Getting Started

/ OpenAM 13

Latest update: 13.0.0

Mark Craig
Gene Hirayama

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

Abstract

Quick introduction to OpenAM for new users and readers evaluating the product. OpenAM provides open source Authentication, Authorization, Entitlement, and Federation software.

Table of Contents

Preface iv
1. Who Should Use this Guide iv
2. Formatting Conventions iv
3. Accessing Documentation Online v
4. Using the ForgeRock.org Site v
5. Getting Support and Contacting ForgeRock v
1. Protecting a Web Site With OpenAM 1
1.1. About OpenAM 1
1.2. Software Requirements To Try Out OpenAM 2
1.3. Setting Up the Software 3
1.4. Trying It Out 20
1.5. Trying Out Stateless Sessions 22
2. Where To Go From Here 23
2.1. User Self-Service Features 23
2.2. Single Sign-On 24
2.3. Standards-Based Federation 24
2.4. Access Policies 25
2.5. Protect Any Web Application 25
2.6. Modern APIs For Developers 26
2.7. Getting Help With Your Project 26
Index 27

Preface

This guide shows you how to install and get started with OpenAM.

1. Who Should Use this Guide

This guide is written for access management designers and administrators who build, deploy, and maintain OpenAM services for their organizations. This guide covers the tasks you need to quickly get OpenAM running on your system.

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

2. Formatting Conventions

Most examples in the documentation are created in GNU/Linux or Mac OS X operating environments. If distinctions are necessary between operating environments, examples are labeled with the operating environment name in parentheses. To avoid repetition file system directory names are often given only in UNIX format as in `/path/to/server`, even if the text applies to `C:\path\to\server` as well.

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

Command-line, terminal sessions are formatted as follows:

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

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

Program listings are formatted as follows:

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

3. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

- The ForgeRock Knowledge Base offers a large and increasing number of up-to-date, practical articles that help you deploy and manage ForgeRock software.

  While many articles are visible to community members, ForgeRock customers have access to much more, including advanced information for customers using ForgeRock software in a mission-critical capacity.

- ForgeRock 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.

4. Using the ForgeRock.org Site

The ForgeRock.org site has links to source code for ForgeRock open source software, as well as links to the ForgeRock forums and technical blogs.

If you are a *ForgeRock customer*, raise a support ticket instead of using the forums. ForgeRock support professionals will get in touch to help you.

5. Getting Support and Contacting ForgeRock

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

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

Chapter 1

Protecting a Web Site With OpenAM

This guide shows you how to quickly set up OpenAM and get started with access management. In reading and following the instructions in this guide, you will learn how to protect a Web page using OpenAM and a Web policy agent.

Important

You need a Linux, Solaris, or Windows system that can run the OpenAM Web policy agent (see the *Web Policy Agent Release Notes* section, *Web Policy Agents Platform Requirements*) with a minimum of 1 GB of available RAM memory, a few hundred MB of free disk space, a web browser, and an Internet connection to download software.

If you are using Mac OS X, set up a virtual machine running Linux to try these procedures because the web policy agent is not built for Apache HTTP Server on Mac OS X.

1.1. About OpenAM

OpenAM provides a service called *access management*, which manages access to resources, such as a web page, an application, or web service, available over the network. Once it is set up, OpenAM provides an infrastructure for managing users, roles, and access to resources. In this chapter, you manage access to a single web page.

OpenAM centralizes access control by handling both *authentication* and *authorization*. Authentication is the process of identifying an individual, for example, by confirming a successful login. Authorization is the process of granting access to resources to authenticated individuals.

OpenAM centralizes authentication by using a variety of authentication modules that connect to identity repositories that store identities and provide authentication services. The identity repositories can be implemented as LDAP directories, relational databases, RADIUS, Windows authentication, one-time password services, and other standards-based access management systems.

OpenAM lets you chain together the authentication services used. Authentication chains let you configure stronger authentication for more sensitive resources for example. They also let you set up modules that remember a device when the user logs in successfully. Or that evaluate the risk given the login circumstances and therefore can require more credentials when a user is logging in from an unusual location. This chapter uses OpenAM's built-in identity repository and authentication modules to make it easier to get started.

OpenAM centralizes authorization by letting you use OpenAM to manage access policies separate from applications and resources. Instead of building access policy into a web application, you install

a policy agent with the web application to request policy decisions from OpenAM. This way you can avoid issues that could arise when developers must embed policy decisions into their applications. With OpenAM, if policy changes or an issue is found after the application is deployed, you have only to change the policy definition in OpenAM, not deploy a new version of the application. OpenAM makes the authorization decisions, and policy agents enforce the decisions on OpenAM's behalf.

The rest of this chapter has you demonstrate OpenAM access management by installing OpenAM, creating a policy, and installing a policy agent on a web server to enforce the policy for a web page.

1.2. Software Requirements To Try Out OpenAM

This chapter shows you how to install the software OpenAM needs to protect a web page. You will learn how to install Apache HTTP Server, Apache Tomcat, OpenAM core server with OpenAM Console, and OpenAM Apache Policy Agent. Installation instructions for Java Development Kit (JDK) are not included in this chapter, as OpenAM is a Java web application, and the JDK is pre-installed.

- Java Development Kit

  OpenAM is a Java web application, and requires a Java Development Kit installed on the system where it runs.

  The OpenAM web policy agent installer is also a Java program.

- Apache HTTP Server

  Apache HTTP Server serves the web page OpenAM protects.

- Apache Tomcat

  Because OpenAM is a Java web application, it runs in a web container, in this case, Apache Tomcat.

- OpenAM core server with OpenAM Console

  This is the main web application for OpenAM. OpenAM sets up an OpenDJ directory server at configuration time to use, in this case, to hold OpenAM's configuration and to serve as an identity store and authentication service.

- OpenAM Apache Policy Agent

  Install a policy agent in Apache HTTP Server to intercept requests from users and enforce access policy decisions OpenAM makes. The policy agent intercepts requests from users, and enforces access policy decisions made by OpenAM. The policy agent enforces policy by redirecting users to OpenAM for authentication and by contacting OpenAM to get authorization decisions for resources, such as the web page to protect.

Follow the steps in the following sections of this chapter to learn how OpenAM protects a web site without changing the web site itself.

1.3. Setting Up the Software

This section includes the following procedures that detail how to set up OpenAM to protect a web page:

- Procedure 1.1, "To Prepare Your Hosts File"
- Procedure 1.2, "To Install Apache HTTP Server"
- Procedure 1.3, "To Install Apache Tomcat"
- Procedure 1.4, "To Install OpenAM"
- Procedure 1.5, "To Configure a Policy in OpenAM"
- Procedure 1.6, "To Create a Web Policy Agent Profile"
- Procedure 1.7, "To Install OpenAM Web Policy Agent"

The procedures in this section are written for use on a Linux system. If you are running Microsoft Windows, adapt the examples accordingly.

Procedure 1.1. To Prepare Your Hosts File

OpenAM requires that you use fully qualified domain names when protecting web resources. This is because OpenAM uses HTTP cookies to keep track of sessions for single sign-on (SSO), and setting and reading cookies depends on the server name and domain.

You can get started with OpenAM without setting up separate systems for each fully qualified domain name. Give your system `openam.example.com` and `www.example.com` aliases by editing your hosts file.

Alternatively, if you already have a DNS set up, you can use that instead of your hosts file.

- Add the aliases to your hosts file using your preferred text editor.

```
$ sudo vi /etc/hosts
Password:

### Edit /etc/hosts ###

$ cat /etc/hosts | grep openam
127.0.0.1    localhost openam.example.com www.example.com
```

Procedure 1.2. To Install Apache HTTP Server

Apache HTTP Server is a popular web server that is supported by OpenAM's web policy agents. Apache HTTP Server might already be installed on your system, but since you are installing software for the sole purpose of getting started with OpenAM, install the web server separately instead of modifying any existing installations.

Full installation instructions are available online.

1. Verify the correct tools are installed to build Apache HTTP Server 2.2 from source.

   For Linux distributions, you need development tools including the C compiler. How you install these depends on your distribution.

   For Red Hat and CentOS distributions:

   ```
   # yum groupinstall 'Development Tools'
   ```

   For Ubuntu distributions:

   ```
   $ sudo apt-get install build-essential checkinstall
   ```

2. Download Apache HTTP Server 2.2 sources from the Apache download page.

   The OpenAM web policy agent requires Apache Portable Runtime 1.3 or later, so make sure you download Apache HTTP Server 2.2.9 or later.

3. Extract the download.

4. Configure the sources for compilation.

   The `--prefix` option can be used to install the Web server in a location where you can write files.

   ```
   $ cd ~/Downloads/httpd-2.2.25
   $ ./configure --prefix=/path/to/apache
   ```

5. Compile Apache HTTP Server.

   ```
   $ make
   ```

6. Install Apache HTTP Server.

   ```
   $ make install
   ```

7. Edit the configuration to set the server name to `www.example.com` and the port to one, such as 8000 that the web server process can use when starting with your user ID.

   ```
   $ vi /path/to/apache/conf/httpd.conf
   $ grep 8000 /path/to/apache/conf/httpd.conf
   Listen 8000
   ServerName www.example.com:8000
   ```

8. Test the installation to ensure Apache HTTP Server is working.

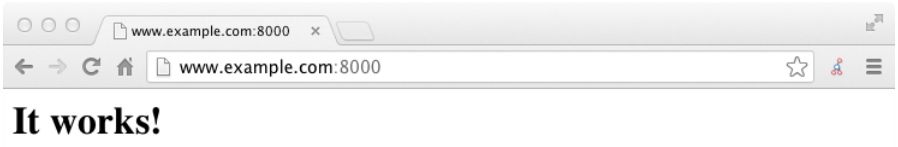
   a. Make sure that your system's firewall does not block the port that Apache HTTP Server uses.

See the documentation for your version of your system regarding how to allow traffic through the firewall on a specific port. A variety of firewalls are in use on Linux systems. The one in use depends on your specific distribution.

b. Start the web server.

```
$ /path/to/apache/bin/apachectl -k start
```

c. Point your browser to following URL: http://www.example.com:8000.



This is the page to protect with OpenAM. Do not proceed with the next steps unless this page appears.

Procedure 1.3. To Install Apache Tomcat

OpenAM runs as a Java web application inside an application container. Apache Tomcat is an application container that runs on a variety of platforms. The following instructions are loosely based on the `RUNNING.txt` file delivered with Tomcat.

1. Make sure you have a recent JDK release installed.

   One way of checking the version of the JDK is to list the version of the **javac** compiler.

   ```
   $ javac -version
   ```

   If the **javac** compiler is not found, then either you do not have a Java Development Kit installed, or it is installed, but not on your `PATH`.

   Section 2.2, "Java Requirements" in the *Release Notes* indicates what JDK versions are supported. Supported JDK versions also work for Tomcat.

2. Download Apache Tomcat 7 from its download page.

3. Extract the download.

   ```
   $ cd /path/to
   $ unzip ~/Downloads/apache-tomcat-7.0.42.zip
   $ mv apache-tomcat-7.0.42 tomcat
   ```

4. On UNIX-like systems, make the scripts in Tomcat's `bin/` directory executable.

```
$ chmod +x /path/to/tomcat/bin/*.sh
```

5. Set the `JAVA_HOME` environment variable to the file system location of the Java Development Kit.

   On Linux, set `JAVA_HOME` as follows.

   ```
   export JAVA_HOME=/path/to/jdk
   ```

6. Create a Tomcat `setenv.sh` (Unix/Linux) or `setenv.bat` (Windows) script to set the `JAVA_HOME` environment variable to the file system location of the Java Development Kit, and to set the heap and permanent generation size or metaspace size appropriately.

   If you are using JDK 7:

   ```
   export JAVA_HOME="/path/to/usr/jdk"
   export CATALINA_OPTS="$CATALINA_OPTS -Xmx2g -XX:MaxPermSize=256m"
   ```

   If you are using JDK 8:

   ```
   export JAVA_HOME="/path/to/usr/jdk"
   export CATALINA_OPTS="$CATALINA_OPTS -Xmx2g -XX:MaxMetaspaceSize=256m"
   ```

7. Make sure that your system's firewall does not block the port that Apache Tomcat uses.

   See the Apache documentation for instructions for allowing traffic through the firewall on a specific port for the version of Tomcat on your system. A variety of firewalls are in use on Linux systems. The version your system uses depends on your specific distribution.

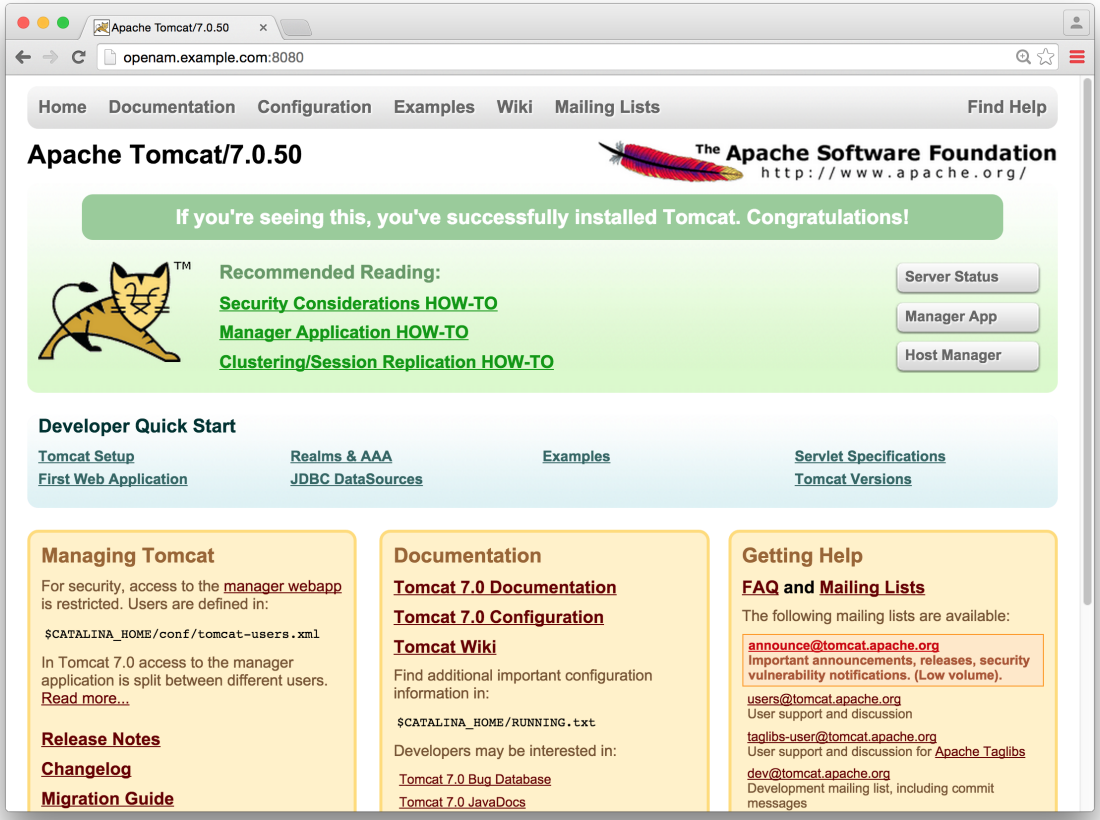
8. Start Tomcat.

   ```
   $ /path/to/tomcat/bin/startup.sh
   ```

   It might take Tomcat several seconds to start. When Tomcat has successfully started, you should see information indicating how long startup took in the `/path/to/tomcat/logs/catalina.out` log file.

   ```
   INFO: Server startup in 4655 ms
   ```

9. Browse to Tomcat's home page, such as `http://openam.example.com:8080`.



Tomcat will serve the OpenAM web application. Make sure you have successfully gotten to this point before you proceed.

Procedure 1.4. To Install OpenAM

Deploy OpenAm into Tomcat and then configure it for use.

1. Download the OpenAM `.war` file. Access the *ForgeRock* web site, and then click the Download tab. On the Your BackStage pass to ForgeRock resources page, click Downloads. On the Downloads page, click OpenAM. On the OpenAM Enterprise page, click `war` in the top-right corner, and then click Download to get the `.war` file.

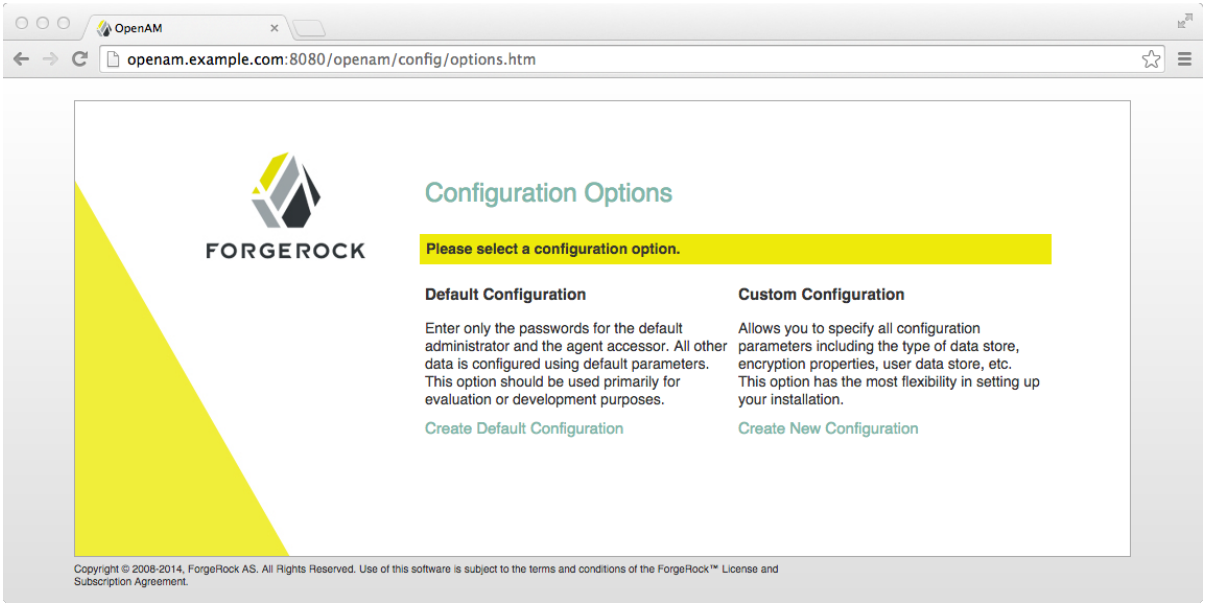
2. Deploy the `.war` file in Tomcat as `openam.war`.

```
$ mv ~/Downloads/OpenAM-13.0.0.war /path/to/tomcat/webapps/openam.war
```

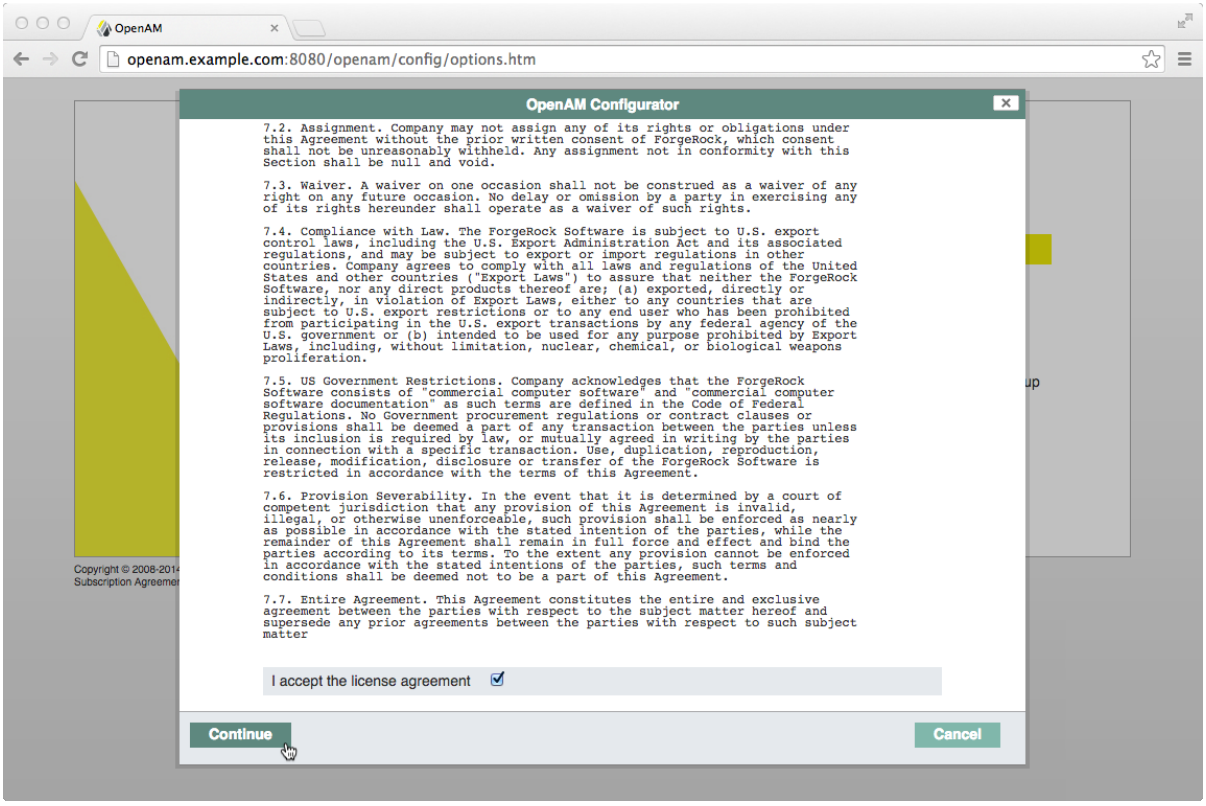
Tomcat deploys OpenAM under the `/path/to/tomcat/webapps/openam/` directory. You can access the web application in a browser at `http://openam.example.com:8080/openam/`.

3. Browse to OpenAM where it is deployed in Tomcat, in this example, `http://openam.example.com:8080/openam/`, to configure the application.

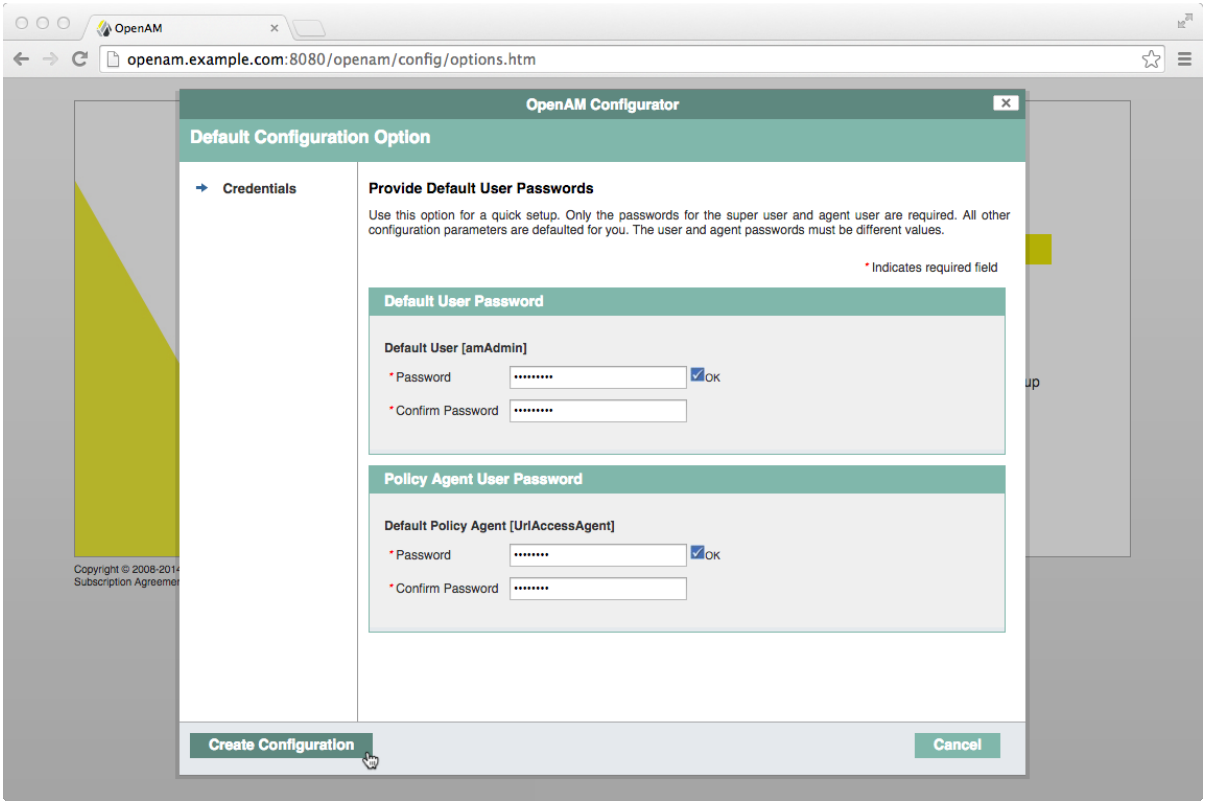
4. On the OpenAM home page, click Create Default Configuration.



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



6. Set the Default User [amAdmin] password to `changeit` and the Default Policy Agent [UrlAccessAgent] password to `secret12`, and then click Create Configuration to configure OpenAM.



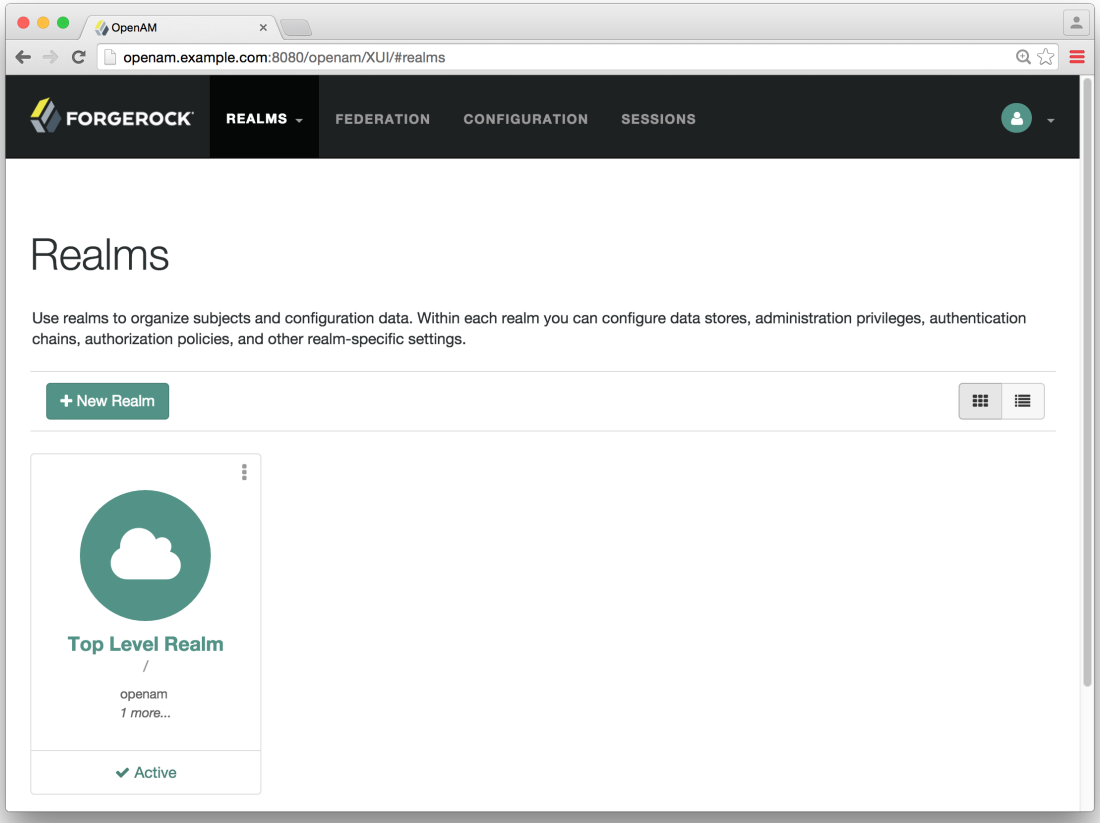
Note

If you were configuring OpenAM for real-world use, you would not use either of those passwords, but this is only to get started with OpenAM. The `amadmin` user is the OpenAM administrator, who is like a superuser in that `amadmin` has full control over the OpenAM configuration.

The `UrlAccessAgent` is not used in this guide.

7. Click the Proceed to Login link, then log in as `amadmin` with the password specified in the previous step, `changeit`.

   After login, OpenAM should direct you to the Realms page.



OpenAM stores its configuration, including the embedded OpenDJ directory server in the folder named `~/openam/` in your home directory. The folder shares the same name as your server instance. It also has a hidden folder, `~/.openamcfg/`, with a file used by OpenAM when it starts up. If you ruin your configuration of OpenAM somehow, the quickest way to start over is to stop Tomcat, delete these two folders, and configure OpenAM again.

OpenAM core server and OpenAM Console are now configured. Make sure you have successfully logged in to OpenAM Console before you proceed.

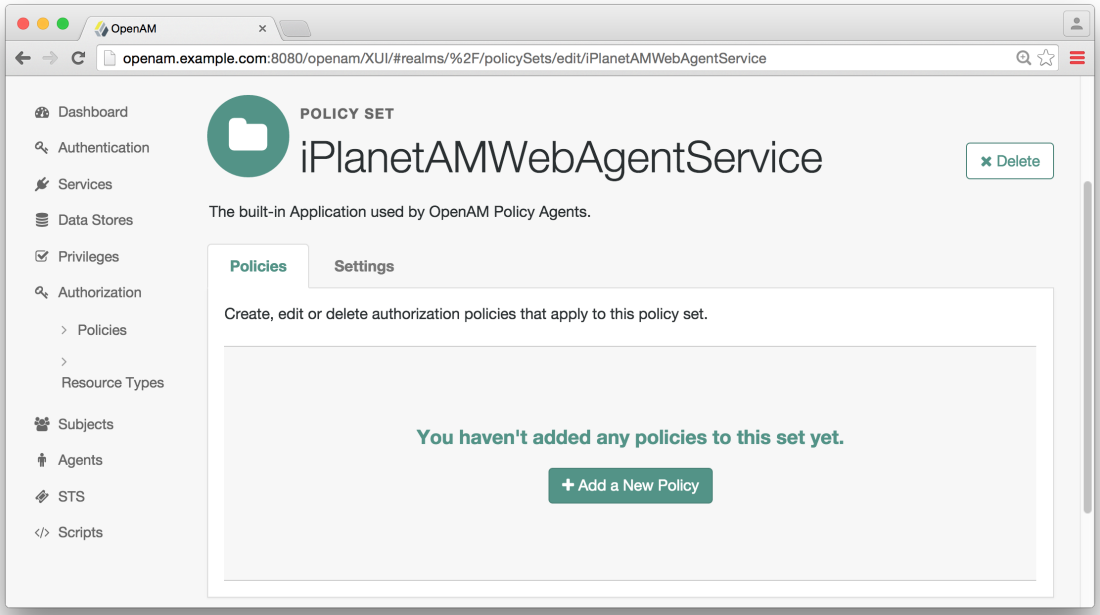
Procedure 1.5. To Configure a Policy in OpenAM

OpenAM authenticates users and then makes authorization decisions based on access policies that indicate user entitlements. Follow these steps to create a policy that allows all authenticated users to perform an HTTP GET (for example, to browse) the Apache HTTP home page that you set up earlier.

1. In the OpenAM console, select the Top Level Realm on the Realms page.

   OpenAM allows you to organize identities, policies, and policy agent profiles into realms as described in Chapter 4, "*Configuring Realms*" in the *Administration Guide*. For now, use the default Top Level Realm.

2. On the Realm Overview page, navigate to Authorization > Policy Sets > `Default Policy Set` > Add a Policy.



   For more information on the relationship between realms, policy sets, and policies, see Section 3.1, "About Authorization in OpenAM" in the *Administration Guide*.

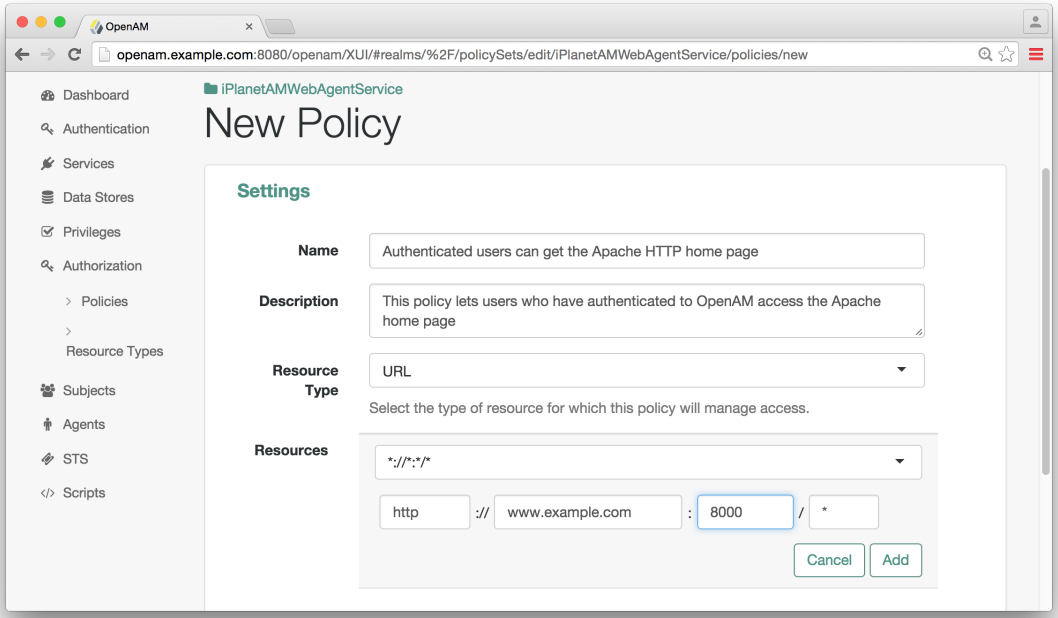
3. On the New Policy page, enter the following data:

   a. In the Name field, give your new policy the name `Authenticated users can get Apache HTTP home page`.

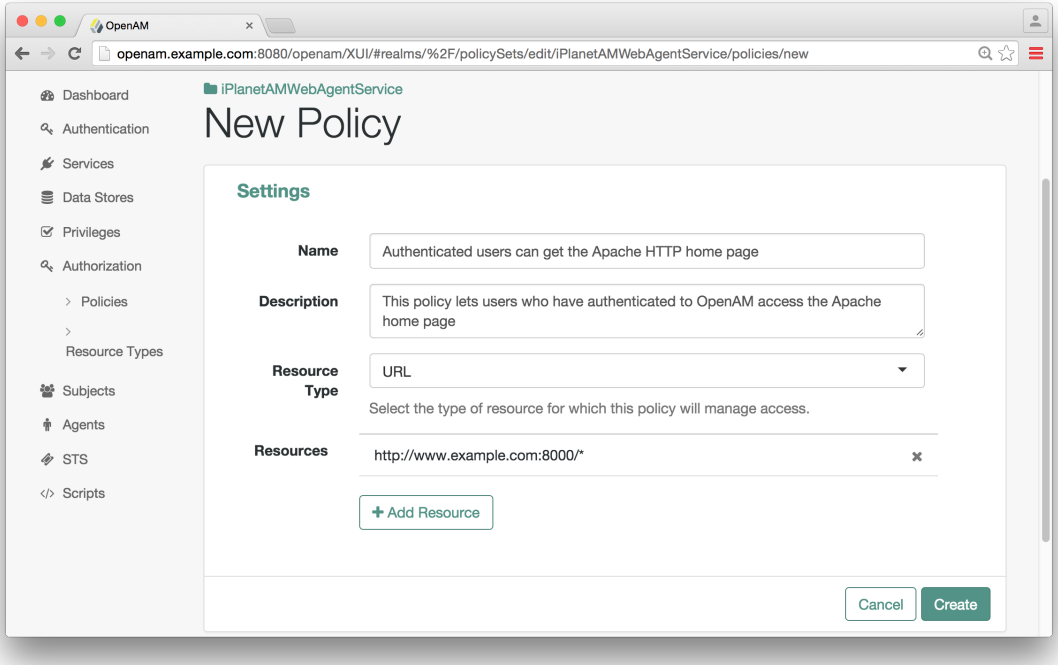
   b. In the Description field, enter a descriptive statement about the policy. For this example, enter `This policy lets users who have authenticated to OpenAM access to the Apache home page`.

c. On the Resource Type drop-down list, select `URL`.

d. On the Resources drop-down list, select the URL pattern for your policy. In this example, select `*://*:*/*`, then enter the resource URL: `http://www.example.com:8000/*`, and then click Add.



e. Click Create to save your settings.

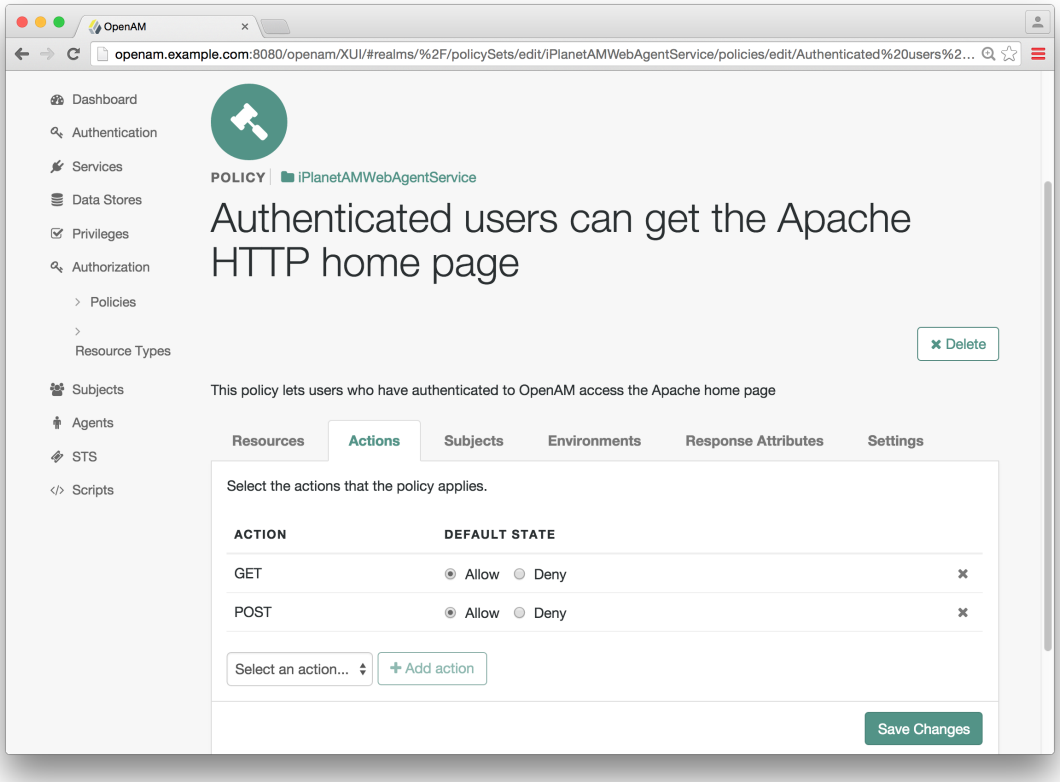


4. On your policy page, select the Actions tab, and then enter the following information:

   a. On the Add an action drop-down list, select `GET`.

   b. On the Add an action drop-down list, select `POST`.

   c. Save your changes.

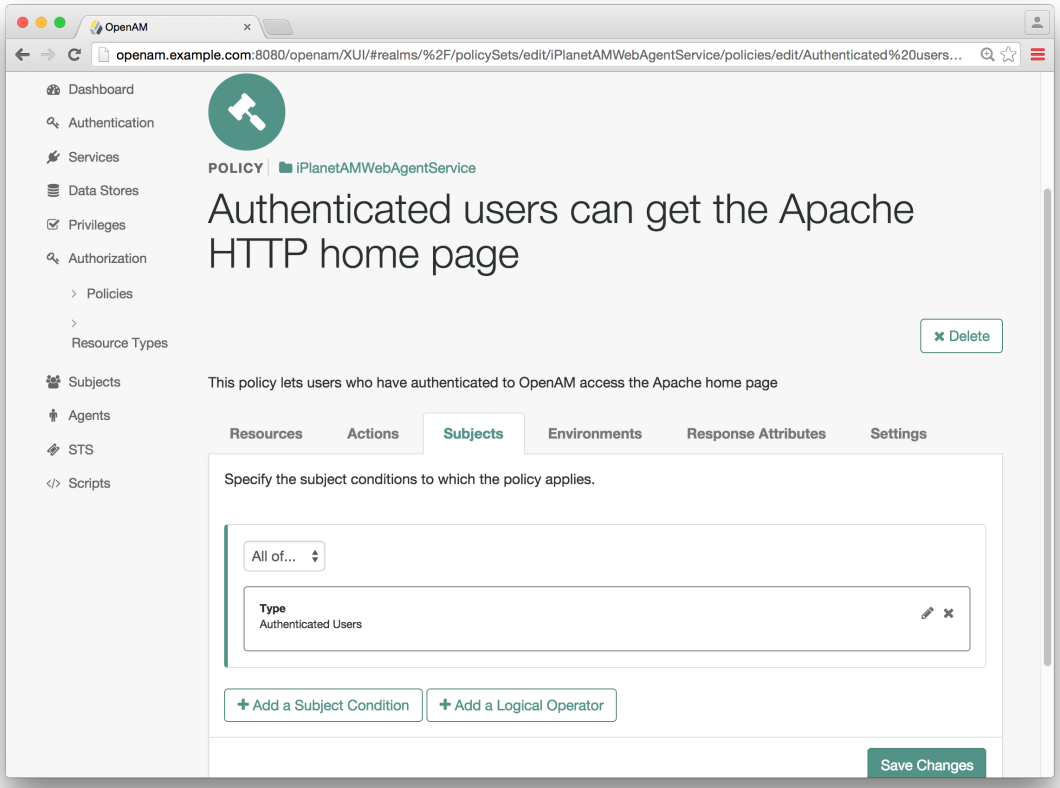


5. On your policy page, navigate to Subjects and enter the following data:

   a. On the All of drop-down list, review the list and select `All of...`.

   b. On the Type section, click the Edit icon. On the Type drop-down list, select `Authenticated Users`, and then click the checkmark.

   c. Save your changes.



6. Review your configuration. To make changes to the configuration, click the relevant tab and amend the configuration.

Next, you must create a web policy agent profile before installing the agent in Apache HTTP Server to enforce your new policy.

Procedure 1.6. To Create a Web Policy Agent Profile

OpenAM stores profile information about policy agents centrally by default. You can manage the policy agent profile through OpenAM Console. The policy agent retrieves its configuration from its OpenAM profile at installation and start up, and OpenAM notifies the policy agent of changes to its configuration. Follow these steps before installing the policy agent itself.

1. In OpenAM Console, browse to Realms > / Top Level Realm > Agents > Web, and then click New in the Agents table.

2. In the page to configure your new web policy agent, set the following values.

   Name

   `WebAgent`

   Password

   `password`

   Configuration

   Keep the default, `Centralized`

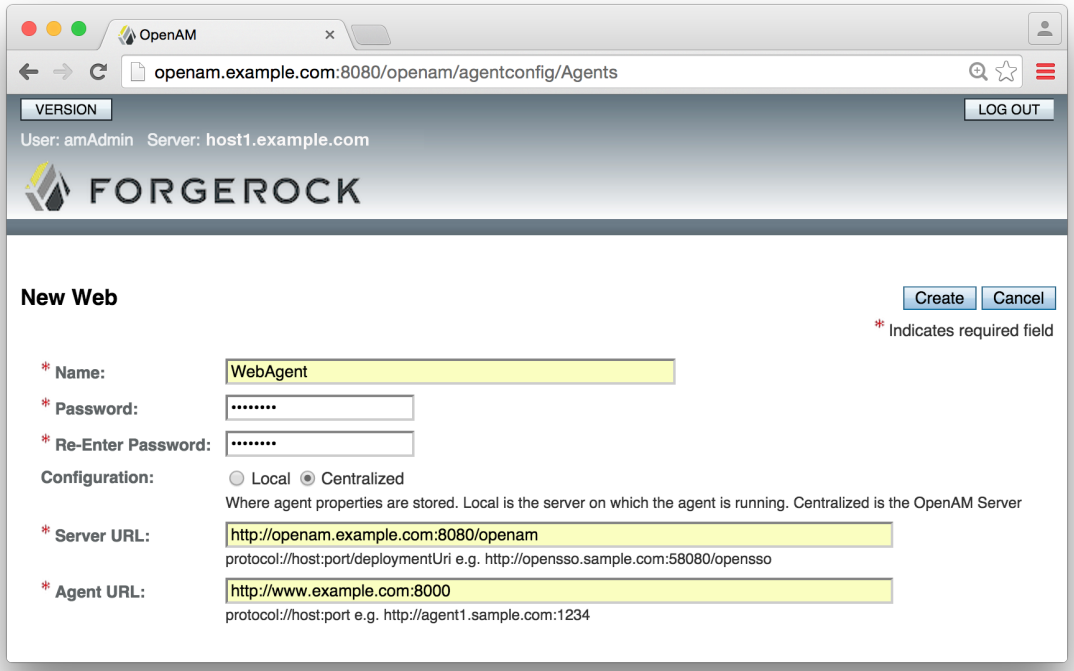
   Server URL

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

   Agent URL

   `http://www.example.com:8000`

   8000 is the port number you set previously for Apache HTTP Server.



3. Click Create to save the new web policy agent profile in OpenAM.

   Next, install a policy agent in Apache HTTP Server to enforce your new policy.

Procedure 1.7. To Install OpenAM Web Policy Agent

OpenAM policy agents enforce policies defined in OpenAM. While the policy agent's job is to verify that uses have the appropriate privileges to the resources they request, the policy agents do not make policy decisions. They call on OpenAM to make policy decisions using information presented by the user (or the user's client application), such as the SSO token in the HTTP cookie, which OpenAM uses to manage user sessions. A policy agent is, in essence, a gatekeeper for OpenAM.

The agent runs inside of Apache HTTP Server as a library, which the server loads at startup time. When a request comes in, the agent redirects users to OpenAM for authentication and calls on OpenAM for policy decisions as necessary.

1. Download the OpenAM policy agent for your version of Apache HTTP Server from the ForgeRock *Enterprise Downloads* page .

2. Create a password file, for example `$HOME/.pwd.txt`, that the agent installer reads when first connecting to OpenAM to read its profile. The file should only contain the password string, on a single line.

   The password file should be read-only by the user who installs the policy agent.

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

   The password is stored encrypted after installation.

3. Make sure OpenAM is running.

   You can verify this by logging into OpenAM Console.

4. Stop Apache HTTP Server while you install the policy agent.

   ```
   $ /path/to/apache/bin/apachectl stop
   ```

5. Extract the download.

   ```
   $ cd /path/to
   $ unzip ~/Downloads/Apache-v22-Linux-64-Agent-4.1.zip
   ```

6. Install the web policy agent in Apache HTTP Server, making sure that you provide the correct information to the installer as shown in the following example.

   When you run the command, you will be prompted to read and accept the software license agreement for the agent installation. You can suppress the license agreement prompt by including the `--acceptLicence` parameter. The inclusion of the option indicates that you have read and accepted the terms stated in the license. To view the license agreement, open `<server-root>/legal-notices/license.txt`.

   ```
   $ cd /path/to/web_agents/apache22_agent/bin
   $ ./agentadmin --install --acceptLicense
   ...

   ---------------------------------------------
   SUMMARY OF YOUR RESPONSES
   ---------------------------------------------
   Apache Server Config Directory : /path/to/apache/conf
   OpenAM server URL : http://openam.example.com:8080/openam
   Agent URL : http://www.example.com:8000
   Agent Profile name : WebAgent
   Agent Profile Password file name : $HOME/.pwd.txt

   ...
   ```

7. Start Apache HTTP Server, and verify that the web policy agent is configured correctly.

```
$ /path/to/apache/bin/apachectl -k start
$ tail /path/to/apache/logs/error_log
...[notice] Apache/2.2.25 (Unix) OpenAM WPA/4.1 configured -- resuming
 normal operations
```

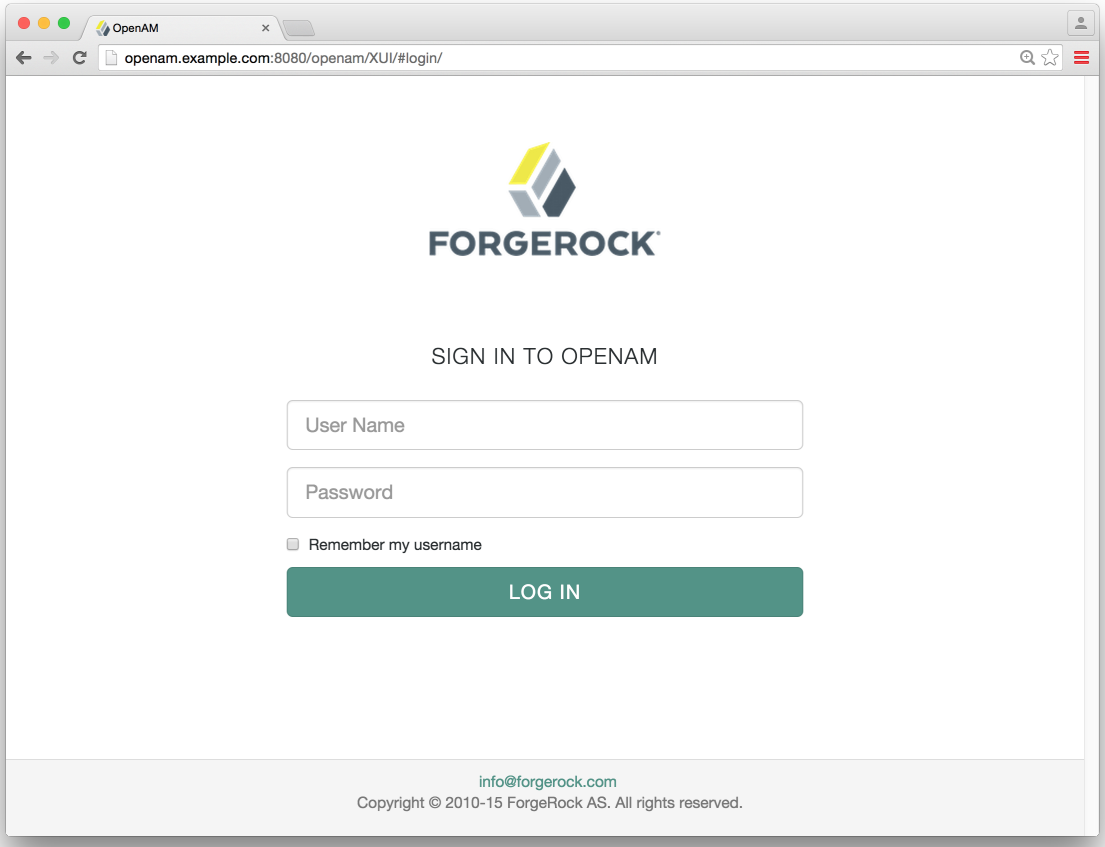
You can now try your installation to see OpenAM in action.

1.4. Trying It Out

Now that you have completed Section 1.3, "Setting Up the Software", you can access the protected web page to see OpenAM at work.

1. Log out of OpenAM Console.
2. Browse to `http://www.example.com:8000` to attempt to access the Apache "It works!" page.

   At this point, the policy agent intercepts your request for the page. Your browser does not return a cookie indicating an OpenAM session, so the policy agent redirects you to OpenAM to authenticate.
3. Log in as the built-in default OpenAM demonstration user `demo` with password `changeit`.



On successful login, OpenAM sets a session cookie named `iPlanetDirectoryPro` in your browser for the domain `.example.com`. The cookie is then returned to servers in the `example.com` domain, such as, `openam.example.com` and `www.example.com`.

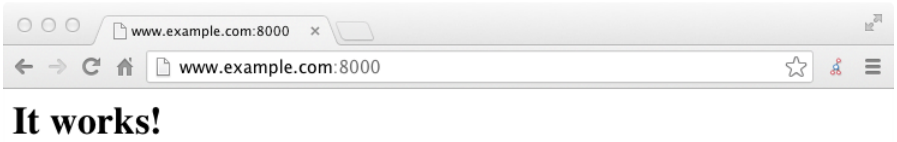
If you examine this cookie in your browser, you see that it has a value, such as `AQIC5wM2LY4SfcwciyfvJcQDUIB7kIWEH187Df_txqLdAVc.*AAJTSQACMDEAAlNLABMxMDYwNzY1MjQ0NTE0ODI2NTkx*`. This is the SSO Token value. The value is in fact an encrypted reference to the session that is stored only by OpenAM. So, only OpenAM can determine whether you are actually logged in, or instead, that the session is no longer valid and you need to authenticate again.

The OpenAM session is used for SSO. When the browser presents the cookie to a server in the domain, the agent on the server can check with OpenAM using the SSO Token as a reference to

the session. This lets OpenAM make policy decisions based on who is authenticated, or prompt for additional authentication, if necessary.

Your SSO session can end in a few ways. For example, when examining the cookie in your browser, you should notice that it expires when the browser session ends (when you shut down your browser). Alternatively, you can log out of OpenAM explicitly. Sessions can also expire. OpenAM sets two limits, one that causes your session to expire if it remains inactive for a configurable period of time (default: 30 minutes), and another that caps the session lifetime (default: 2 hours).

4. After successful login, you are redirected to the Apache "It works!" page.



In the background, OpenAM redirected your browser again to the page you tried to access originally, `http://www.example.com:8000`. This time, the web policy agent intercepted the request and found the SSO Token so it could request a policy decision from OpenAM regarding whether the user with the SSO Token has access to get `http://www.example.com:8000/`. OpenAM replied to the policy agent that it could allow access, and the policy agent allowed Apache HTTP Server to send back the web page.

Congratulations on protecting your first web site with OpenAM! Notice that you had only to install software and to configure OpenAM. You did not have to change your web site at all in order to add SSO and to set up access policies.

OpenAM can do much more than protect web pages. Read the next chapter to learn more.

1.5. Trying Out Stateless Sessions

In the Section 1.4, "Trying It Out" section, you successfully configured OpenAM and viewed the `iPlanetDirectoryPro` session cookie. The session cookie contains information for OpenAM or a policy agent to locate the session data object on the server from which the session originated. Sessions that are stored in a server's memory are called *stateful*, which is the default configuration at the realm level.

OpenAM also supports *stateless* sessions, in which the authenticated user's session is stored on the client-side (for example, in a browser), not in memory. The session cookie cannot be updated until the session ends, when the user logs out or the session expires.

To try out stateless sessions, see Chapter 9, "*Configuring Session State*" in the *Administration Guide*.

Chapter 2
Where To Go From Here

OpenAM can do much more than protect web pages. In addition to being the right foundation for building highly available, Internet-scale access management services, OpenAM has a rich set of features that make it a strong choice for a variety of different deployments. This chapter presents the key features of OpenAM and indicates where in the documentation you can find out more about them.

2.1. User Self-Service Features

OpenAM provides user self-registration and password reset services that allow users access to applications without the need to call your help desk.

OpenAM has access to the identity repositories where user profiles are stored. OpenAM is therefore well placed to help you manage self-service features that involve user profiles.

- **User Self-Registration**. OpenAM provides user self-registration as a feature of OpenAM's REST APIs. New users can easily self-register in OpenAM without assistance from administrators or help desk staff.

  For information on configuring self-registration, see Section 8.3.4, "Configuring User Self-Registration" in the *Administration Guide*.

  For details on building your own self-registration application using the REST API, see Section 2.1.4.1, "Registering Users" in the *Developer's Guide*.

- **Password Reset**. With OpenAM's self-service password reset, users can help reset passwords passwords, as well as update their existing passwords. OpenAM handles both the case where a user knows their password and wants to change it, and also the case where the user has forgotten their password and needs to reset it, possibly after answering security questions.

  For details on setting up password reset capabilities, see Section 8.3.5, "Configuring the Forgotten Password Reset Feature" in the *Administration Guide*.

  For details on building your own application to handle password reset using the REST API, see Section 2.1.4.2, "Retrieving Forgotten Usernames" in the *Developer's Guide*.

- **Dashboard Service**. Users often have a number of applications assigned to them, especially if your organization has standardized SaaS, for example for email, document sharing, support ticketing, customer relationship management, web conferencing, and so forth. You can create an interface for users to access these web-based and internal applications using OpenAM's dashboard service.

The OpenAM cloud dashboard service makes this relatively easy to set up. For basic information on using the service, see Chapter 18, "*Configuring the Dashboard Service*" in the *Administration Guide*.

OpenAM's user-facing pages are fully customizable and easy to skin for your organization. The Installation Guide has details on how to customize user-facing pages.

2.2. Single Sign-On

Single sign-on (SSO) is a core feature of OpenAM. Once you have set up OpenAM, you protect as many applications in the network domain as you want. Simply install policy agents for the additional servers, and add policies for the resources served by the applications. Users can authenticate to start a session on any site in the domain and stay authenticated for all sites in the domain without needing to log in again (unless the session ends, or unless a policy requires stronger authentication. For details, see Chapter 10, "*Configuring Single Sign-On Within One Domain*" in the *Administration Guide*.

Many organizations manage more than one domain. When you have multiple distinct domains in a single organization, cookies set in one domain are not returned to servers in another domain. In many organizations, sub-domains are controlled independently. These domains need to be protected from surreptitious takeovers like session cookie hijacking. OpenAM's cross-domain single sign-on (CDSSO) provides a safe mechanism for your OpenAM servers in one domain to work with policy agents from other domains, while allowing users to sign-on once across many domains without needing to reauthenticate. CDSSO allows users to sign on in one of your domains and not have to sign on again when they visit another of your domains.

CDSSO works through cooperation between policy agents and the `CDCServlet` in OpenAM. Together, the policy agents and OpenAM use federation capabilities to translate from one domain to another. For details on how to configure policy agents for CDSSO, see Chapter 11, "*Configuring Cross-Domain Single Sign-On*" in the *Administration Guide*.

Note

CDSSO only works with *stateful* sessions. CDSSO does not work with *stateless* sessions.

2.3. Standards-Based Federation

When you need to federate identities across different domains and different organizations with separate access management solutions, then you need interoperable federation technologies. Perhaps your organization acts as an identity provider for other organizations providing services. Perhaps you provide the services and allow users to use their identity from another organization to access your services. Either way, OpenAM has the capability to integrate well in federated access management scenarios.

OpenAM supports standards-based federation technologies.

- Security Assertion Markup Language (SAML) 2.0 grew out of earlier work on SAML v1.x and the Liberty Alliance. SAML defines XML-based, standard formats and profiles for federating identities. SAML v2.0 is supported by a wide range of applications including major software as a service (SaaS) offerings. OpenAM supports SAML v2.0 and earlier standards, and can function as a hub in deployments where different standards are used. For details on OpenAM's SAML v2.0 capabilities, see Chapter 12, "*Managing SAML v2.0 Federation*" in the *Administration Guide*.

  When your deployment serves as an identity provider for a SAML federation, OpenAM makes it easy to develop applications called Fedlets that your service providers can easily deploy to participate in the federation. For details see Chapter 3, "*Building SAML v2.0 Service Providers With Fedlets*" in the *Developer's Guide*.

- OAuth 2.0 and OpenID Connect 1.0 are open standards for authorization using REST APIs to allow users to authorize third-party access to their resources. These standards make it easier to federate modern web applications. OAuth for example is widely used in social applications.

  OpenAM offers support for both OAuth 2.0 and OpenID Connect 1.0. OpenAM can serve as an authorization server and as a client of OAuth 2.0, while managing the profiles of the resource owners. When acting as a client, OpenAM policy agents can be used on resource servers to enforce authorization. For details, see Chapter 13, "*Managing OAuth 2.0 Authorization*" in the *Administration Guide*.

  OpenAM can serve as the OpenID Connect 1.0 provider with support for Basic and Implicit client profiles as well as discovery, dynamic registration, and session management. For details, see Chapter 14, "*Managing OpenID Connect 1.0 Authorization*" in the *Administration Guide*.

2.4. Access Policies

In the first chapter of this guide you created an OpenAM access policy and saw how it worked. OpenAM can handle large numbers of access policies, each of which gives you control over user provisioning and user entitlements. For details, see Chapter 3, "*Defining Authorization Policies*" in the *Administration Guide*.

OpenAM also supports standards-based access policies defined using the eXtensible Access Control Markup Language (XACML). XACML defines an XML Attribute-Based Access Control (ABAC) language with Role-Based Access Control (RBAC) features as well. For details on using XACML policies with OpenAM, see Section 3.3.5, "Importing and Exporting Policies" in the *Administration Guide*.

2.5. Protect Any Web Application

In the first chapter of the guide you installed a web policy agent to enforce OpenAM's authorization decisions on Apache HTTP Server. That web policy agent is only one of many policy agents that work

with OpenAM. Chapter 5, "*Configuring Policy Agent Profiles*" in the *Administration Guide* describes policy agents for different web servers, for a variety of Java EE web application containers, for protecting SOAP-based web services, and for OAuth 2.0 clients.

For details about web policy agents also see the *Web Policy Agent User's Guide*.

For details about Java EE policy agents also see the *Java EE Policy Agent User's Guide*.

Furthermore OpenIG Identity Gateway works with applications where you want to protect access, but you cannot install a policy agent. For example, you might have a web application running in a server for which no policy agent has been developed. Or you might be protecting an application where you simply cannot install a policy agent. In that case, OpenIG functions as a flexible reverse proxy with standard SAML v2.0 capabilities. For details see the *OpenIG documentation*.

2.6. Modern APIs For Developers

For client application developers, OpenAM offers REST, Java, and C APIs.

- OpenAM REST APIs make the common CRUD (create, read, update, delete) easy to use in modern web applications. They also offer extended actions and query capabilities for access management functionality.

  To get started, see Section 2.1, "Using the REST API" in the *Developer's Guide*.

- OpenAM Java APIs provided through the OpenAM Java SDK let your Java and Java EE applications call on OpenAM for authentication and authorization in both OpenAM and federated environments.

  To get started, see Section 2.2, "Using the OpenAM Java SDK" in the *Developer's Guide*.

- The OpenAM C SDK provides APIs for native applications, such as new web server policy agents. The C SDK is built for Linux, Solaris, and Windows platforms.

  To get started, see Section 2.3, "Using the OpenAM C SDK" in the *Developer's Guide*.

OpenAM provides built-in support for many identity repositories, web servers and web application containers, access management standards, and all the flexible, configurable capabilities mentioned in this chapter. Yet, for some deployments you might still need to extend what OpenAM's capabilities. For such cases, OpenAM defines Service Provider Interfaces (SPIs) where you can integrate your own plugins. For a list of extension points, and some examples, see Section 1.2, "OpenAM SPIs" in the *Developer's Guide*.

2.7. Getting Help With Your Project

You can purchase OpenAM support subscriptions and training courses from ForgeRock and from consulting partners around the world and in your area. To contact ForgeRock, send mail to info@forgerock.com. To find a partner in your area, see http://forgerock.com/partners/find-a-partner/.

Index

A

Apache HTTP server
  installing, 3
Apache Tomcat
  installing, 5
APIs
  about, 26

H

hosts file
  preparing, 3

O

OpenAM
  Apache HTTP server, 3
  Apache Tomcat, 5
  APIs, 26
  hosts file, 3
  how it manages access, 1
  installing, 7
  policies, 11, 25
  protecting web apps, 25
  required software, 2
  setting up, 3
  Single sign-on, 24
  Standards-based federation, 24
  support, 26
  training, 26
  User self-service features, 23
  Verifying the installation, 20
  Web policy agent, 18
  Web policy agent profiles, 16
  Where to go after install, 23

P

policies
  accessing, 25
  configuring, 11

R

required software, 2

S

Single sign-on
  about, 24
Standards-based Federation
  about, 24
Support subscriptions, 26

T

training, 26

U

User self-service features
  about, 23

W

Web applications
  protecting, 25
Web policy agent
  creating profiles, 16
  installing, 18