpenAM session, OAuth 2.0, and SAML 2.0 tokens. CTS is set up in a generalized token storage format, which by default is always used for OAuth 2.0 tokens. If configured, it can also be used to persist session and SAML 2.0 tokens.

The easiest CTS configuration is to run the default configuration option in the installer, which uses an embedded OpenDJ directory server to store both configuration and CTS tokens including the CTS schema. If your deployment requires it, you can go beyond the default embedded directory server and deploy separate external directory servers to store configuration and CTS data.

Important

CTS relies on OpenDJ to store and replicate its tokens. Only OpenDJ is supported for CTS. No other directory server is supported currently.

If you deploy separate external stores for configuration and CTS data respectively, note that external *configuration* stores support OpenDJ and DSEE servers, while external *CTS* stores only support OpenDJ currently.

CTS tokens are volatile and change frequently, while other data stored in an OpenDJ server is considerably more static. Therefore, the performance tuning requirements are quite different for both types of data.

If you choose to set up CTS in an external OpenDJ instance, you will have to install OpenDJ separately from OpenAM. For more information, see the *OpenDJ Installation Guide*.

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

The procedures to run these tasks are presented in the following sections.

7.1 CTS Configuration Parameters

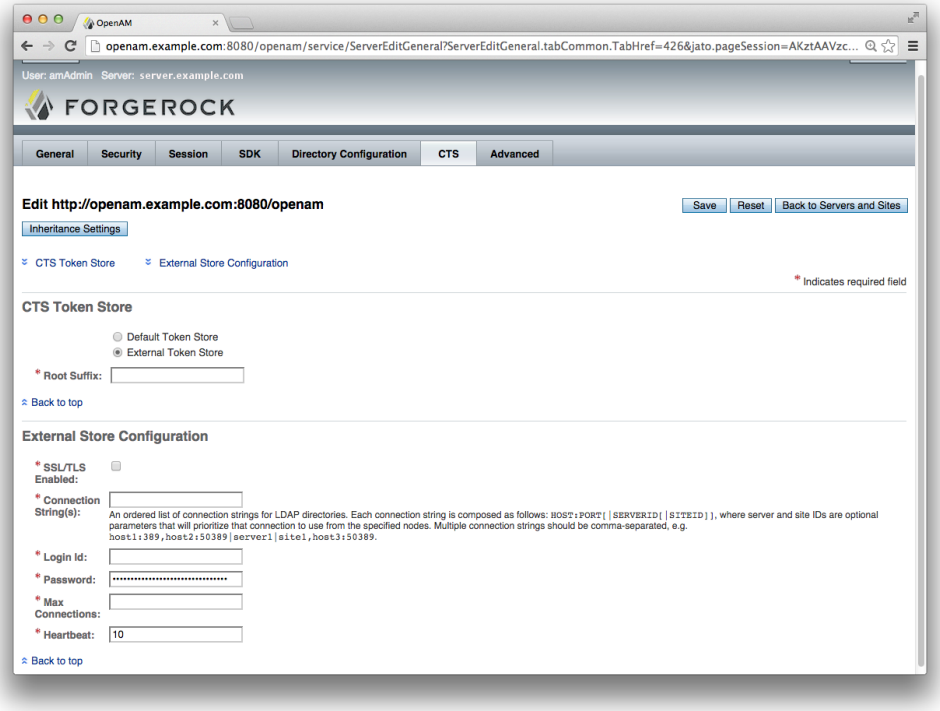
If you want to reconfigure an existing implementation of CTS, any reconfiguration orphans any tokens that are currently stored, which may require users to log in again. To keep this from happening, disable client access to OpenAM before making any changes. Any changes require a server restart to put 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.



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`

  `dc=cts,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`

  By default, disabled. Click the checkbox to enable SSL or TLS for the connection to the directory server.

- `Connection String(s)`

Specifies the ordered list of connection strings for the external OpenDJ directory servers. The format is: `HOST:PORT[|SERVERID[|SITEID]]`, where `SERVERID` and `SITEID` are optional parameters, which prioritize the particular connection when used by the specified node(s).

```
opendj.example.com:389, opendj2.example.com:50389|server1|site1
```

- `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 7.3, "CTS Access Control Instructions". You can bypass access control by binding with a root DN such as `cn=Directory Manager`.

- `Password`

- `Max Connections`

  When the directory service backing the CTS is external (differs from the directory service backing the OpenAM configuration) then this setting configures the maximum number of connections in the connection pool used to access the directory service for the CTS. One connection is reserved for cleanup of expired tokens. The other connections are available for CTS operations.

  `17` (16 connections for CTS operations, 1 for token cleanup)

- `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.

7.2 CTS Schema and Indexes

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-indices.ldif` files are installed. If you upgrade to OpenAM 12.0.0 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 7.1. 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

7.3 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=cts,dc=example,dc=com`, but set your DN to match your installation.

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

7.4 Preparing an OpenDJ Directory Service for CTS

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

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

The following example script installs a single instance of an OpenDJ directory server as an external CTS store. The procedure assumes that you have installed OpenAM with its default configuration settings.

The example directory server install script uses the following parameters:

Table 7.2. Example OpenDJ Setup Parameters

Parameter	Example Inputs
`Hostname`	Provide a Fully Qualified Domain Name, like opendj.example.com
`LDAP Port`	1389
`generateSelfSignedCertificate`	true
`enableStartTLS`	true
`ldapsPort`	1636
`jmxPort`	1689
`Administration Connector Port`	4444
`Root User DN`	cn=Directory Manager
`Root User DN Password`	(arbitrary)
`baseDN`	Not needed. The base DN for the backend will be configured in a later step.
`ldifFile`	Not needed. This file is typically used to import user entries.
`sampleData`	Not needed
`Start Server After Config`	Yes

Procedure 7.1. To Prepare and Run the External OpenDJ CTS Install Script

1. Download OpenDJ.
2. Create the install script. The following is an example, which you can edit for your specific system.

```
#!/bin/sh

#
# copyright (C) 2014 ForgeRock AS
#
# cts-setup.sh: This script installs and configures an external CTS data store.
# It assumes that you have downloaded the OpenDJ zip file to a local
# folder.

# Reset the tmp folder
T=/tmp/ldif
rm -rf $T
mkdir $T

# Define variables used in this script. Make them specific to your deployment.

TOMCAT_OPENAM_WEBAPP=/usr/local/tomcat/webapps/openam/WEB-INF/template/ldif/sfha
USER="cn=Directory Manager"
PASS="pwd"
PORT=4444
CTS_DN="dc=cts,dc=example,dc=com"

# Make sure to download the OpenDJ Zip file and place it in the same folder as
# this script
DS="OpenDJ-2.6.2.zip"

# Sanity Check
[ ! -d "$TOMCAT_OPENAM_WEBAPP" ] && echo "TOMCAT OPENAM WEBAPP folder inaccessible" \
 && exit 1
[ ! -f "$DS" ] && echo "$DS is not present" && exit 1

# Create a properties file for the OpenDJ install
printf '#\n' >> $T/setup.props
printf '# Sample properties file to set up the OpenDJ directory server\n' >> \
 $T/setup.props
printf '#\n' >> $T/setup.props
printf 'hostname                        = opendj.example.com\n' >> $T/setup.props
printf 'ldapPort                        = 1389\n' >> $T/setup.props
printf 'generateSelfSignedCertificate = true\n' >> $T/setup.props
printf 'enableStartTLS                = true\n' >> $T/setup.props
printf 'ldapsPort                       = 1636\n' >> $T/setup.props
printf 'jmxPort                         = 1689\n' >> $T/setup.props
printf "adminConnectorPort            = $PORT\n" >> $T/setup.props
printf "rootUserDN                    = $USER\n" >> $T/setup.props
printf 'rootUserPassword              = pwd\n' >> $T/setup.props
#printf '#baseDN                       = dc=example,dc=com\n' >> $T/setup.props
#printf '#ldifFile                     = /path/to/Example.ldif\n' >> $T/setup.props
#printf '#sampleData                   = \n' >> $T/setup.props
```

```
# Create the CTS base dn and ACIs entries and write them to a file
# Linefeeds have been added for publication purposes.
printf '#\n' >> $T/add-cts-entries.ldif
printf '# add-cts-entries.ldif: base DN and ACIs\n' >> $T/add-cts-entries.ldif
printf '#\n' >> $T/add-cts-entries.ldif
printf "dn: $CTS_DN\n" >> $T/add-cts-entries.ldif
printf 'objectclass: top\n' >> $T/add-cts-entries.ldif
printf 'objectclass: domain\n' >> $T/add-cts-entries.ldif
printf 'dc: cts\n' >> $T/add-cts-entries.ldif
printf "aci: (targetattr=\"*\")(version 3.0;acl \"Allow entry search\"; \
 allow (search, read)(userdn = \
 "ldap:///uid=openam,ou=admins,$CTS_DN\");)\n" >> $T/add-cts-entries.ldif
printf "aci: (targetattr=\"*\")(version 3.0;acl \"Modify config entry\"; \
 allow (write)(userdn = \
 "ldap:///uid=openam,ou=admins,$CTS_DN\");)\n" >> $T/add-cts-entries.ldif
printf "aci: (targetcontrol=\"2.16.840.1.113730.3.4.3\") \
 (version 3.0;acl \"Allow persistent search\"; \
 allow (search, read)(userdn = \
 "ldap:///uid=openam,ou=admins,$CTS_DN\");)\n" >> $T/add-cts-entries.ldif
printf "aci: (version 3.0;acl \"Add config entry\"; allow (add)(userdn = \
 "ldap:///uid=openam,ou=admins,$CTS_DN\");)\n" >> $T/add-cts-entries.ldif
printf "aci: (version 3.0;acl \"Delete config entry\"; allow (delete)(userdn = \
 "ldap:///uid=openam,ou=admins,$CTS_DN\");)\n" >> $T/add-cts-entries.ldif
printf '\n' >>$T/add-cts-entries.ldif
printf "dn: ou=admins,$CTS_DN\n" >> $T/add-cts-entries.ldif
printf 'objectclass: top\n' >> $T/add-cts-entries.ldif
printf 'objectclass: organizationalUnit\n' >> $T/add-cts-entries.ldif
printf 'ou: admins\n' >> $T/add-cts-entries.ldif
printf '\n' >>$T/add-cts-entries.ldif
printf "dn: uid=openam,ou=admins,$CTS_DN\n" >> $T/add-cts-entries.ldif
printf 'objectclass: top\n' >> $T/add-cts-entries.ldif
printf 'objectclass: person\n' >> $T/add-cts-entries.ldif
printf 'objectclass: organizationalPerson\n' >> $T/add-cts-entries.ldif
printf 'objectclass: inetOrgPerson\n' >> $T/add-cts-entries.ldif
printf 'cn: openam\n' >> $T/add-cts-entries.ldif
printf 'sn: openam\n' >> $T/add-cts-entries.ldif
printf 'uid: openam\n' >> $T/add-cts-entries.ldif
printf 'userPassword: secret12\n' >> $T/add-cts-entries.ldif
printf 'ds-privilege-name: subentry-write\n' >> $T/add-cts-entries.ldif
printf 'ds-privilege-name: update-schema\n' >> $T/add-cts-entries.ldif

# Unzip and install OpenDJ using the properties file
echo ""
echo "... Unpacking OpenDJ and installing ..."
unzip "$DS" && cd opendj
./setup --cli --propertiesFilePath $T/setup.props --acceptLicense --no-prompt

# Create the CTS Backend
echo ""
echo "... Creating backend ..."
echo ""
bin/dsconfig create-backend \
--backend-name cts-store \
--set base-dn:"$CTS_DN" \
--set enabled:true \
--type local-db \
--port $PORT \
--bindDN "$USER" \
--bindPassword $PASS \
--trustAll \
--no-prompt
echo "Backend created"
```

```
# Verify Backend
#bin/dsconfig list-backends \
#--port $PORT \
#--bindDN "$USER" \
#--bindPassword $PASS \
#--trustAll \
#--no-prompt

# Add the Base DN and ACIs
echo ""
echo "...Adding Base DN and ACIs..."
echo ""
bin/ldapmodify \
--port $PORT \
--bindDN "$USER" \
--bindPassword $PASS \
--defaultAdd \
--filename $T/add-cts-entries.ldif \
--useSSL \
--trustAll
echo "BaseDN and ACIs added."

# Verify BaseDN and ACIs
#bin/ldapsearch --port $PORT --bindDN "$USER" --bindPassword $PASS \
 --baseDN "$CTS_DN" --searchscope sub --useSSL --trustAll  "(objectclass=*)"

# Add the Admin Global ACI
echo ""
echo "...Adding Admin Global ACIs..."
echo ""
bin/dsconfig set-access-control-handler-prop \
--add global-aci:'(target = "ldap:///cn=schema")(targetattr = "attributeTypes || \
 objectClasses")(version 3.0; acl "Modify schema"; allow (write) userdn = \
 "ldap:///uid=openam,ou=admins,dc=cts,dc=example,dc=com";)' \
--port $PORT \
--bindDN "$USER" \
--bindPassword $PASS \
--trustAll \
--no-prompt
echo "Global ACI added."

# Verify Global ACI
#bin/dsconfig get-access-control-handler-prop --property global-aci --port $PORT \
 --bindDN "$USER" --bindPassword $PASS -X -n

# Copy the Schema, Indexes, and Container files for CTS
echo ""
echo "... Begin copying schema, indexes, and container ..."
cp $TOMCAT_OPENAM_WEBAPP/cts-add-schema.ldif $T/cts-add-schema.ldif
cat $TOMCAT_OPENAM_WEBAPP/cts-indices.ldif | sed -e 's/@DB_NAME@/cts-store/' > \
 $T/cts-indices.ldif
cat $TOMCAT_OPENAM_WEBAPP/cts-container.ldif | sed -e \
 "s/@SM_CONFIG_ROOT_SUFFIX@/$CTS_DN/" > $T/cts-container.ldif
echo "Schema, index, and container files copied."

# Add the Schema Files
echo ""
echo "... Adding CTS Schema ..."
bin/ldapmodify --port $PORT --bindDN "$USER" --bindPassword $PASS \
 --fileName $T/cts-add-schema.ldif --useSSL --trustAll
```

```
# Add the CTS Indexes
echo ""
echo "... Adding CTS Indexes ..."
bin/ldapmodify --port $PORT --bindDN "$USER" --bindPassword $PASS --defaultAdd \
 --fileName $T/cts-indices.ldif --useSSL --trustAll

# Add the CTS Container Files
echo ""
echo "... Adding CTS Container ..."
bin/ldapmodify --port $PORT --bindDN "$USER" --bindPassword "$PASS" --defaultAdd \
 --fileName $T/cts-container.ldif --useSSL --trustAll

# Rebuild the Indexes
echo ""
echo "... Rebuilding Index ..."
bin/rebuild-index --port $PORT --bindDN "$USER" --bindPassword "$PASS" \
 --baseDN "$CTS_DN" --rebuildALL --start 0 --trustAll

# Verify the Indexes
echo ""
echo "... Verifying Index ..."
bin/verify-index --baseDN "$CTS_DN"

echo ""
echo "Your CTS External Store has been successfully installed and configured."
exit 0
```

3. Make the file executable.

```
chmod u+x cts-setup.sh
```

4. Run the script. Make sure the OpenDJ zip file is in your same directory and update the script accordingly.

```
./cts-setup.sh
```

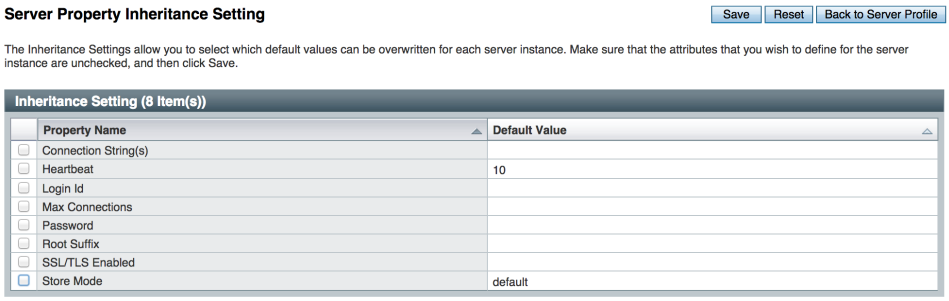
Procedure 7.2. To Configure the CTS Data Store

The following example procedure assumes that you have implemented the previous procedure. Also note that the OpenAM server will require a restart for the configuration changes to take effect.

1. Log into the OpenAM Console.
2. On the main OpenAM Console screen, click Configuration > Servers and Sites > `Server Name` > CTS. For this example, the `Server Name` is `http://openam.example.com:8080/openam`.
3. On the Edit `Server Name` screen, click Inheritance Settings. By default, the CTS properties are inherited from the Configuration data store settings

created in a previous step. We want to clear or uncheck these properties to modify these properties for CTS.

4. On the Server Property Inheritance Setting screen, uncheck the properties that you want to modify for CTS, and then click Save.



   - Connection String(s). For example, `host1:389,host2:50389|server1|site1, host3:50389`.
   - Heartbeat. Default is 10.
   - Login Id. Specifies the directory server's bind DN.
   - Max Connections. Sets the maximum number of connections to the data store.
   - Password. Specifies the directory servers admin password.
   - Port. Specifies the LDAP port of the directory server.
   - Root Suffix. Specifies the base DN of the CTS store.
   - SSL/TLS Enabled. Specifies if SSL or TLS is configured for the connection to the directory server.
   - Store Mode. Specifies if the data store is a default (internal) data store or an external store. This property is required if you want to configure an external CTS store. Note that we recommend all settings to be enabled for editing and populated with the appropriate values.

5. On the Edit `Server Name` screen, click External Token Store and then enter the Root Suffix for the CTS store. For this example, enter `dc=cts,dc=example, dc=com`.

6. Under External Store Configuration, enter the properties for the CTS backend and then click Save.

Connection String(s)
`host1:389,host2:50389|server1|site1,host3:50389`

Login Id
`uid=openam,ou=admins,dc=cts,dc=example,dc=com`

Password
Enter the Login ID password. For this example, enter `secret12`.

7. Restart the OpenAM server to complete the external CTS store configuration. If there is a problem with the configuration, view the error in the `<openam-installation>/debug/Session` debug log, which will indicate that the External CTS connection could not be established.

   You have successfully configured an external OpenDJ directory server and configured OpenAM to use the external CTS store. For more CTS configuration options, see Section 7.6, “Managing CTS Tokens”.

7.5 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:n \
 --no-prompt \
 --trustStorePath /path/to/truststore
```

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

```
$ backup \
 --port 4444 \
 --bindDN "cn="Directory Manager" \
 --bindPassword password \
 --backendID ctsData \
 --backupDirectory /path/to/opendj/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.

7.6 Managing CTS Tokens

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

`com.sun.identity.session.repository.enableEncryption`
: Supports encryption of CTS tokens. Default: `false`.

`com.sun.identity.session.repository.enableCompression`
: Enables GZip-based compression of CTS tokens. Default: `false`.

`com.sun.identity.session.repository.enableAttributeCompression`
: Supports compression over and above the GZip-based compression of CTS tokens. Default: `false`.

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

Warning

When encryption or compression properties are changed, all previous tokens in the LDAP store will be un-readable; thus, invalidating any user's sessions. The user will be required to log in again.

7.7 CTS Tuning Considerations

CTS processes all requests asynchronously in the background, allowing callers (that is, those entities that call CTS) to send subsequent requests without waiting for a previous request to finish processing. The following OpenAM components make CTS requests:

1. Session service for session failover
2. OAuth 2.0 for token persistence
3. SAML 2.0 for token persistence
4. REST API for functions like Forgotten passwords

All create, read, update, delete, query (CRUDQ) requests to CTS are placed into an asynchronous buffer before being handled by an asynchronous processor. This ensures the caller can continue without waiting for CTS to complete processing.

Once the queue is full, all operations are "blocked" before an operation can be placed on the queue. Once on the queue, the caller can continue as normal.

CTS is designed to automatically throttle throughput in the event that the buffer fills up with requests. Therefore, if you require a balance between performance versus system memory, OpenAM provides two properties that can be used to tune CTS: queue size and queue timeout.

`org.forgerock.services.cts.async.queue.size`
: Default size: 5000. Determines how many request operations can be buffered before the queue size becomes full, after which the caller will be required to wait for the buffered requests to complete processing. All CRUDQ operations are converted to tasks, which are placed on the queue, ensuring that operations happen in the correct sequence.

`org.forgerock.services.cts.async.queue.timeout`
: Default timeout is 120 seconds. Determines how long a caller will wait in the even that the buffer is full. If the timeout expires, the caller receives an error. The timeout property is used in any system configuration where the LDAP server throughput is considerably slower than the OpenAM server, which can result in blocked requests as the backlog increases.

To set the queue size and timeout properties, navigate to the OpenAM Console: Configuration > Servers and Sites > Default Server Settings > Advanced, and then click Add.

7.8 General Recommendations for CTS Configuration

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:

- **First Look at Embedded**. Start your implementation, if possible, with the CTS options available with the OpenDJ instance embedded in OpenAM, which is the simplest deployment option to implement. 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.

  Note that the embedded CTS store is generally not recommended for high volume deployments and assumes a small scale deployment with a relatively simple topology.

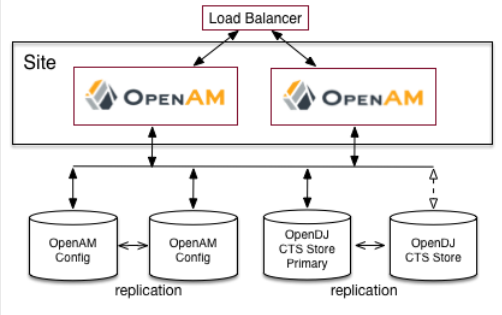
- **Isolate the Different Stores**. Isolate the user, configuration, and session stores from OpenAM into separate external OpenDJ servers.

- **Configure External CTS Stores for High Volumes**. 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 as CTS generally causes much more replication traffic than less volatile configuration data. Alternatively, if you need global replication of session, SAML 2.0, and OAuth 2.0 tokens, this would also justify a move to dedicated systems as it provides an extra level of control over how much replication is taking place.

On the Admin Console, specify the main external OpenDJ directory server for the CTS store and designate additional OpenDJ instances for session failover using the `Connection String(s)` property. This property allows you to configure multiple OpenDJ directory servers for your CTS data stores without a load balancer.

Note

The CTS requires that it talks to a single server or load balancer to reduce the risk of replication errors.



To improve performance, ensure that you have properly-sized directory servers for your external CTS stores. In addition, you can enable token compression as discussed in Section 7.6, “Managing CTS Tokens”. When enabled, token compression reduces load requirements on the network connection between data stores in exchange for processing time compressing tokens.

- **Prevent random or round-robin load balancing to external CTS stores.** No load balancer is needed between OpenAM and an external CTS store. Instead configure multiple directory server replicas in the Connection Strings of the configuration. For details, see Section 7.1, “CTS Configuration Parameters”.

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

  For complex deployments you might opt for an external directory service as the CTS store with a load balancer between OpenAM and the directory service. In this case, the choice of load balancing algorithm is important to ensure consistency under load within the CTS layer. High loads with a round-robin

or random algorithm cause replication conflicts within the CTS layer, which the CTS layer is unable to resolve. The load balancer *must* operate only for failover, and not to balance LDAP traffic between directory servers. In other words, the load balancer in front of the external CTS store *must* use an Active/Passive configuration, whereby the load balancer sends all requests to the same directory server until that server becomes unavailable, and then all requests go to the standby directory server, and so on. Load balancers *must not* use an Active/Active configuration, as this leads to the type of errors described above.

Once configured, the OpenDJ directory service replicates CTS data transmitted from OpenAM servers to connected OpenDJ servers. The amount of replication traffic can be significant, especially if replication proceeds over a WAN. You can limit this replication traffic by separating OpenDJ instances into directory and replication servers. For more information on how this is done with OpenDJ, see the OpenDJ documentation on *Stand-alone Replication Servers*.

Chapter 8

Setting Up OpenAM Session Failover

This chapter covers setting up session failover (SFO). Session failover allows another OpenAM server to manage a session when the server that initially authenticated the user is down. This means the user does not need to log in again, even though the server that authenticated them is down.

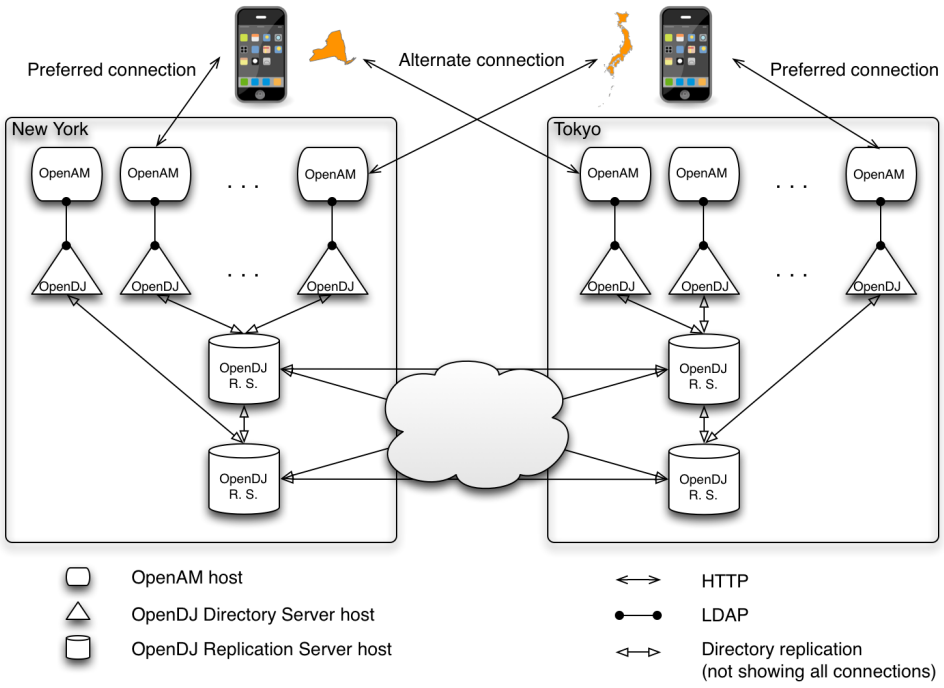
Session failover (high availability for sessions) builds on OpenAM service availability. Before configuring session failover, you must therefore first make the overall OpenAM service highly available. This is done by setting up OpenAM in a site configuration. You can find instructions for setting up a site configuration in the chapter, *Installing Multiple Servers*.

Session failover also relies on a shared Core Token Service (CTS) to store user session data. The service is shared with other OpenAM servers in the same OpenAM Site. When an OpenAM server goes down, other servers in the Site can read user session information from the CTS, so the user with a valid session does not have to log in 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. By default the Core Token Service uses the embedded OpenDJ directory server. For more information on configuring the Core Token Service, see the chapter, *Configuring the Core Token Service (CTS)*.

In deployments with multiple OpenAM Sites, session failover can function across Sites. In order for this to work, all Sites must use the same, global, underlying Core Token Service, which is replicated across all Sites. Then when an entire Site fails or becomes unavailable, OpenAM servers in another Site detect the failure of the Site's load balancer and attempt to recover the user session from the global Core Token Service.

In the event of a failure, client applications can connect to an OpenAM server in an active data center as shown in Figure 8.1, “Core Token Service For Global Session Failover”.

Figure 8.1. Core Token Service For Global Session Failover



For more information on how this is done with OpenDJ directory server, see the OpenDJ documentation on *Managing Data Replication*.

Procedure 8.1. To Configure Session Failover After Installation

Session failover requires an OpenAM Site configuration with a Core Token Service.

If you did not configure session persistence and availability during initial configuration, first complete the steps in the procedure, *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 the configuration server does not support the Core Token Service, 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. To activate the Session Persistence and High Availability Failover option, check the Enabled box.

5. To ensure that local OpenAM instances resolve sessions from the Core Token Service session store rather than by using crosstalk, check the Reduce Crosstalk Enabled box. For more information about crosstalk, see the section, *To Configure Site Load Balancing*.

   Do not disable reduced crosstalk unless advised to do so by ForgeRock Technical Support.

6. Set reduced crosstalk options.

   Session logout/destroy broadcasting enables notification to all servers in an OpenAM site when a user logs out or her session is destroyed by the OpenAM server. The broadcast notifications are in addition to normal session logout/destroy notifications sent to interested clients and servers.

   Without session logout/destroy broadcasting, it is possible for a user to log out from one OpenAM server and then access her session on another server during the brief window between the logout and session store replication. Enabling session logout/destroy broadcasting ensures that logged out and destroyed sessions have the correct state on all OpenAM servers.

   - Select Disabled if you do not want the OpenAM server to broadcast session logout/destroy messages. Session logout/destroy broadcasting is disabled by default. Disabling broadcasting is suitable when you do not need the highest level of security. Disable broadcasting when you do not expect users to maliciously attempt to access logged out or destroyed sessions.

   - Specify one of the two broadcast options to achieve a higher level of security, at a cost of incurring additional network I/O. Select "Broadcast only to local site servers" if your session store supports a single OpenAM site. Select "Broadcast to servers in all sites" if your session store supports multiple OpenAM sites.

   The Reduced Crosstalk Purge Delay option specifies the amount of time (in minutes) before sessions are purged from OpenAM servers after the server receives session logout/destroy broadcast notification. The delay ensures that sessions are in memory during the time between session logout/destruction and session store replication.

The default purge delay is 1 minute, which should be adequate unless session store replication is abnormally slow on your network.

7. Click Add to save your work.
8. Restart all the OpenAM servers in the site.

Chapter 9

Removing OpenAM Software

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

Procedure 9.1. 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 `dc=openam,dc=forgerock,dc=org`.

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

M

O

P

S

U

V