Installation Guide

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

Table of Contents

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

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.

Who Should Use This Guide

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

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

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.

Formatting Conventions

Most examples in the documentation are created in GNU/Linux or Mac OS X operating environments. If distinctions are necessary between operating environments, examples are labeled with the operating environment name in parentheses. To avoid repetition file system directory names are often given only in UNIX format as in `/path/to/server` , even if the text applies to `C:\path\to\server` as well. Absolute path names usually begin with the placeholder `/path/to/` . This path might translate to `/opt/` , `C:\Program Files\` , or somewhere else on your system. Command-line, terminal sessions are formatted as follows:

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

Command output is sometimes formatted for narrower, more readable output even though formatting parameters are not shown in the command. Program listings are formatted as follows:

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

Accessing Documentation Online

Open Identity Platform Community publishes comprehensive documentation online:

- The Open Identity Platform Community Documentation offers a large and increasing number of up-to-date, practical articles that help you deploy and manage Open Identity Platform software.
- Open Identity Platform product documentation, such as this document, aims to be technically accurate and complete with respect to the software documented. It is visible to everyone and covers all product features and examples of how to use them.

Joining the Open Identity Platform Community

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

Getting Support and the Contacting Open Identity Platform Community

Open Identity Platform Community Approved Vendors provide support services, professional services, trainings, and partner services to assist you in setting up and maintaining your deployments.

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 servers 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.

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

IMPORTANT	Do not configure a top-level domain as your cookie domain as browsers will reject them. Top-level domains are browser-specific. Some browsers, like Firefox, also consider special domains like Amazon's web service (for example, ap-southeast-2.compute.amazonaws.com) to be a top-level domain. Check the effective top-level domain list at https://publicsuffix.org/list/effective_tld_names.dat to ensure that you do not set your cookie to a domain in the list.

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. The current OpenAM release supports Java Development Kit 8, 11, 17 or 21 LTS version.

Settings For Sun/Oracle Java Environments

When using an Adoptium, Azul or Oracle Java environment set at least the following options.

`-server`

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

IMPORTANT

If you are using Java 11 and above, add additional options to provide OpenAM access to the encapsulated Java modules:

```
--add-exports java.base/sun.security.util=ALL-UNNAMED
--add-exports java.security.jgss/sun.security.krb5=ALL-UNNAMED
--add-exports java.base/sun.security.x509=ALL-UNNAMED
--add-exports java.base/sun.security.tools.keytool=ALL-UNNAMED
--add-exports java.xml/com.sun.org.apache.xerces.internal.dom=ALL-UNNAMED
```

For additional JVM tuning and security recommendations, see Java Virtual Machine Settings in the *Administration Guide*.

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.

Setting Maximum File Descriptors

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

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

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

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

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

```
$ ulimit -n
65536
```

You can check the Linux system overall maximum as follows.

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

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

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

   ```
   LimitNOFILE=65536
   ```

5. To reload the daemon, run:

   ```
   $ systemctl daemon-reload
   ```

6. Restart Tomcat:

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

7. Check the file descriptors:

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

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

Preparing an External Identity Repository

OpenAM accesses user identity data from one or more identity repositories. OpenAM ships with an embedded OpenDJ directory server that you can install as part of the OpenAM configuration process. You can use the embedded directory server as your identity repository for smaller deployments, and avoid the additional overhead of maintaining a separate directory server.

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

If you are configuring OpenAM to share data with other applications, or if you expect your deployment will have a large amount of users, connect OpenAM to an external identity repository.

Important Considerations for Using External Identity Repositories

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

Specifying the directory administrator, for example, `cn=Directory Manager` as the OpenAM data store administrator is not recommended for production deployments as it will give OpenAM directory administrator privileges to the identity repository. Instead, create a separate OpenAM administrator account with fewer access privileges than the directory administrator so that you can assign the appropriate level of privileges for the OpenAM data store administrator.

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

Schema Update Privileges

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

Directory Read and Write Access Privileges

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

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

Preparing Your External Identity Repository

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

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

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

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

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

Preparing an Identity Repository With Dynamic Schema Updates

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

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

To Prepare an External OpenDJ Identity Repository with Dynamic Schema Updates

1. Create the OpenAM data store administrator account.

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

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

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

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

   dn: uid=openam,ou=admins,dc=example,dc=com
   objectClass: top
   objectClass: person
   objectClass: organizationalPerson
   objectClass: inetOrgPerson
   ```

```
cn: OpenAM Administrator
sn: OpenAM
userPassword: changeMe
ds-privilege-name: update-schema
ds-privilege-name: subentry-write
ds-privilege-name: password-reset
```

Then, run the `ldapmodify` command to create the user.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="* || aci")(version 3.0;acl "Allow identity modification";
   allow (write)(userdn = "ldap:///uid=openam,ou=admins,dc=example,dc=com");)
aci: (targetattr!="userPassword||authPassword")(version 3.0;
   acl "Allow identity search"; allow (search, read)(userdn = "ldap:///
   uid=openam,ou=admins,dc=example,dc=com");)
aci: (targetcontrol="2.16.840.1.113730.3.4.3")(version 3.0;acl "Allow
   persistent search"; allow (search, read)(userdn = "ldap:///
   uid=openam,ou=admins,dc=example,dc=com");)
aci: (version 3.0;acl "Add or delete identities"; allow (add, delete)
   (userdn = "ldap:///uid=openam,ou=admins,dc=example,dc=com");)
aci: (targetcontrol="1.3.6.1.4.1.42.2.27.8.5.1")(version 3.0;acl "Allow behera
draft control"; allow (read)
   (userdn = "ldap:///uid=openam,ou=admins,dc=example,dc=com");)
```

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

```
dn: dc=example,dc=com
```

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

Then run the `ldapmodify` command to implement the ACIs:

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

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

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

Preparing an Identity Repository With Manual Schema Updates

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

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

To Prepare an External OpenDJ Identity Repository With Manual Schema Updates

1. Create the OpenAM data store administrator account.

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

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

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

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

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

Then run the `ldapmodify` command to create the user:

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

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

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

   First, identify the path that contains LDIF file for OpenAM schema extensions. The path is `/path/to/openam/ldif/directory_type`, where `directory_type` is one of the following:

   - `ad` for Microsoft Active Directory
   - `adam` for Microsoft Active Directory Lightweight Directory Services
   - `odsee` for Oracle Directory Server Enterprise Edition

- `opendj` for OpenDJ and Oracle Unified Directory
- `tivoli` for IBM Tivoli Directory Server

Then run the `ldapmodify` command to import the user, device print, and dashboard schema extensions. For example, to add schema extensions for an OpenDJ directory server, run the following `ldapmodify` commands:

```
$ cd /path/to/openam/ldif/opendj

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

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

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

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

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

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

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

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

```
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="* || aci")(version 3.0;acl "Allow identity modification";
   allow (write)(userdn = "ldap:///uid=openam,ou=admins,dc=example,dc=com");)
aci: (targetattr!="userPassword||authPassword")(version 3.0;
   acl "Allow identity search"; allow (search, read)(userdn = "ldap:///
   uid=openam,ou=admins,dc=example,dc=com");)
aci: (targetcontrol="2.16.840.1.113730.3.4.3")(version 3.0;acl "Allow
   persistent search"; allow (search, read)(userdn = "ldap:///
   uid=openam,ou=admins,dc=example,dc=com");)
aci: (version 3.0;acl "Add or delete identities"; allow (add, delete)
   (userdn = "ldap:///uid=openam,ou=admins,dc=example,dc=com");)
aci: (targetcontrol="1.3.6.1.4.1.42.2.27.8.5.1")(version 3.0;acl "Allow behera
draft control"; allow (read)
   (userdn = "ldap:///uid=openam,ou=admins,dc=example,dc=com");)
```

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

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

Then run the `ldapmodify` command to implement the ACIs:

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

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

Configuring OpenAM Data Stores That Access External Identity Repositories

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

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

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

  For more information about running the OpenAM configurator, see "Installing OpenAM Core Services".

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

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

  For more information about defining OpenAM data stores, see "Configuring Data Stores" in the *Administration Guide*.

Indexing External Identity Repositories Attributes

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

To Index External Identity Repository Attributes

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

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

  - Use the following commands with OpenDJ 2.6:

```
$ dsconfig \
 create-local-db-index \
 --port 4444 \
 --hostname opendj.example.com \
 --bindDN "cn=Directory Manager" \
 --bindPassword password \
 --backend-name userRoot \
 --index-name iplanet-am-user-federation-info-key \
```

```
 --set index-type:equality \
 --no-prompt

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

- Use the following commands with OpenDJ 3 and later:

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

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

Testing External Identity Repository Access from OpenAM

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

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

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

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

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

Preparing an External Configuration Data Store

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

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

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

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

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

To Install an External OpenDJ Directory Server

The following example procedure shows how to prepare a single OpenDJ directory server instance as an external configuration data store. The OpenDJ instance implements a single

backend for the OpenAM configuration data. The procedure assumes that you have also prepared an external identity repository and an external CTS store, separate from the configuration data store.

NOTE

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

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

1. Prepare your OpenDJ installation, then download the OpenDJ software. See the OpenDJ documentation about Installing OpenDJ Servers.

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

Example options are as follows:

Example OpenDJ Setup Parameters

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

2. Change to the OpenDJ directory.

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

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

The `dsconfig` command used to create the backend depends on the version of OpenDJ

directory server.

- Use the following command with OpenDJ 2.6:

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

- Use the following command with OpenDJ 3 and later, where the value of the `--type` option depends on the backend database type to use, such as `je` or `pdb`. This example creates a JE backend:

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

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

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

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

```
dn: dc=example,dc=com
objectclass: top
objectclass: domain
```

```
dc: example
aci: (targetattr="*")(version 3.0;acl "Allow CRUDQ operations";
 allow (search, read, write, add, delete)
 (userdn = "ldap:///uid=openam,ou=admins,dc=example,dc=com");)
aci: (targetcontrol="2.16.840.1.113730.3.4.3")(version 3.0;acl "Allow
 persistent search"; allow (search, read)(userdn = "ldap:///uid=openam
 ,ou=admins,dc=example,dc=com");)
aci: (targetcontrol="1.2.840.113556.1.4.473")(version 3.0;acl "Allow
 server-side sorting"; allow (read)(userdn = "ldap:///
 uid=openam,ou=admins,dc=example,dc=com");)

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

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

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

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

   ```
   $ bin/ldapmodify \
   --port 1389 \
   --bindDN "cn=Directory Manager" \
   --bindPassword password \
   --defaultAdd \
   --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 copy the text from the following example, make sure that the value of `global-aci` is all on a single line.

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

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

Configuration Data Store Attributes

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

Add the required CTS schema definitions. You can find the CTS schema definitions at

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

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

9. Add the schema file to the directory server.

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

10. Add the required user store schema definitions. You can find the schema definitions at /path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opendj/opendj_user_schema.ldif.

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

11. Add the schema file to the directory server.

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

12. Add the schema definitions to the configuration repository. You can find the schema definitions at /path/to/tomcat/webapps/openam/WEB-INF/template/ldif/opendj/opendj_config_schema.ldif.

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

13. Add the schema file to the directory server.

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

14. OpenAM uses the attributes in "Configuration Data Store Attributes" to search for

configuration data. On the OpenDJ directory server, use the `dsconfig` command to add these indexes to your external configuration store. Repeat this step to index the `iplanet-am-user-federation-info-key` and `sun-fm-saml2-nameid-infokey` attributes if you are deploying federation.

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

- Use the following commands with OpenDJ 2.6:

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

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

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

- Use the following commands with OpenDJ 3 and later:

```
$ dsconfig create-backend-index \
--port 4444 \
--hostname opendj.example.com \
--bindDN "cn=Directory Manager" \
--bindPassword password \
--backend-name cfgStore \
```

```
--index-name sunxmlkeyvalue \
--set index-type:equality \
--set index-type:substring \
--no-prompt

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

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

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

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

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

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

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

Preventing Anonymous Access to an External Configuration Store

By default, OpenDJ allows unauthenticated or anonymous connections to directory servers. For external configuration stores, this default is a security vulnerability.

In production deployments, you want to allow unauthenticated connections to the root entry only, so that LDAP clients can obtain server information for the OpenDJ server, while at the same time, denying anonymous connections to all directory server instances.

You can prevent anonymous access from LDAP clients to the OpenDJ server while allowing unauthenticated access to the root entry by configuring access control instructions (ACIs) and removing global ACIs from the directory server instances.

To Prevent Anonymous Access in External Configuration Stores

1. To allow unauthenticated access to the OpenDJ root entry, set the `global-aci` using the `dsconfig` command:

```
$ ./dsconfig set-access-control-handler-prop --add 'global-
aci:(target="ldap:///")(targetscope="base")(targetattr="objectClass||namingCont
exts||supportedAuthPasswordSchemes||supportedControl||supportedExtension||suppo
rtedFeatures||supportedLDAPVersion||supportedSASLMechanisms||vendorName||vendor
Version")(version 3.0; acl "User-Visible Root DSE Operational Attributes";
allow (read,search,compare) userdn="ldap:///anyone";)' --hostname
server.example.com --port 4444 --bindDN "cn=Directory Manager" --bindPassword
password --trustAll --no-prompt
```

2. Remove the default access to anonymous users using the `dsconfig` command:
   - On OpenDJ 3.x or later:

```
$ ./dsconfig set-access-control-handler-prop --remove 'global-
aci:(targetattr!="userPassword||authPassword||debugsearchindex||changes||ch
angeNumber||changeType||changeTime||targetDN||newRDN||newSuperior||deleteOl
dRDN")(version 3.0; acl "Anonymous read access"; allow
(read,search,compare) userdn="ldap:///anyone";)' --hostname
server.example.com --port 4444 --bindDN "cn=Directory Manager"
--bindPassword password --trustAll --no-prompt
```

   - On OpenDJ 2.6.x:

```
$ ./dsconfig set-access-control-handler-prop --remove 'global-
aci:(targetattr!="userPassword||authPassword||changes||changeNumber||change
Type||changeTime||targetDN||newRDN||newSuperior||deleteOldRDN")(version
3.0; acl "Anonymous read access"; allow (read,search,compare)
userdn="ldap:///anyone";)' --hostname server.example.com --port 4444
--bindDN "cn=Directory Manager" --bindPassword password --trustAll --no
-prompt
```

3. If you have other custom ACIs that affect anonymous users, review them and update as necessary. To generate a list of ACIs, run the following command:

```
$ ./ldapsearch --hostname openam.example.com --port 1389 --bindDN "cn=Directory
manager" \
  --bindPassword "password" --baseDN "cn=config" --searchScope sub "cn=Access
Control Handler"
```

4. Repeat these steps for all appropriate OpenDJ instances.

For additional information, see How do I prevent anonymous access in DS/OpenDJ (All version) in the *ForgeRock Knowledge Base*.

Obtaining OpenAM Software

Download OpenAM releases from the releases page on the GitHub.

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-OpenAM 15.1.7-SNAPSHOT.zip
```

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

`ClientSDK-OpenAM 15.1.7-SNAPSHOT.jar`

The OpenAM Java client SDK library

`ExampleClientSDK-CLI-OpenAM 15.1.7-SNAPSHOT.zip`

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

`ExampleClientSDK-WAR-OpenAM 15.1.7-SNAPSHOT.war`

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

`IDPDiscovery-OpenAM 15.1.7-SNAPSHOT.war`

The IDP discovery `.war` file, deployed as a service to service providers that must discover which identity provider corresponds to a SAML v2.0 request.

For details, see "Deploying the Identity Provider Discovery Service" in the *Administration Guide*.

`Fedlet-OpenAM 15.1.7-SNAPSHOT.zip`

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

`OpenAM-OpenAM 15.1.7-SNAPSHOT.war`

The deployable `.war` file.

`SSOAdminTools-OpenAM 15.1.7-SNAPSHOT.zip`

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

`SSOConfiguratorTools-OpenAM 15.1.7-SNAPSHOT.zip`

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

`openam-soap-sts-server-OpenAM 15.1.7-SNAPSHOT.war`

A pre-built SOAP STS server `.war` file.

For details, see "Deploying SOAP STS Instances" in the *Administration Guide*.

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-OpenAM 15.1.7-SNAPSHOT.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 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 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.

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

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`.

Enabling RSA SecurID Support

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

1. Unpack the OpenAM .war file.

```
$ mkdir /tmp/openam
$ cd /tmp/openam/
$ jar -xf ~/Downloads/openam/OpenAM-OpenAM 15.1.7-SNAPSHOT.war
```

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

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

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

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

4. Deploy the new .war file. See "To Deploy OpenAM".

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

Preparing Apache Tomcat

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

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 "Preparing a Java Environment" for details.

Open Identity Platform Community 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.

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

<Connector> configuration elements are found in the configuration file, `/path/to/tomcat/conf/server.xml`. The following excerpt shows an example <Connector> with the `URIEncoding` 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` and that you use Oracle JDK 8.

```
#!/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"
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
```

Slashes in Resource Names

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

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

```
CATALINA_OPTS= "-server \
              -Dorg.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH=true"
```

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

Cookie Domains

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

Tuning Apache Multi-Processing Modules

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

for a particular site.

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

Preparing OpenAM for JBoss and WildFly

You can deploy OpenAM on JBoss AS, JBoss EAP, and WildFly. Some preparation is required to deploy on these application servers.

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

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

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

To Prepare JBoss or WildFly for OpenAM

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

documentation

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

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

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

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

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

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

To Prepare OpenAM for JBoss and WildFly

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

1. If you have not already done so, create a temporary directory and expand the `OpenAM-OpenAM 15.1.7-SNAPSHOT.war` file.

```
$ cd /tmp
$ mkdir /tmp/openam ; cd /tmp/openam
$ jar xvf ~/Downloads/OpenAM-OpenAM 15.1.7-SNAPSHOT.war
```

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

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

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

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

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

4. Rebuild the `openam.war` file.

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

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

Preparing Oracle WebLogic

To deploy OpenAM in WebLogic, perform the following steps:

1. Update the JVM options as described in "Preparing a Java Environment".
2. Customize the `OpenAM-OpenAM 15.1.7-SNAPSHOT.war` file as described in "To Prepare OpenAM for Oracle WebLogic".

To Prepare OpenAM for Oracle WebLogic

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

Change the OpenAM `war` file as follows:

1. Create a temporary directory and expand the `OpenAM-OpenAM 15.1.7-SNAPSHOT.war` file:

```
$ cd /tmp
$ mkdir /tmp/openam ; cd /tmp/openam
$ jar xvf ~/Downloads/OpenAM-OpenAM 15.1.7-SNAPSHOT.war
```

2. Locate the `bootstrap.properties` file in the `WEB-INF/classes` directory of the expanded `war` file.
3. Update the `# configuration.dir=` line in the `bootstrap.properties` file to specify a path with read and write permissions. For example:

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

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

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

4. Rebuild the `openam.war` file:

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

Preparing IBM WebSphere

Before you deploy OpenAM, use the Administrator console to update JVM options as described in "Preparing a Java Environment". In addition, configure WebSphere to load classes from OpenAM bundled libraries before loading classes from libraries delivered with WebSphere. The following steps must be completed after you deploy OpenAM into WebSphere.

1. In WebSphere administration console, browse to Application > Application Type > WebSphere enterprise applications > *OpenAM Name* > Class loading and update detection.
2. Set Class loader order > Classes loaded with local class loader first (parent last).
3. Ensure that the value of the *WAR class loader policy* property is set to the default value: `Class loader for each WAR file in application`.
4. Save your work.

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 (recommended) 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.

OpenAM Installation Options

Installation Action	Documentation Reference
Install quickly for evaluation using default settings	"To Deploy OpenAM" and "To Configure OpenAM With Defaults" Alternatively, follow the full example in Getting Started With OpenAM.
Install OpenAM server, choosing settings	"To Deploy OpenAM" and "To Custom Configure OpenAM"
Erase the configuration and start over	"To Delete an OpenAM Configuration Before Redeploying"
Add an OpenAM server to a site	"To Deploy OpenAM", and "To Add a Server to a Site"
Troubleshoot an OpenAM installation	"To Troubleshoot an OpenAM Installation"
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"
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.

To Deploy OpenAM

The `OpenAM-OpenAM 15.1.7-SNAPSHOT.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-OpenAM 15.1.7-SNAPSHOT.war /path/to/tomcat/webapps/openam.war
```

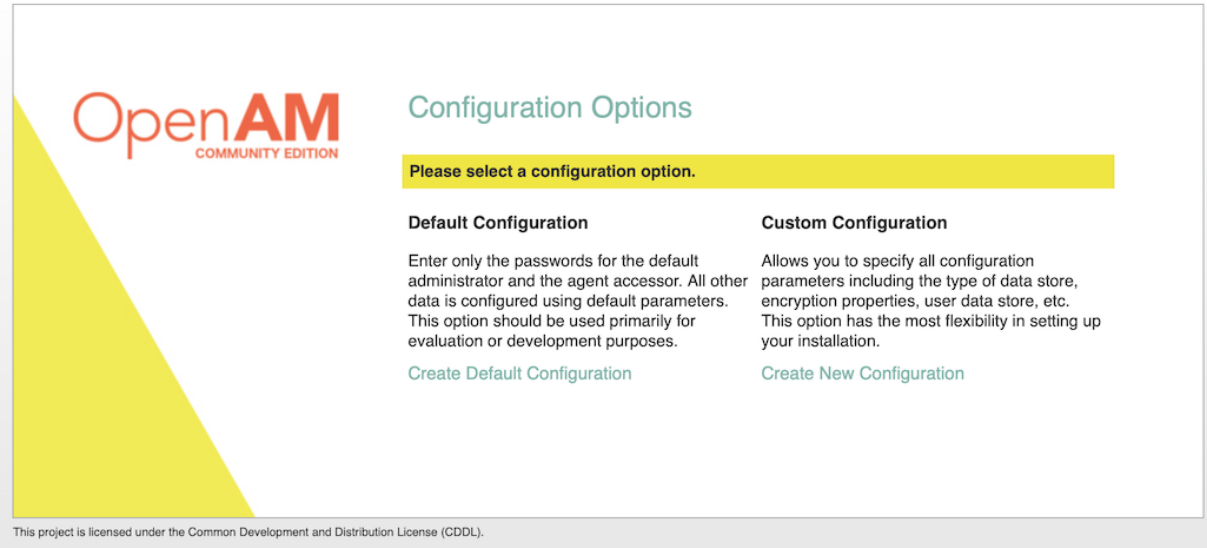
You change the file name to `openam.war` when deploying in Tomcat so that the deployment URI is `/openam`.

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

IMPORTANT	To properly configure OpenAM, OpenAM requires a deployment URI with a non-empty string after `/`. Do not deploy OpenAM at the root context. Do not rename the `.war` file to `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`.



To Configure OpenAM With Defaults

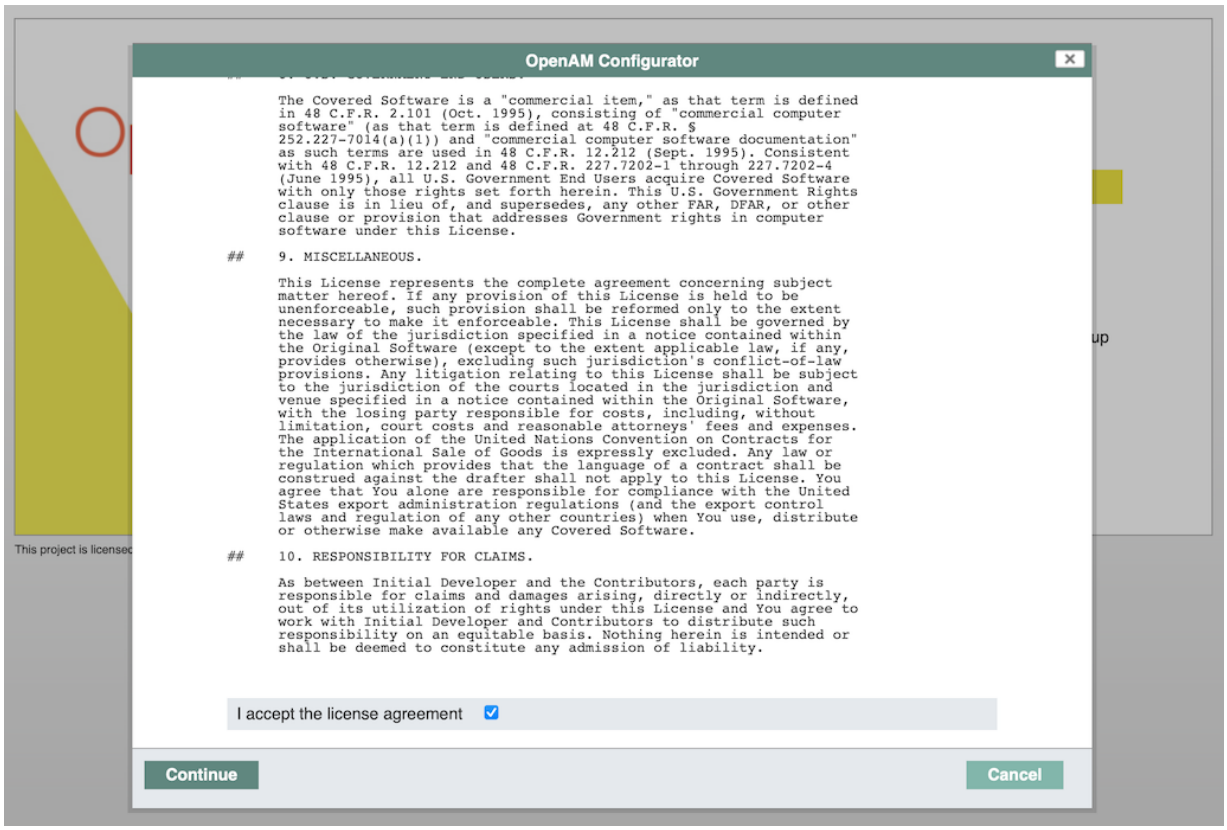
The default configuration option configures the embedded OpenDJ server using default ports. If the ports are already in use, OpenAM uses free ports as both configuration store and identity store.

The default configuration sets the cookie domain based on the full URL that was used to access the configurator, such as `example.com`, `server.west.example.com`, or `example.local`.

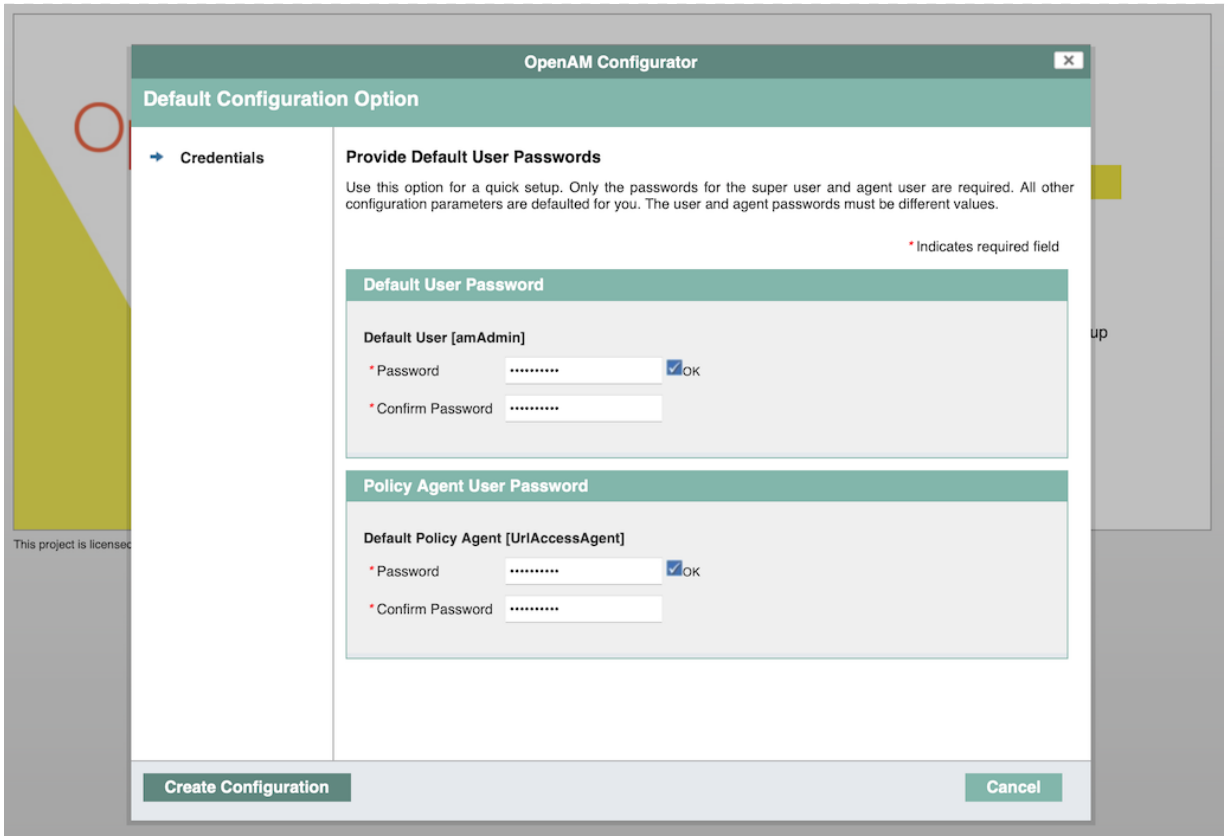
Configuration settings are saved to the home directory of the user running the web application container in a directory named after the deployment URI. In other words if OpenAM is

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.

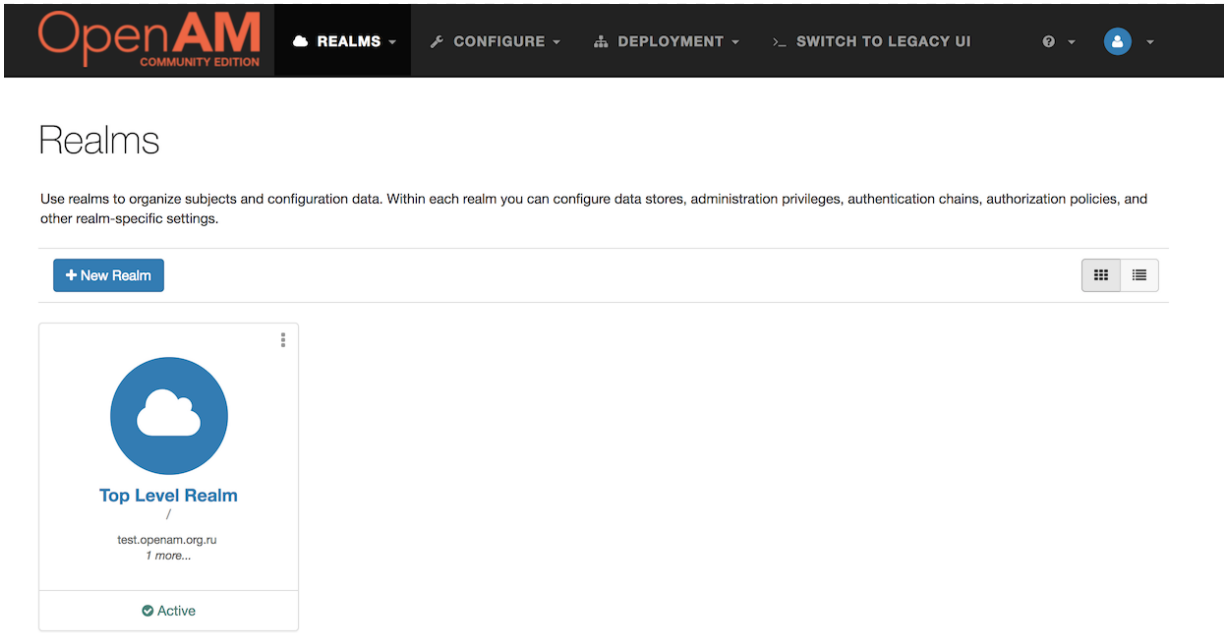


3. Provide different passwords for the default OpenAM administrator, `amadmin`, and default Policy Agent users.



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 Realms.



To Delete an OpenAM Configuration Before Redeploying

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

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

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

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

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

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

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

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

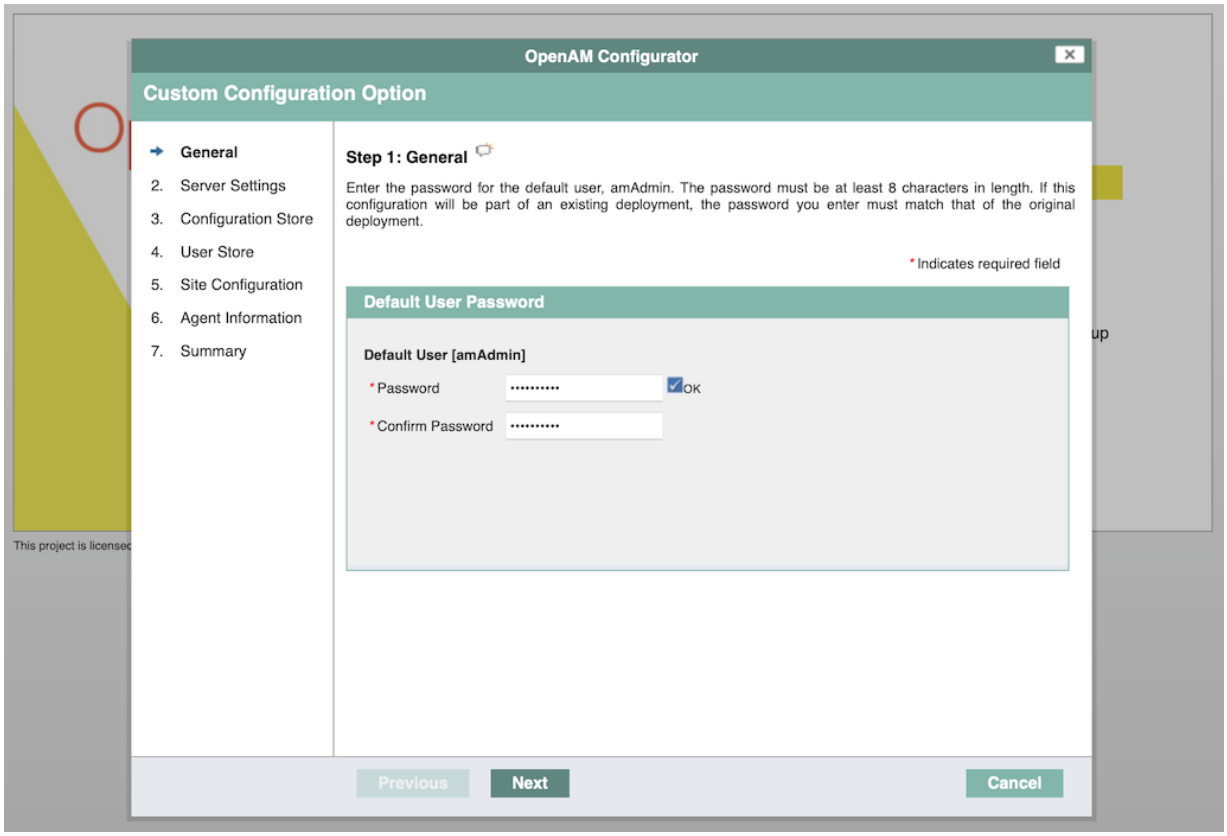
3. Restart the OpenAM web application.

The following example starts the Tomcat container.

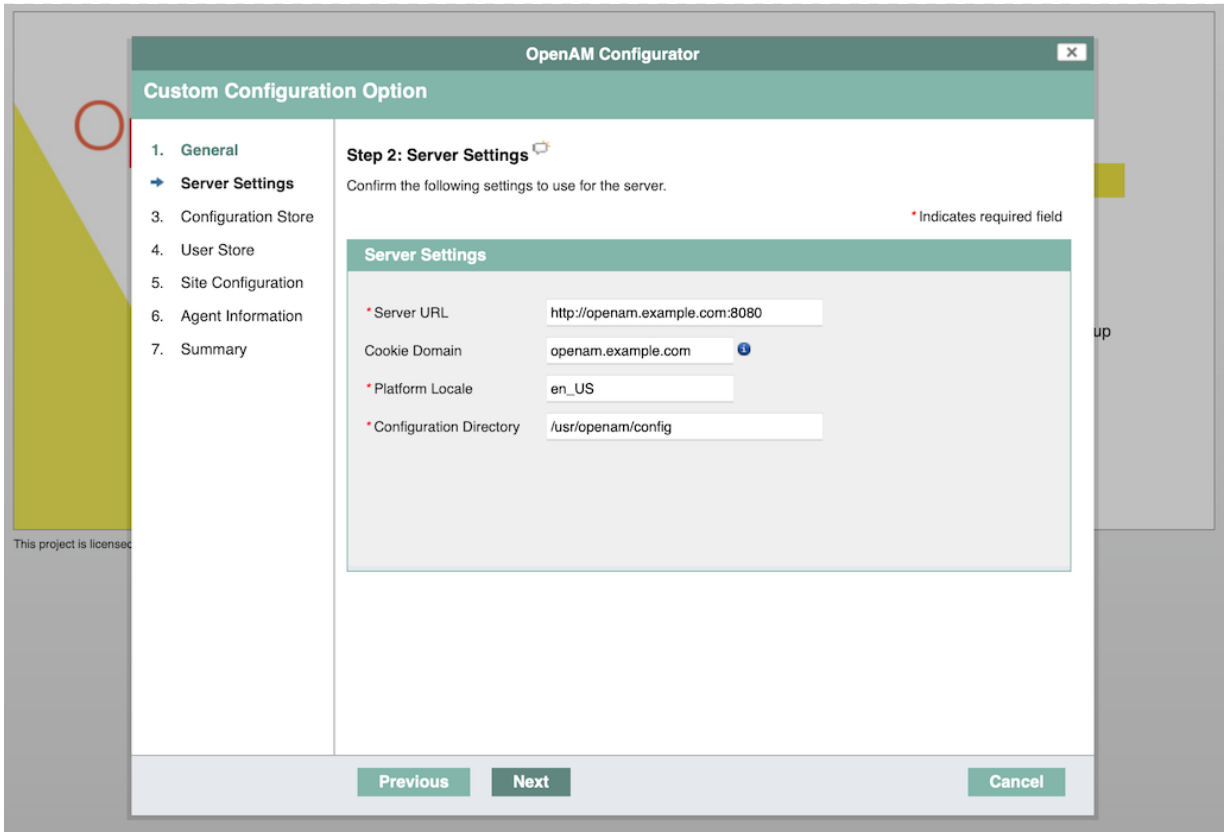
```
$ /path/to/tomcat/bin/startup.sh
Password:
Using CATALINA_BASE:   /path/to/tomcat
Using CATALINA_HOME:   /path/to/tomcat
Using CATALINA_TMPDIR: /path/to/tomcat/temp
Using JRE_HOME:        /path/to/jdk/jre
Using CLASSPATH:
       /path/to/tomcat/bin/bootstrap.jar:/path/to/tomcat/bin/tomcat-juli.jar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

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 with at least eight characters for the OpenAM Administrator, `amadmin`.



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



Server URL

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

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

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

Cookie Domain

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

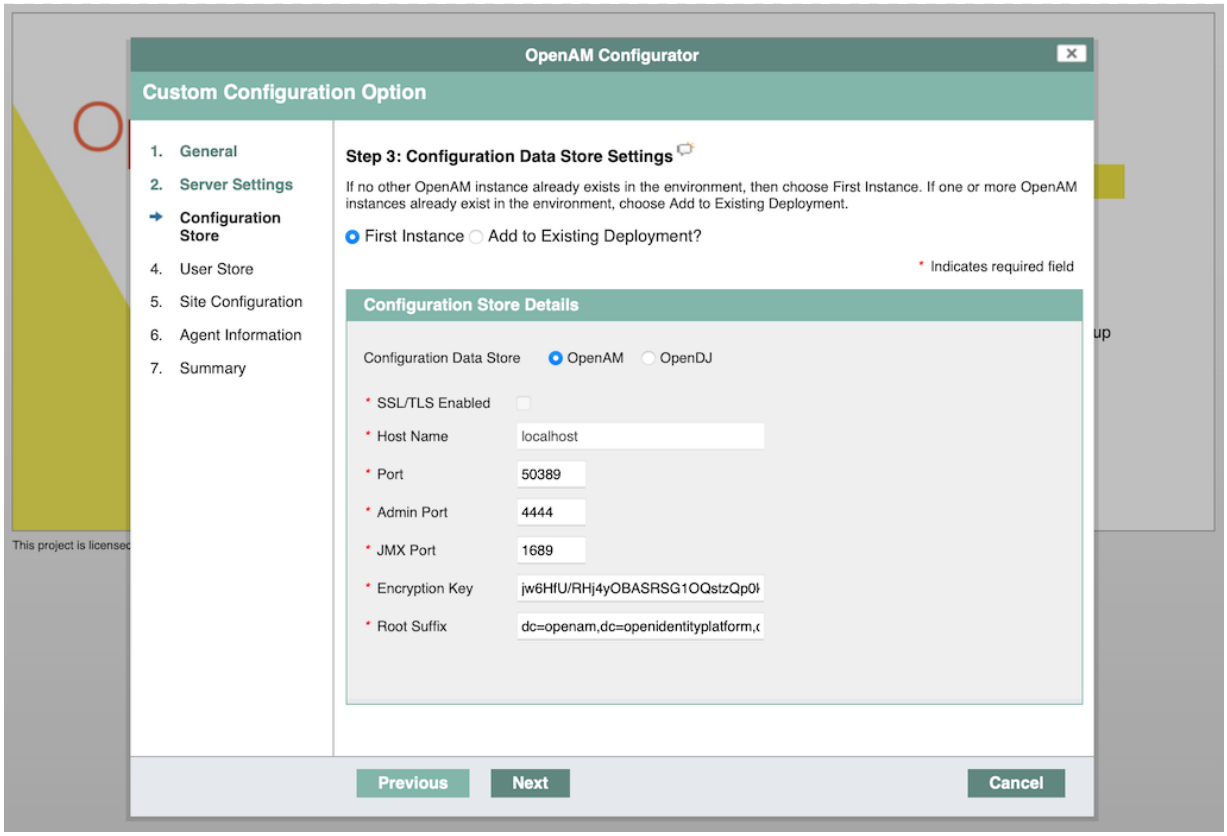
Platform Locale

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

Configuration Directory

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

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

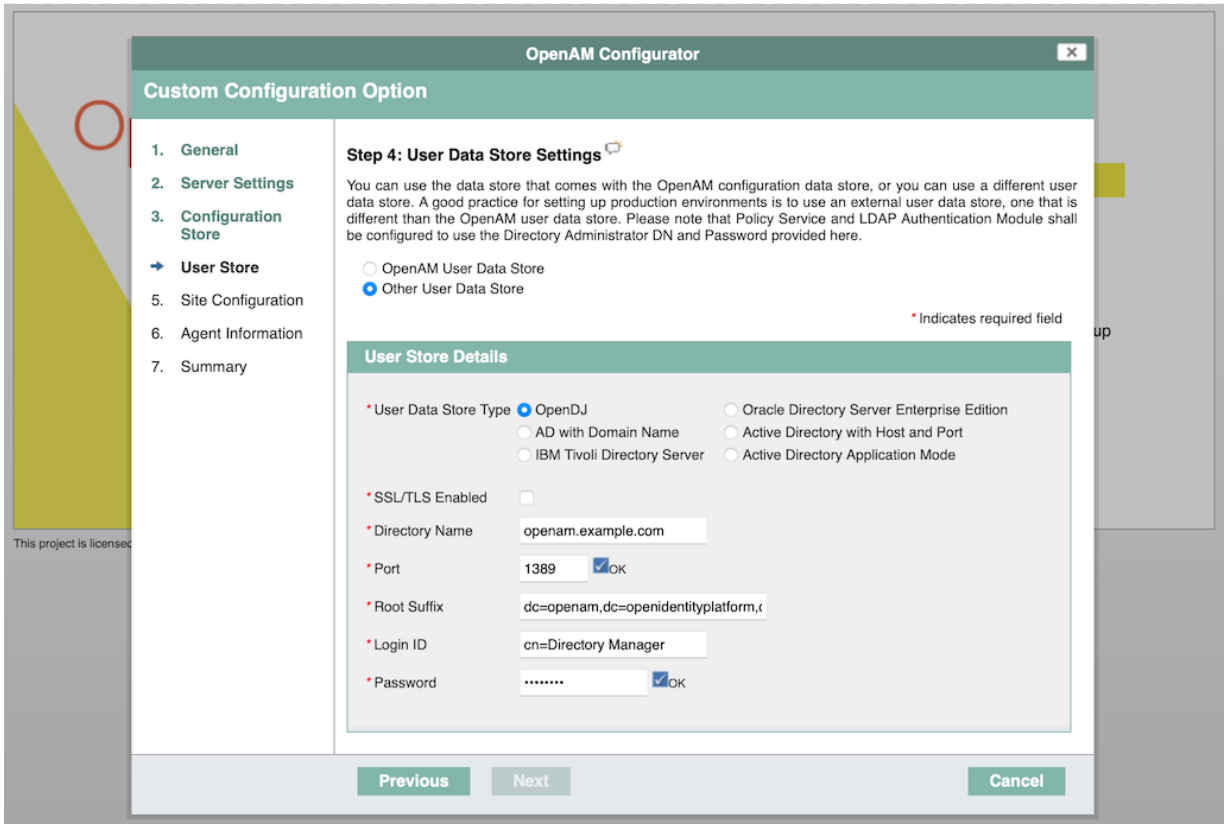


You can also add this OpenAM installation to an existing deployment, providing the URL of the site. See "To Add a Server to a Site" for details.

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

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

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



User Data Store Type

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

SSL/TLS Enabled

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

Directory Name

FQDN for the host housing the directory service.

Port

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

Root Suffix

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

Login ID

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

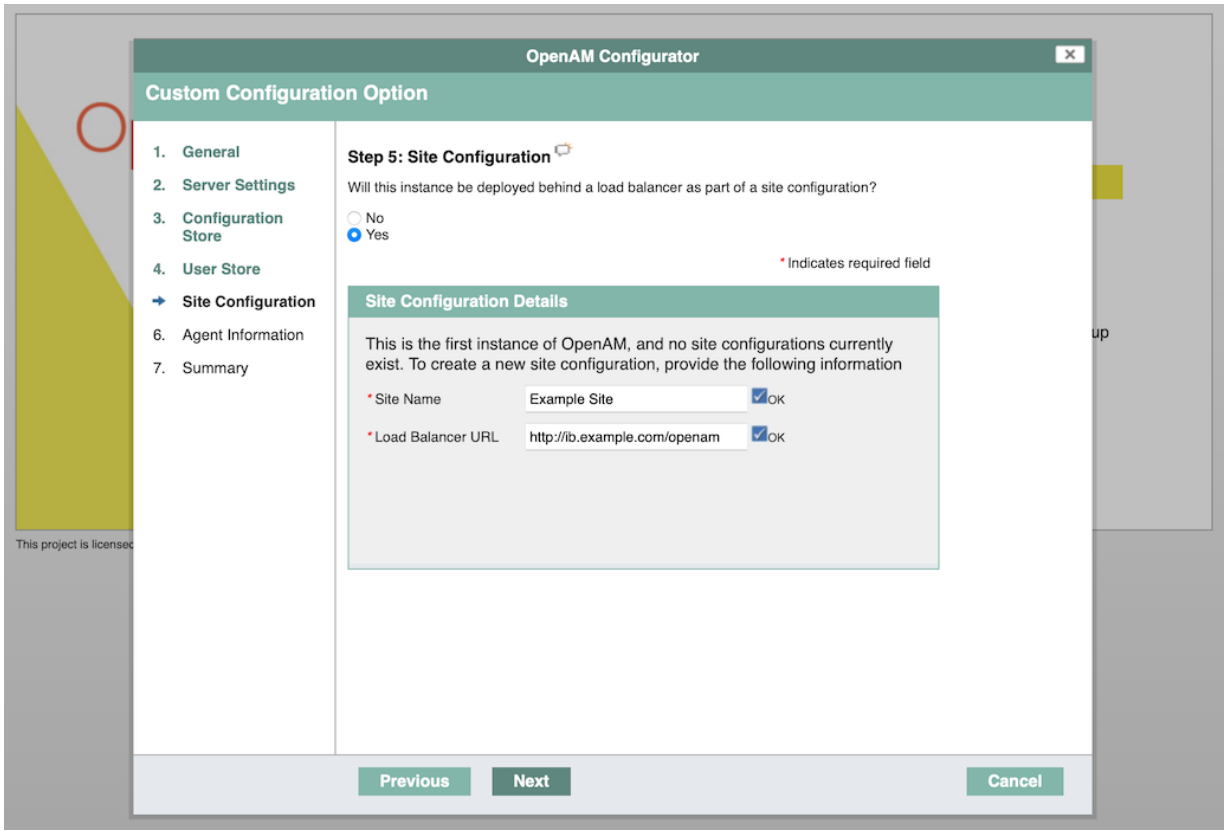
Password

Password for the directory administrator user.

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

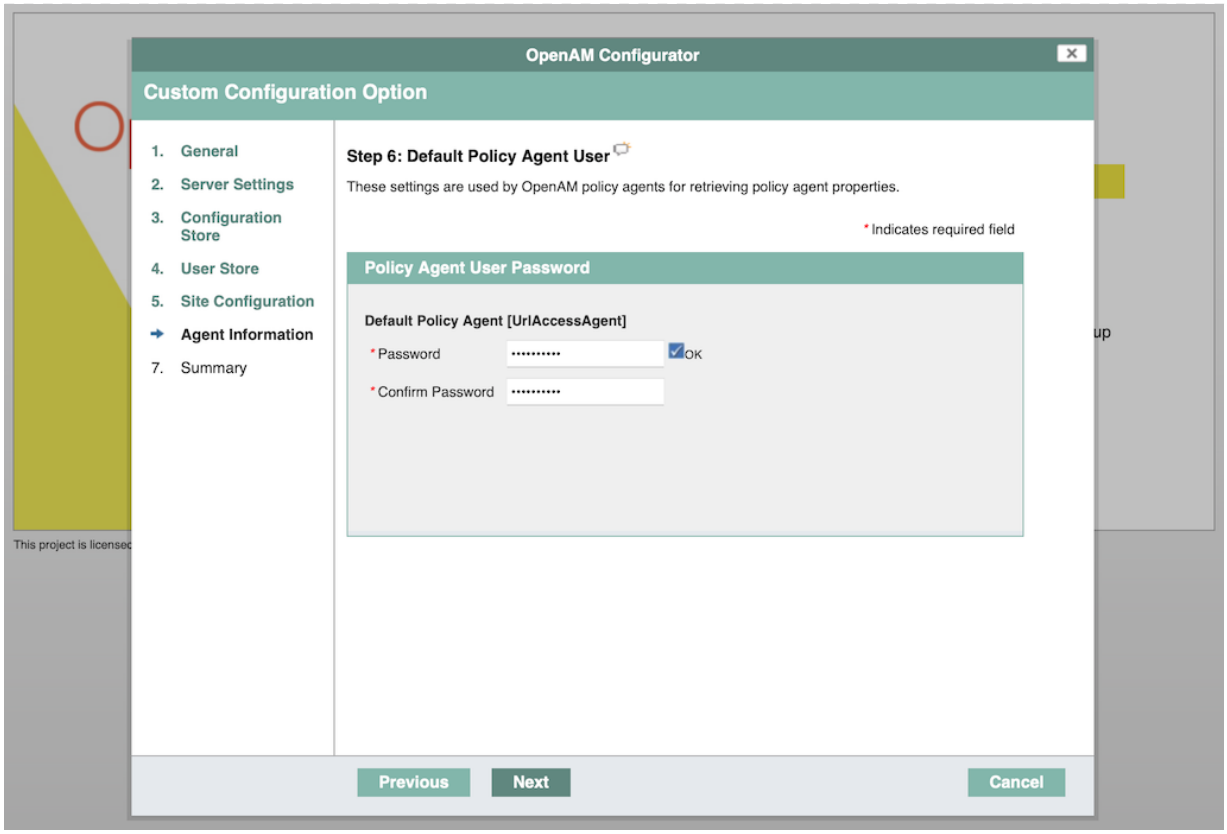
   If you have a site configuration with a load balancer, you can enable session high availability persistence and failover.[1] OpenAM then stores sessions across server restarts, so that users do not have to login again.

   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".

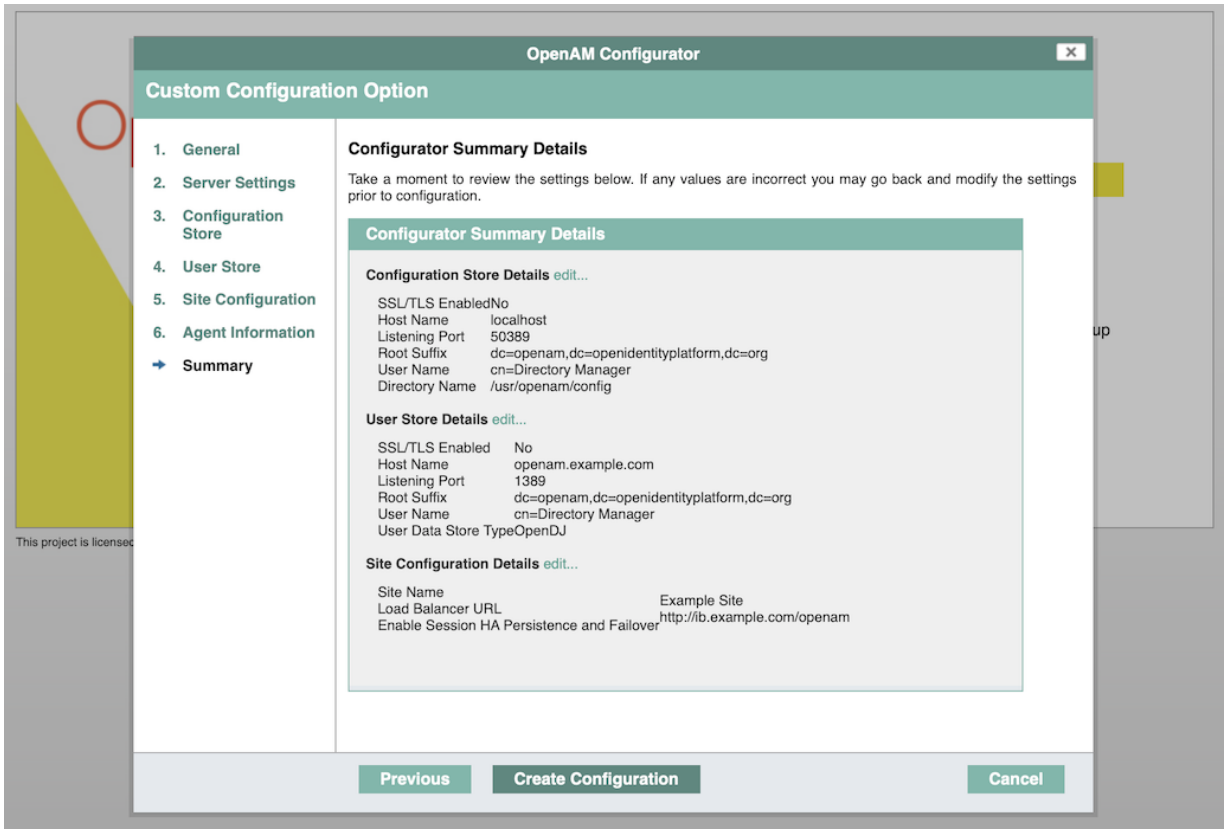


   It is possible to set up a site after initial installation and configuration, as is described in "Setting Up OpenAM Session Failover".

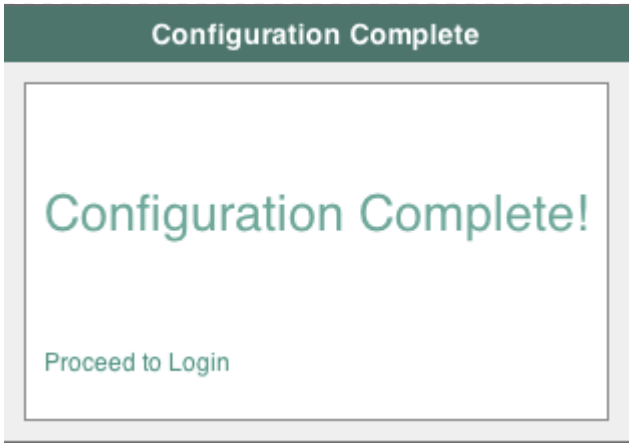
8. In the Agent Information screen, provide a password with at least eight characters to be used by policy agents to connect to OpenAM.



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

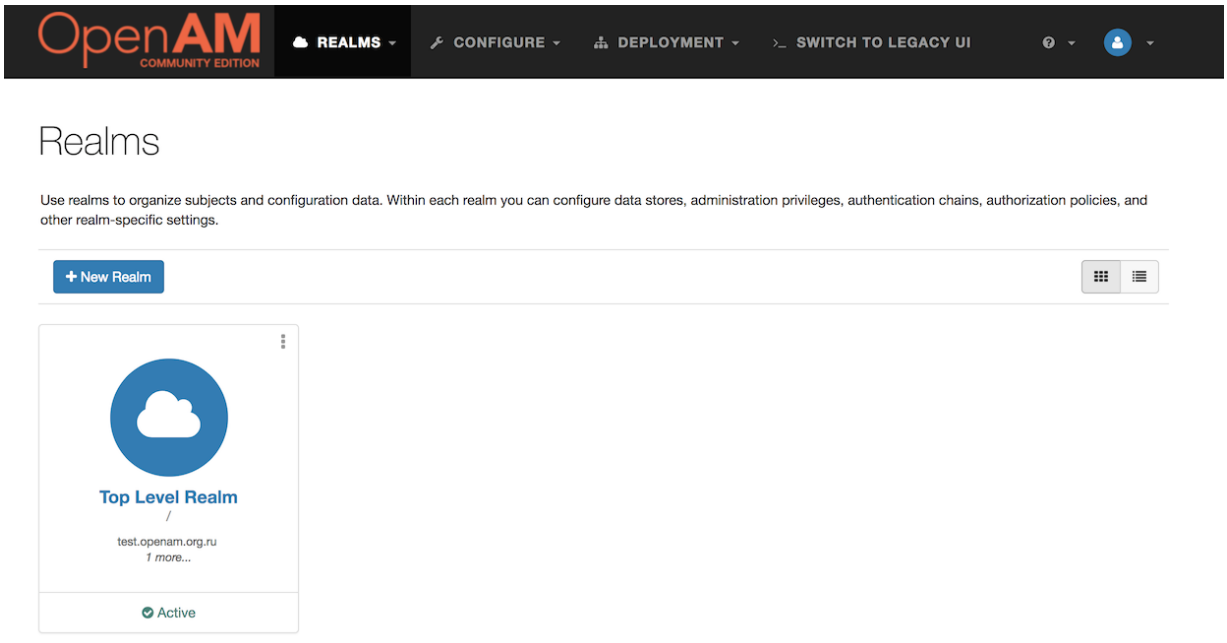


After you click Create Configuration in the summary screen, configuration proceeds, logging progress that you can read in your browser and later, in the installation log. The process ends, and OpenAM shows the Proceed to Login prompt.



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

To Add a Server to a Site

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

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

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 the URL of the first OpenAM server in the site.

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

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

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

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

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.

To Troubleshoot an OpenAM Installation

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

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

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

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

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

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

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

[1] You can configure OpenAM to store sessions *statefully* or *statelessly*. Stateful sessions are stored in memory on the OpenAM server. They are also written to disk by the"Configuring the Core Token Service"if you select the Enable Session HA and Persistence and Failover option in the Site Configuration screen. Stateless sessions are stored in HTTP cookies. The Enable Session HA and Persistence and Failover setting does not apply to stateless sessions. For more information about stateful and stateless sessions, see"Configuring Session State"in the*Administration Guide*.

Installing OpenAM Tools

OpenAM tools are found in `.zip` files where you unpacked the archive of the entire package, such as `~/Downloads/openam`. A list and description of these files follows.

`SSOAdminTools-OpenAM 15.1.7-SNAPSHOT.zip`

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

See "To Set Up Administration Tools".

`SSOConfiguratorTools-OpenAM 15.1.7-SNAPSHOT.zip`

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

See "To Set Up Configuration Tools".

To Set Up Administration Tools

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

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

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

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

4. Unpack the tools:

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

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

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

6. (Optional) If you use IBM Java, add `-D"amCryptoDescriptor.provider=IBMJCE"` and

`-D"amKeyGenDescriptor.provider=IBMJCE"` options to the `setup` or `setup.bat` script before you install the tools.

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

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

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

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

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

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

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

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

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

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

`ssoadm.bat` script before using the script.

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

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

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

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

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

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

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

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

    b. Make the text file read-only:

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

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

    ```
    $ cd /path/to/openam-tools/admin/openam/bin/
    $ ./ssoadm list-servers -u amadmin -f $HOME/.pwd.txt
    http://openam.example.com:8080/openam
    ```

11. If desired, enable the `ssoadm.jsp` page as described in "OpenAM ssoadm.jsp" in the *Administration Guide*.

    You can run most (but not all) of the `ssoadm` subcommands from the `ssoadm.jsp` page in OpenAM after the page has been enabled.

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

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

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

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

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

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

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

To Set Up Configuration Tools

1. Verify 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-OpenAM 15.1.7-SNAPSHOT.zip
   Archive:  ~/Downloads/openam/SSOConfiguratorTools-OpenAM 15.1.7-SNAPSHOT.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-OpenAM 15.1.7-SNAPSHOT.jar
 extracting: lib/servlet-api-2.5.jar
```

4. Configure OpenAM server in a silent mode by using the openam-configurator-tool-OpenAM 15.1.7-SNAPSHOT.jar tool after you deploy the `.war` file.

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

   The openam-configurator-tool-OpenAM 15.1.7-SNAPSHOT.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=openam.example.com
ACCEPT_LICENSES=true
DATA_STORE=embedded
DIRECTORY_SSL=SIMPLE
DIRECTORY_SERVER=openam.example.com
DIRECTORY_PORT=50389
DIRECTORY_ADMIN_PORT=4444
DIRECTORY_JMX_PORT=1689
ROOT_SUFFIX=dc=openam,dc=forgerock,dc=org
DS_DIRMGRDN=cn=Directory Manager
DS_DIRMGRPASSWD=password
```

   When 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-{openam-version}.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 `SIMPLE` as follows:

```
$ 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=openam.example.com
ACCEPT_LICENSES=true
DATA_STORE=embedded
DIRECTORY_SSL=SIMPLE
DIRECTORY_SERVER=openam.example.com
DIRECTORY_PORT=50389
DIRECTORY_ADMIN_PORT=4444
```

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

6. Then, when the OpenAM `.war` file is deployed and the server is running, configure the server to use HTTPS using the openam-configurator-tool-OpenAM 15.1.7-SNAPSHOT.jar tool with the properties file as follows.

```
java '-Djavax.net.ssl.trustStore=PATH_TO_JKS_TRUSTSTORE' \
      -jar openam-configurator-tool-OpenAM 15.1.7-SNAPSHOT.jar \
      --file config.properties
```

For additional information about the command-line tool, see the reference documentation for configurator.jar(1) in the *Reference*.

Installation Considerations for Multiple Servers

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

Things to Consider When Installing Multiple Servers

When installing multiple servers, consider the following points:

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

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

- The cookie domain is set to the server's full URL that was used to access the configurator, such as `example.net`, `server.west.example.com`, or `example.local`.
- You can use a load balancer layer to protect OpenAM services as well. The load balancer can restrict access to OpenAM services, throttle traffic, offload HTTPS encryption, and so forth.

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

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

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

To Configure a Site With a First OpenAM Server

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 navigate to Deployment > Sites.
2. Click New to start configuring the new site.
3. On the New Site page enter the site name, and set the Primary URL to the load balancer URL that is the entry point for the site, such as `https://lb.example.com/openam`.

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

   Client applications and policy agents access the servers in the site through the site URL.
4. Click Save to keep the site configuration.
5. Navigate to Deployment > Servers > *Server Name* > General.
6. Set the Parent Site to the name of the site you just created, and then select Save Changes.

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

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

Configuring Load Balancing for a Site

Load balancer configuration requirements differ for OpenAM sites configured to use stateful and stateless sessions.[2] For more information about OpenAM session types, see "Configuring Session State" in the *Administration Guide*.

Load Balancer Configuration for Stateful Sessions

An OpenAM site configured to use stateful sessions achieves the best performance when the server that originally authenticated a user continually manages that user's session, unless that server is no longer available.

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

To Configure Site Load Balancing for Deployments With Stateful Sessions

1. For each OpenAM server in the site, navigate to Deployment > Servers > *Server Name* > General and set Parent Site to the site you created. Then, save your work.
2. Ensure that the `amlbcookie` cookie has a unique for each OpenAM server.
   a. For each OpenAM server in the site, navigate to Deployment > Servers > *Server Name* > Advanced and review the value of the `com.iplanet.am.lbcookie.value` property. By

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

Keep the value of the `amlbcookie` cookie set to the OpenAM server ID to reduce crosstalk among the OpenAM servers when using Web Policy Agent 4.1.x with CDSSO mode enabled.

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

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

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

Changes take effect only after you restart the 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.

Load Balancer Termination

When traffic to and from the load balancer is protected with HTTPS, the approach described in "To Configure Site Load Balancing for Deployments With Stateful Sessions" requires that you terminate

the connection on the load balancer. You then either re-encrypt the traffic from the load balancer to OpenAM, or make connections from the load balancer to OpenAM over HTTP.

Request Forwarding Caveats

Sticky load balancing based on the value of the `amlbcookie` cookie does not guarantee request forwarding to the corresponding OpenAM server in all cases. For example, 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.

The OpenAM server that does not hold the user's session can attempt to locate the user's session by retrieving it from the Core Token Service's token store, or by communicating with other OpenAM servers in an OpenAM site using back-channel communication over the network. This back-channel communication is called *crosstalk*.

By default, OpenAM sites are configured with the Reduce Crosstalk option enabled. With this option enabled, the OpenAM server that does not hold the user's session attempts to retrieve it from the Core Token Service's token store if session failover is enabled.

For example, suppose you deploy several OpenAM servers in a site configured for session failover. If the site's load balancer directs a user's request to a server other than the OpenAM server that held the user's session, then the server will attempt to retrieve the session from the Core Token Service, provided you have not modified the default OpenAM configuration.

If you disable the Reduce Crosstalk option, the OpenAM server that does not hold the user's session attempts to retrieve it by using crosstalk. Because crosstalk generates network traffic, locating sessions from the Core Token Service's token store is preferred for performance reasons.

Requests to update sessions, such as requests to log out, reset the session idle time, 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.

Load Balancer Configuration for Stateless Sessions

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

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

Handling HTTP Request Headers

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

Configure your load balancer or proxy to pass the information to OpenAM by using request

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

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

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

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

Handling Multiple Cookie Domains When Using Wildfly

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

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

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

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

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 Open Identity Platform 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 may want to change the default styling and branding as well as customize different realms.

- By default, end users see the XUI pages.

  See "Customizing the End User Interface" 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.

  See "Customizing the Classic User Interface (Legacy)" for details.

  To enable the classic UI, disable the XUI.

  You can disable XUI globally for an OpenAM server in OpenAM console under Configure > Authentication > Core Attributes > 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 the XML callback files for authentication modules used 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 the OpenAM console under Deployment > Servers > *Server Name* > Advanced. Before using OpenAM in production, set `org.forgerock.openam.core.resource.lookup.cache.enabled` back to the default setting, `true`.

Customizing the End User Interface

This section covers customizing the default user interface, known as the XUI.

Theming the XUI

This section explains how to use themes to alter the appearance of user-facing XUI pages.

The XUI is built with the Bootstrap framework, and supports Bootstrap themes to customize the look and feel of the user interface.

Only user-facing XUI pages support themes. The OpenAM administration console cannot be themed.

You can apply themes to specific realms, and also to specific authentication chains within those realms. OpenAM includes a *default* theme, and an inverted *dark* theme.

To Apply a Theme to the XUI

This procedure demonstrates adding a custom Bootstrap theme to the XUI.

1. Copy your custom Bootstrap theme to a directory in `/path/to/tomcat/webapps/openam/XUI/themes/`. A custom Bootstrap theme should consist of one or more CSS files, and optionally media and font files.

   As an example, the *dark* theme is available in: `/path/to/tomcat/webapps/openam/XUI/themes/dark/`.

2. Edit the `/XUI/config/ThemeConfiguration.js` file, to reference the CSS files in the theme, and to map the theme to realms and authentication chains:

   a. Locate the `themes` element, and under it create a new element with the name of your theme. The following example adds a theme called `myTheme`:

   ```
   define("config/ThemeConfiguration", {
       themes: {
           // There must be a theme named "default".
           "default": { ... },
           "fr-dark-theme": { ... },
           "myTheme": {}
       },
       mappings: [ ... ]
   });
   ```

   b. In the new theme element, create a `stylesheets` array containing the theme's two CSS files, followed by the required `css/structure.css` file.

   ```
   define("config/ThemeConfiguration", {
       themes: {
           // There must be a theme named "default".
           "default": { ... },
           "fr-dark-theme": { ... },
           "myTheme": {
               stylesheets: [
                   "themes/dark/css/bootstrap.min.css",
                   "themes/dark/css/theme-dark.css",
                   "css/structure.css"
               ]
           }
       },
       mappings: [ ... ]
   });
   ```

Note that you must specify paths relative to the `XUI` directory.

If required, specify additional settings specific to the new theme, such as the logos to use or the footer information. For information on the available settings, see "XUI Configuration Parameters" in the *Reference*.

c. Locate the `mappings` array, and create a new element under it to map your new theme to realms and authentication chains.

   Elements in the `mappings` array are evaluated in order from top to bottom. The first theme that matches the current realm and/or authentication chain is applied. Any subsequent mappings, even if true, are ignored once a match is found.

   If no match is found, the `default` theme is applied.

   i. Create a `theme` element, and set the value to the name of your new theme:

   ```
   define("config/ThemeConfiguration", {
       themes: { ... },
       mappings: [
           {
               theme: "myTheme"
           }
       ]
   });
   ```

   ii. (Optional) Optionally, create a `realms` array, and include the realms the theme will apply to:

   ```
   define("config/ThemeConfiguration", {
       themes: { ... },
       mappings: [
           {
               theme: "myTheme",
               realms: ["/", "/test-realm", /^\/a/]
           }
       ]
   });
   ```

   You can use a regular expression to specify the realms the theme should apply to. For example `/^\/a/` will apply the theme to all realms that start with `/a`, including `/ab` and `/a/c`.

   If you do not include a realms array, the theme is applied to all realms.

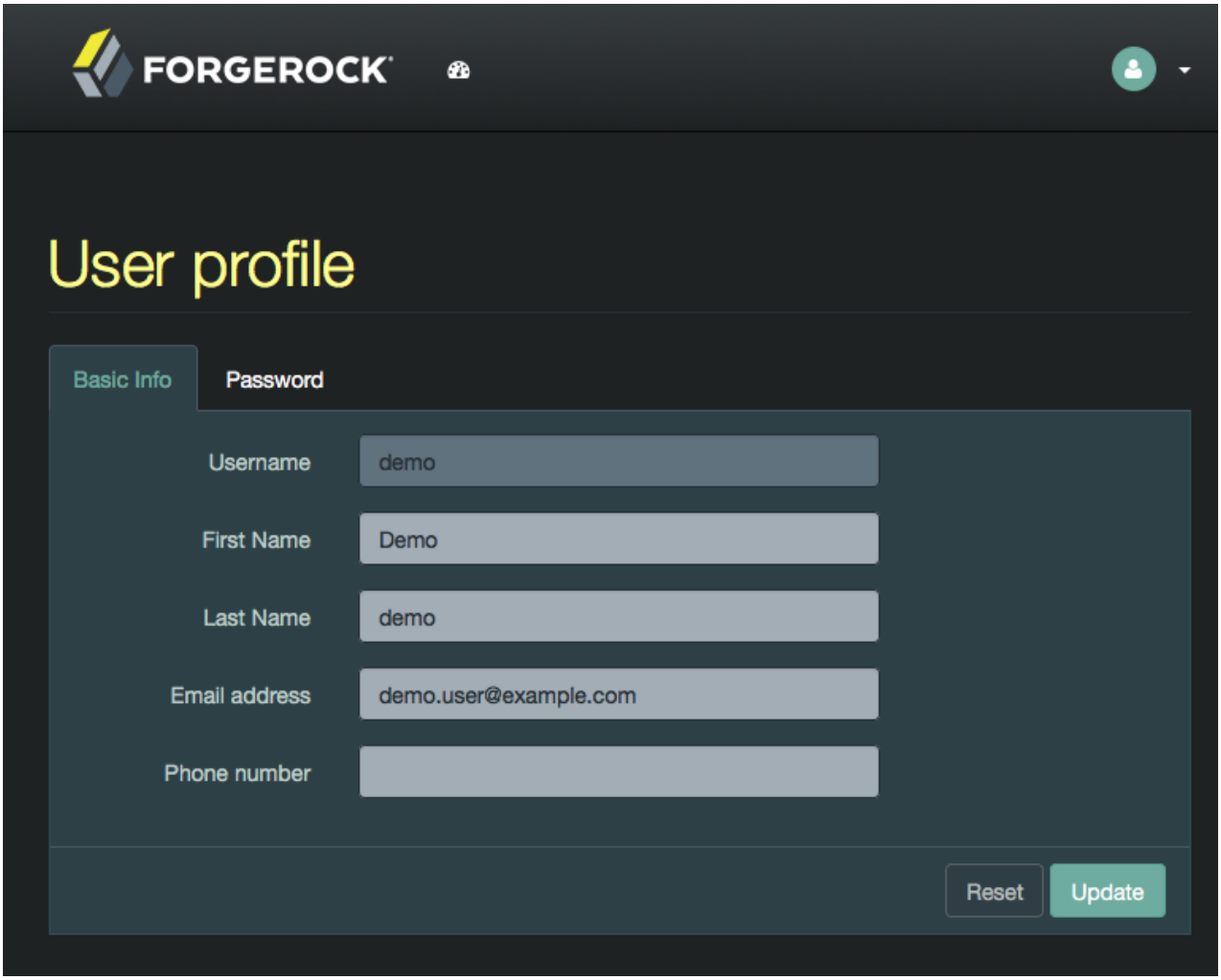
   iii. (Optional) Optionally, create an `authenticationChains` array, and include the authentication chains the theme will apply to when used:

```
define("config/ThemeConfiguration", {
    themes: { ... },
    mappings: [
        {
            theme: "myTheme",
            realms: ["/", "/test-realm", /^\/a/],
            authenticationChains: ["auth-chain-one"]
        }
    ]
});
```

If you specify both realms and authentication chains, the theme is only applied when both criteria are true.

3. Save your work.

   The next time a user logs in to the XUI they will see the new theme applied:



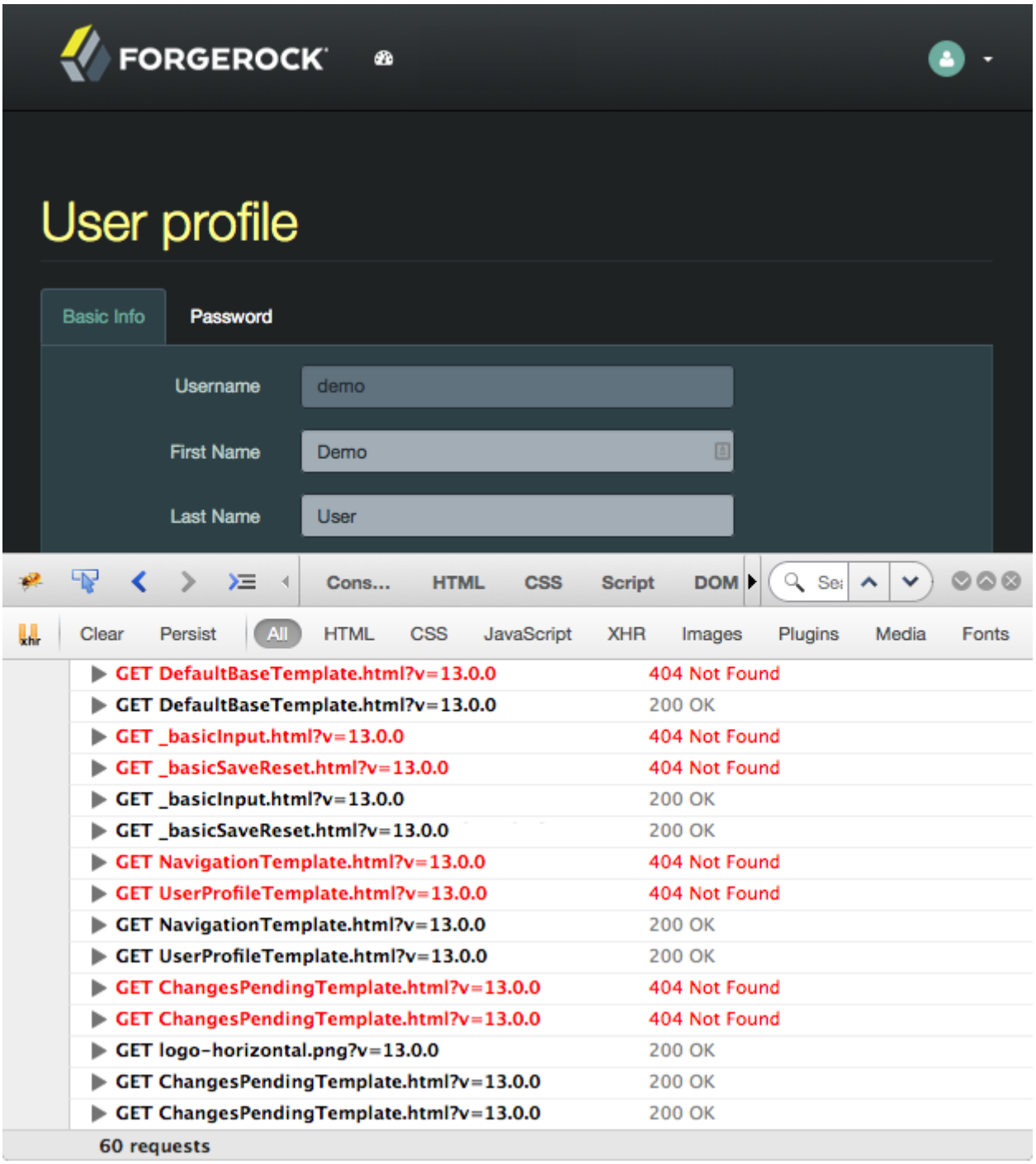
Customizing XUI Layout

This section explains how to alter the layout of end user-facing XUI pages.

XUI pages are built with HTML templates, which in turn may contain reusable snippets of HTML stored in files referred to as *partials*.

The XUI stores the default templates in `/path/to/tomcat/webapps/openam/XUI/templates` and the default partials in `/path/to/tomcat/webapps/openam/XUI/partials`. You can override some, or all of these files by making duplicates containing edits and instructing the XUI to use the duplicates in place of the defaults.

If you provide a subset of the templates and partials provided with OpenAM, the XUI will fall back to the default set if a customized version is not provided. Note however that this will result in HTTP 404 Not Found errors in the background, which are visible in browser developer tools, but not visible to the end user:



To avoid HTTP 404 Not Found errors when customizing XUI layouts, duplicate the entire

`/XUI/templates` and `/XUI/partials` directories into your custom theme directory, rather than only copying files that will be edited.

To Customize XUI Layout

This procedure demonstrates customizing the default XUI layout by overriding a partial file.

Follow these steps on the server where OpenAM is deployed:

1. Copy the directories containing the templates and partials you want to customize to a directory in `/path/to/tomcat/webapps/openam/XUI/themes/`, ensuring that you maintain the same directory structure.

   The following example copies the directory containing the default partials used for login pages into the `dark` theme directory, maintaining the `/partials/login/` directory structure:

   ```
   $ cd /path/to/tomcat/webapps/openam/XUI
   $ mkdir -p themes/dark/partials
   $ cp -r partials/login/ themes/dark/partials/
   ```

2. Edit the copied template or partial files with the changes you require.

   For example, to include an HTML `<hr/>` tag to create a horizontal line that renders above password fields on login pages, edit the following file: `/path/to/tomcat/webapps/openam/XUI/themes/dark/partials/login/_Password.html`

   ```
   <hr />
   <label for="{{id}}" class="aria-label sr-only">{{prompt}}</label>
   <input type="password"
       id="{{id}}"
       name="callback_{{index}}"
       class="form-control input-lg"
       placeholder="{{prompt}}"
       value="{{value}}"
       data-validator="required"
       required
       data-validator-event="keyup"
       {{#equals index 0}}autofocus{{/equals}}>
   ```

3. Edit the `/path/to/tomcat/webapps/openam/XUI/config/ThemeConfiguration.js` file, and add a `path` element that points to the newly edited templates or partials within the theme they will apply to.

   The following example alters the `fr-dark-theme` to use the custom login partials:

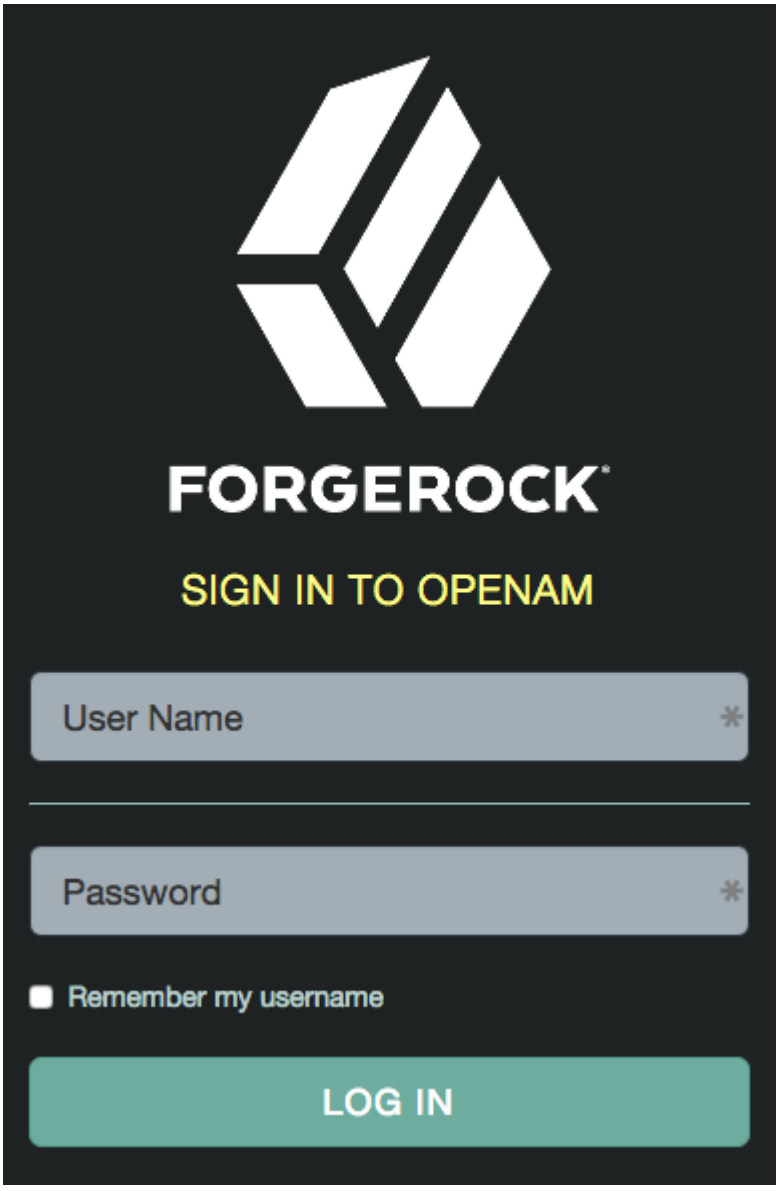
   ```
   "fr-dark-theme": {
        path: "themes/dark/",
        stylesheets: [ ... ],
   ```

```
    settings: { ... }
}
```

Note that the trailing slash in the `path` value is required.

4. Save your work.

   The next time a user visits the login page in the XUI they will see the new partial applied, with the horizontal line above the password field:



Localizing the XUI

This section explains how to localize the text that is generated for the user-facing XUI pages.

The text the XUI displays comes from from `translation.json` files located in locale-specific directories.

To customize the English text, edit `/path/to/tomcat/webapps/openam/XUI/locales/en/translation.json` under the directory where OpenAM is deployed.

To prepare a translation for a new locale, copy the provided `/path/to/tomcat/webapps/openam/XUI/locales/en` directory to `/path/to/tomcat/webapps/openam/XUI/locales/locale`, and edit the duplicate by changing the values, and taking care not to change the JSON structure or to render it invalid.

The *locale* should be specified as per rfc5646 - Tags for Identifying Languages. For example, `en-GB`.

Customizing the Classic User Interface (Legacy)

To customize the classic UI, first copy the pages to customize to the proper location, and then 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 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 deployments, 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.

- "To Copy the Pages to Customize For the Top-Level Realm"
- "To Copy the Pages to Customize For Another Realm"
- "To Customize Files You Copied"

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, `ja` for Japanese, or `zh_CN` for Simplified Chinese.

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

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

   See "How OpenAM Looks Up UI Files" for details.

3. You can now either follow the steps in "To Copy the Pages to Customize For Another Realm", or in "To Customize Files You Copied".

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. Copy the default files and, optionally, the localized files to suffix `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 "To Customize Files You Copied".

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

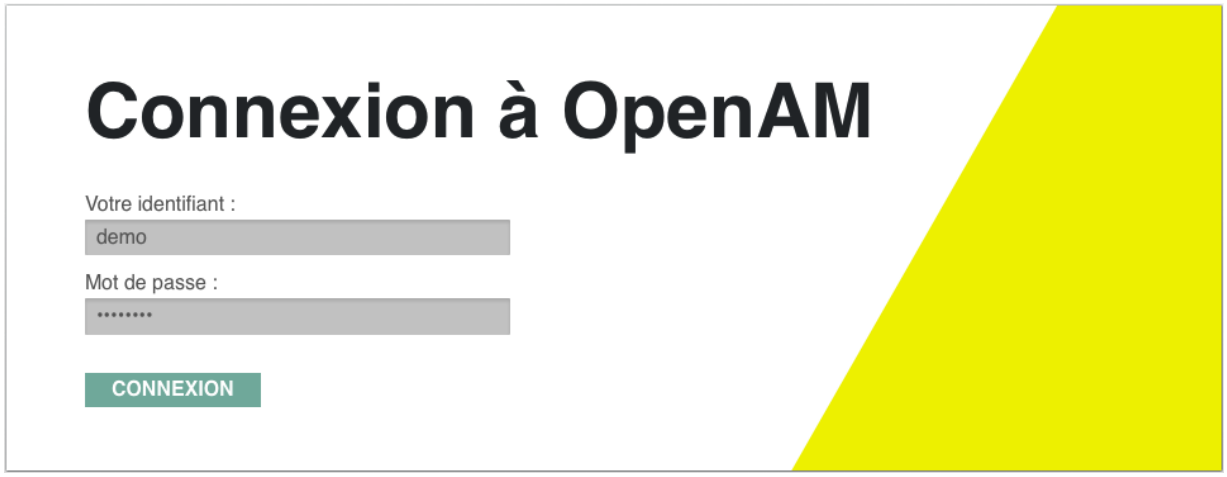
through the `.xml` files.

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` in `openam_fr/html/DataStore.xml`.

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

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-OpenAM 15.1.7-SNAPSHOT.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-OpenAM 15.1.7-
   SNAPSHOT.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-OpenAM 15.1.7-SNAPSHOT.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-OpenAM 15.1.7-SNAPSHOT.jar /path/to/tomcat/webapps/openam/WEB-INF/lib/
```

   When preparing for production deployment make the modification in the OpenAM war file, `OpenAM-OpenAM 15.1.7-SNAPSHOT.war`, instead.

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

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 configure locales as follows. 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 setting value under Deployment > Servers > *Server Name* > General > System or you can set 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 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 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
```

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
```

Configuring the Core Token Service

The Core Token Service (CTS) provides a persistent and highly available token storage for OpenAM session, OAuth 2.0, SAML v2.0, and UMA tokens. CTS is set up in a generalized token storage format, which by default is always used for OAuth 2.0 and UMA tokens. If configured, it can also be used to persist session, session blacklist, and SAML v2.0 tokens.

OpenAM's Session Failover (SFO) mechanism uses the Core Token Service (CTS) to store its *stateful* session data[1]. During SFO, OpenAM sends an SSO token to its clients, either as a cookie in a browser or in a JSON response to the `authentication` endpoint. This allows OpenAM to retrieve the session object from memory to resume the session.

General Recommendations for CTS Configuration

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

- **Only Use the Embedded Configuration Store for Limited, Single-Server Test Cases**. By default, OpenAM writes CTS entries in the OpenAM configuration store: either an embedded or external configuration store. If you configured OpenAM to use an embedded configuration store, limit your use of this default deployment to very small-scale, single-server test deployments—in multi-server deployments with load balancing, the active/active topology used by multiple embedded configuration stores can lead to write collisions.
- **Isolate the Different Stores**. CTS entries are large, around 5KB, but are short-lived, whereas configuration data is static and long-lived. User entries are more dynamic than configuration data but much less volatile than CTS data. Therefore, isolating the user, configuration, and CTS data from OpenAM into separate stores allow for different tuning and storage settings per token store type.
- **Configure External CTS Stores for High Volumes**. If you require a higher-level performance threshold, you may want to move the CTS token storage to one or more dedicated systems, as CTS generally causes much more replication traffic than less volatile configuration data. Note that CTS data are highly volatile with high writes (about 90%) and low reads (about 10%).

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

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

  - Active/passive deployments, in which OpenAM's connection to the CTS token store is limited to a single master instance with failover instances.

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

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

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

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

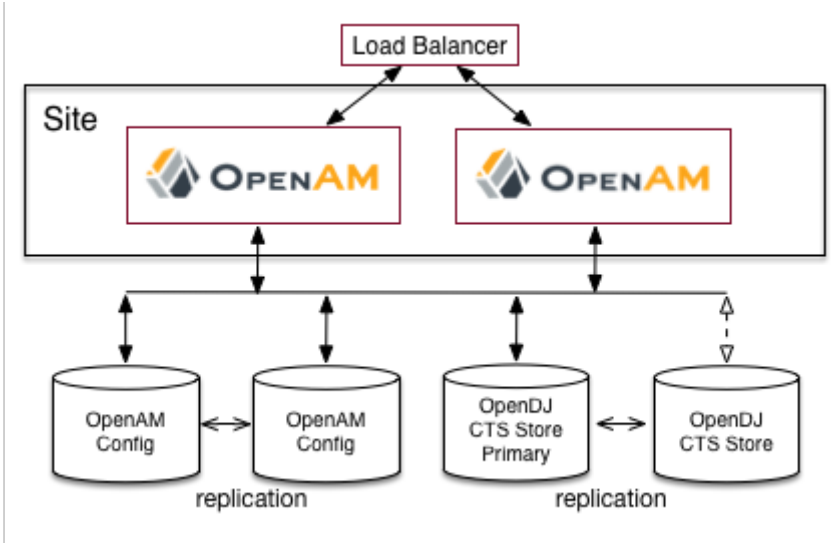
  Do not deploy CTS token stores behind a load balancer. Instead, specify connections to the directory server instances that comprise the CTS token store by using the Connection String(s) property in the CTS configuration.

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

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

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

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



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

- **Consider Dedicated Replication Servers**. Once configured, the OpenDJ server replicates CTS data transmitted from OpenAM servers to connected OpenDJ servers. The amount of replication traffic can be significant, especially if replication proceeds over a WAN. You can limit this replication traffic by separating OpenDJ instances into directory and replication servers. For more information on how this is done with OpenDJ, see the OpenDJ documentation on Standalone Replication Servers.
- **Replicate CTS Across Sites to Support Global Session Availability**. CTS supports uninterrupted session availability in deployments with multiple sites if all sites use the same global underlying CTS store replicated across all sites. If an entire site fails or becomes unavailable, OpenAM servers in another site can detect the failure of the site's load balancer and attempt to use sessions from the global Core Token Service.

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

  For more information on CTS for global session high availability with OpenDJ server, see "Setting Up OpenAM Session Failover" and the OpenDJ documentation on Managing Data Replication.

CTS Deployment Steps

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

In general, CTS causes more volatile replication traffic due to the nature of its short-lived tokens

compared to regular configuration data. To handle the data volatility, you can configure OpenAM to use the embedded directory server as a dedicated configuration 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 a WAN.

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

"Prepare the OpenDJ Directory Service for CTS"

"Import CTS Files"

"Non-Admin User Creation and ACI Import"

"CTS Index Import and Build"

"OpenAM CTS Configuration"

"Testing Failover"

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

Prepare the OpenDJ Directory Service for CTS

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

To Download and Install OpenDJ

1. Go to the GitHub Releases page, click and then download the recent version of OpenDJ directory server.
2. Unzip the OpenDJ distribution and run `setup`, which launches a GUI application called the QuickSetup Wizard. If you want to run `setup` interactively from the command line, use `setup --cli`.
3. Install OpenDJ with the installation parameters necessary for your deployment. Note, however, that SSL may be required in production deployments. This example uses the following parameters:

```
Accept license?: yes
Initial Root User DN for the Directory Server: cn=Directory Manager
Password for the Initial Root User: <password value>
Fully Qualified Hostname: cts.example.com
LDAP Listening Port: 1389
Administration Connector Port: 4444
Create Base DNs: yes
Backend Type*: JE Backend ([1])
```

```
Base DN for Directory Data: dc=cts,dc=example,dc=com
Option for Populating Database: Option 2 - Only create base entry
Do You Want to Enable SSL: no (may be required for your deployment)
Do You Want to Enable StartTLS: no (may be required for your deployment)
Do You Want To Start The Server: yes
What Would You Like To Do: 1 - Set up server with parameters above
```

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

Import CTS Files

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

To Import the CTS Configuration

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

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

If you are using OpenDJ 4.0 or later:

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

If you are using OpenDJ 3.5 or earlier:

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

The output should be:

```
Processing MODIFY request for cn=schema
```

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

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

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

If you are using OpenDJ 4.0 or later:

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

If you are using OpenDJ 3.5 or earlier:

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

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

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

If you are using OpenDJ 4.0 or later:

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

If you are using OpenDJ 3.5 or earlier:

```
$ ./ldapmodify \
--port 1389 \
```

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

The output should be:

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

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

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

Non-Admin User Creation and ACI Import

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

To Create a Non-Admin User

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

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

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

   Finally, make sure that you replace the *password* tag with your actual password:

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

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

2. Add the new user to the CTS repository:

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

The output should be:

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

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

```
./dsconfig \
  set-access-control-handler-prop \
  --no-prompt \
  --hostname cts.example.com \
  --port 4444 \
  --bindDN "cn=Directory Manager" \
  --bindPassword password \
  --add 'global-aci:(target = "ldap:///cn=schema")(targetattr = "attributeTypes
```

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

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

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

   Verify that the following entry is present:

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

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

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

6. Import the ACIs into the CTS repository:

```
./ldapmodify \
  --defaultAdd \
```

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

The output should be:

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

CTS Index Import and Build

To Import and Rebuild the CTS Indexes

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

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

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

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

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

2. Rebuild all indexes and then verify them:

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

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

3. Restart the OpenDJ instance.

OpenAM CTS Configuration

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

To Configure CTS in OpenAM

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

   CTS Token Store Parameters

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

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

   External Store Configuration Parameters

Parameter	Value	Notes
SSL/TLS Enabled	False	
Connection String(s)	`cts.example.com:1389`	
Login ID	`uid=openam_cts,ou=admins,dc=cts,dc=example,dc=com`	
Password	*password*	

Parameter	Value	Notes
Heartbeat	`10`	For production, this value needs to be tuned.

4. Click Save Changes
5. On the OpenAM console, navigate to Configure > Global Services, and then click Session.
6. In Secondary Configuration Instance, click New, select the site from the drop-down list, and then click Add.
7. In the Global Attributes section, configure the parameters as follows:

   Global Attributes Parameters

Parameter	Value
Session persistence and High Availability Failover Enabled	True
Reduce Crosstalk Enabled	True
Session Logout/Destroy Broadcast	`Disabled`
Reduced Crosstalk Purge Delay	`1`

   NOTE When using the Reduce Crosstalk feature, OpenAM goes to the CTS data store to retrieve session information, rather than poll the other OpenAM servers in the pool, which may hold the sessions in memory. Therefore, you must consider the load, latency, and characteristics of the target environment to decide if the Reduce Crosstalk option should be enabled.

8. Click Save.
9. Restart all OpenAM servers in the site and test the configuration.

Testing Failover

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

To Test Failover

1. In Chrome, log in to the second OpenAM instance with the `amadmin` user, and click on `sessions`.
2. In Firefox, log in to the first OpenAM instance with a test user.
3. In Chrome, verify that the test user exists in the first OpenAM instance's session list and not in the second instance.
4. Shut down the first OpenAM instance.
5. In Firefox, rewrite the URL to point to the second OpenAM instance. If successful, the browser should not prompt for login.

6. Confirm the session has failed over. In Chrome, list the sessions on the second instance, the test user's session should be present.
7. Restart the first OpenAM instance to complete the testing.

CTS Backups and OpenDJ Replication Purge Delay

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

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

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

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

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

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

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

```
  --recurringTask "x * * * *" \
  --completionNotify backupadmin@example.com \
  --errorNotify backupadmin@example.com
```

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

Managing CTS Tokens

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

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

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

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

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

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

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

`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: 300000 ms.

`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: 60000 ms.

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

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

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

CTS Tuning Considerations

The following OpenAM components make CTS requests:

- Session service for stateful session failover
- Session service for stateless session blacklisting
- OAuth 2.0 for token persistence
- SAML v2.0 for token persistence
- UMA for token persistence
- REST API for functions like forgotten passwords

All create, update, and delete requests to CTS are placed into an asynchronous buffer before being handled by an asynchronous processor. This ensures that callers performing write operations can continue without waiting for CTS to complete processing.

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

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

`org.forgerock.services.cts.async.queue.size`

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

`org.forgerock.services.cts.async.queue.timeout`

Default timeout is 120 seconds. Determines the length of time a caller will wait when the buffer is full. If the timeout expires, the caller receives an error. The timeout property is used in any system configuration where the LDAP server throughput is considerably slower than the OpenAM server, which can result in blocked requests as the backlog increases.

To set the queue size and timeout properties, in the OpenAM Console, navigate to Configure > Server Defaults > Advanced, enter the key name and value, and then click Add.

For additional information on tuning CTS, see "Tuning LDAP CTS and Configuration Store Settings" in the *Administration Guide* in the OpenAM Administration Guide in the *Administration Guide*.

[1] OpenAM also supports *stateless* sessions, which are not stored in memory but are sent to the client, typically, in a browser-based cookie. For more information, see "Configuring Session State" in the *Administration Guide*.

Setting Up OpenAM Session Failover

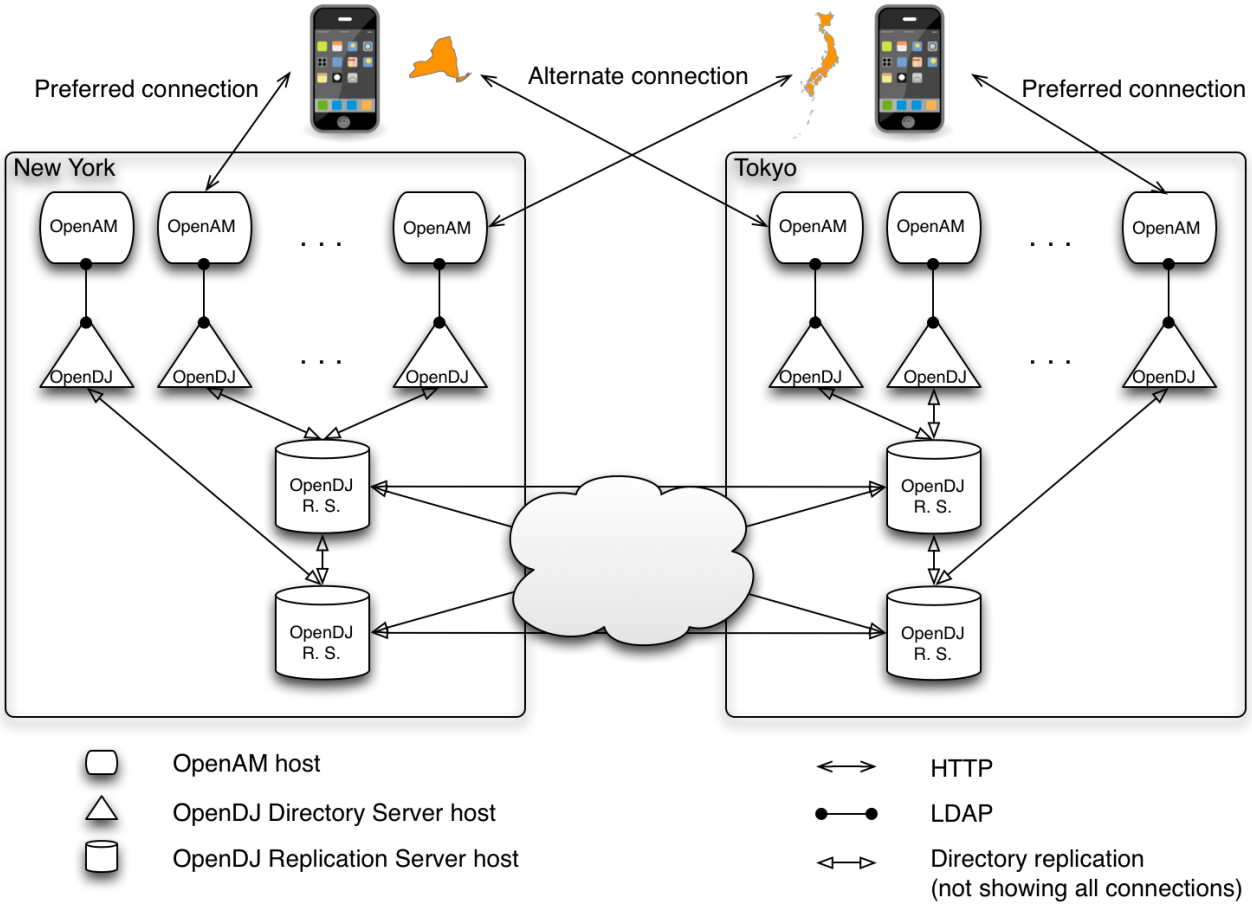
This chapter provides instructions for setting up session failover (SFO). Session failover allows another OpenAM server to manage a session when the server that initially authenticated the user is down. This means the user does not need to log in again, even though the server that authenticated them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

Session failover (high-availability for sessions) builds on OpenAM service availability. Before configuring session failover, you must first make the overall OpenAM service highly available by setting up OpenAM in a site configuration. You can find instructions for setting up a site configuration in "Installation Considerations for 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".

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. 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 "Core Token Service For Global Session Failover".



For more information on CTS for global session failover with OpenDJ directory server, see the OpenDJ documentation on Managing Data Replication.

NOTE

You can configure OpenAM to store sessions *statefully* or *statelessly*. Stateful sessions are stored in memory on the OpenAM server, while stateless sessions are stored in HTTP cookies. An OpenAM deployment configured for session failover stores stateful (but *not* stateless) sessions in the Core Token Service. Therefore, the session failover mechanism described in this section applies to stateful sessions only.

Because stateless sessions reside in HTTP cookies, they do not need to be retrieved from a persistent data store in the event of a server failure—they can be retrieved from the cookies. Therefore, OpenAM does not store stateless sessions in the CTS store.

For more information about stateful and stateless sessions, see "Configuring Session State" in the *Administration Guide*.

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 for Deployments With Stateful Sessions", and then follow these steps.

1. In the OpenAM console for one of the servers in the site, under Configure > Global Services, 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 instead of crosstalk, check the Reduce Crosstalk Enabled box. For more information about crosstalk, see the section, "Installation Considerations for Multiple Servers".

   Do not disable reduced crosstalk unless advised to do so by Open Identity Platform Approved Vendors 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.

   OpenAM enables session failover immediately after you save the configuration changes. It is not necessary to restart the servers in your site.

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.

To Remove OpenAM Core Software

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

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

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

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

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

   A full uninstall of OpenAM core services and configuration files consists of removing the following directories:

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

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

     Or:

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

If you used an external configuration store, you must remove the configuration manually from your external directory server. The default base DN for the OpenAM configuration is `dc=openam,dc=forgerock,dc=org`.

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

3. Undeploy the OpenAM web application.

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

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