Quick Start Guide

/ ForgeRock Access Management 6.5

Latest update: 6.5.5

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

Abstract

Quick introduction to ForgeRock® Access Management for new users and readers evaluating the product. ForgeRock Access Management provides authentication, authorization, entitlement, and federation software.

Table of Contents

Preface

The Quick Start Guide shows you how to quickly install and get started with ForgeRock Access Management.

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

About ForgeRock Identity Platform™ Software

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

Chapter 1
First Steps

This guide shows you how to quickly set up an instance 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 a web agent.

> **Important**
>
> You need a Linux, Solaris, or Windows system that can run the web agent (see the *ForgeRock Access Management Web Agents Release Notes* section, *Web 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 agent is not built for Apache HTTP Server on Mac OS X.

About ForgeRock Access Management

AM 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, AM provides an infrastructure for managing users, roles, and access to resources. In this chapter, you manage access to a single web page.

AM 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.

AM 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.

AM 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 AM's built-in identity repository and authentication modules to make it easier to get started.

AM centralizes authorization by letting you use AM to manage access policies separate from applications and resources. Instead of building access policy into a web application, you install an agent with the web application to request policy decisions from AM. This way you can avoid issues that could arise when developers must embed policy decisions into their applications. With AM, if policy changes or an issue is found after the application is deployed, you have only to change the policy definition in AM, not deploy a new version of the application. AM makes the authorization decisions, and web and Java agents enforce the decisions on AM's behalf.

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

Requirements To Try Out ForgeRock Access Management

This chapter shows the requirements for using AM and how to install the software needed to protect a web page. You will learn how to install Apache HTTP Server, Apache Tomcat, AM core server with the AM console, and AM Apache web agent. Installation instructions for Java Development Kit (JDK) are not included in this chapter, as AM is a Java web application, and the JDK is pre-installed.

- **Disk Space Requirements**. AM's distribution `.war` file includes the core server code with an embedded DS server, which stores AM's configuration data. By default, AM installs the embedded DS and its configuration settings in the `$HOME` directory of the user running the container where AM is installed. For example, on an environment using Apache Tomcat as the container, the embedded DS server is installed on the `tomcat` user's home directory, `/home/tomcat`.

  The DS server requires free disk space equal to or greater than 5 GB, plus 5% of the total size of the filesystem in the $HOME directory of the user running the container.

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

  The AM web agent installer is also a Java program.

- **Apache HTTP Server**. Apache HTTP Server serves the web page AM protects.

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

- **AM core server with the AM console**. This is the main web application for AM. AM sets up a DS server at configuration time to use, in this case, to hold AM's configuration and to serve as an identity store and authentication service.

- **AM Apache Web Agent**. Install a web agent in Apache HTTP Server to intercept requests from users and enforce access policy decisions AM makes. The web agent intercepts requests from users, and enforces access policy decisions made by AM. The web agent enforces policy by redirecting users to AM for authentication and by contacting AM 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 AM protects a web site without changing the web site itself.

Setting Up the Software

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

- "To Prepare Your Hosts File"
- "To Install Apache HTTP Server"
- "To Install Apache Tomcat"
- "To Install ForgeRock Access Management"
- "To Configure a Policy"
- "To Create a Web Agent Profile"
- "To Install a Web Agent"

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

To Prepare Your Hosts File

AM requires that you use fully qualified domain names when protecting web resources. This is because AM 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 AM 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
  ```

To Install Apache HTTP Server

Apache HTTP Server is a popular web server that is supported by AM's web 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 AM, 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 AM web 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.

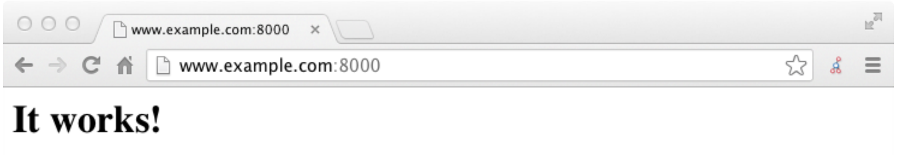
   a. Make sure that your system's firewall does not block the port that Apache HTTP Server uses.

      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.



It works!

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

To Install Apache Tomcat

AM 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`.

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

2. Download a supported version of Apache Tomcat.

   For information about supported versions, see "Web Application Container Requirements" in the *Release Notes*.

3. Extract the download.

   ```
   $ mkdir -p /path/to/tomcat
   $ unzip ~/Downloads/apache-tomcat-X.X.XX.zip -d /path/to/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 JDK, and to set the heap and metaspace size appropriately:

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

7. If you have a custom installation that differs from the documented Tomcat installation, make sure to set Tomcat's `CATALINA_TMPDIR` to a writable directory to ensure the installation succeeds. This temporary directory is used by the JVM (`java.io.tmpdir`) to write disk-based storage policies and other temporary files.

8. 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.

9. 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
   ```

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

    If Apache Tomcat works correctly, the "If you're seeing this, you've successfully installed Tomcat. Congratulations!" page appears.

    Tomcat is now able to serve the AM web application. Make sure you have successfully gotten to this point before you proceed.

To Install ForgeRock Access Management

Deploy AM into Tomcat and then configure it for use.

1. Download the AM `.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 AM. On the AM Enterprise page, click `war` in the top-right corner, and then click Download to get the `.war` file.

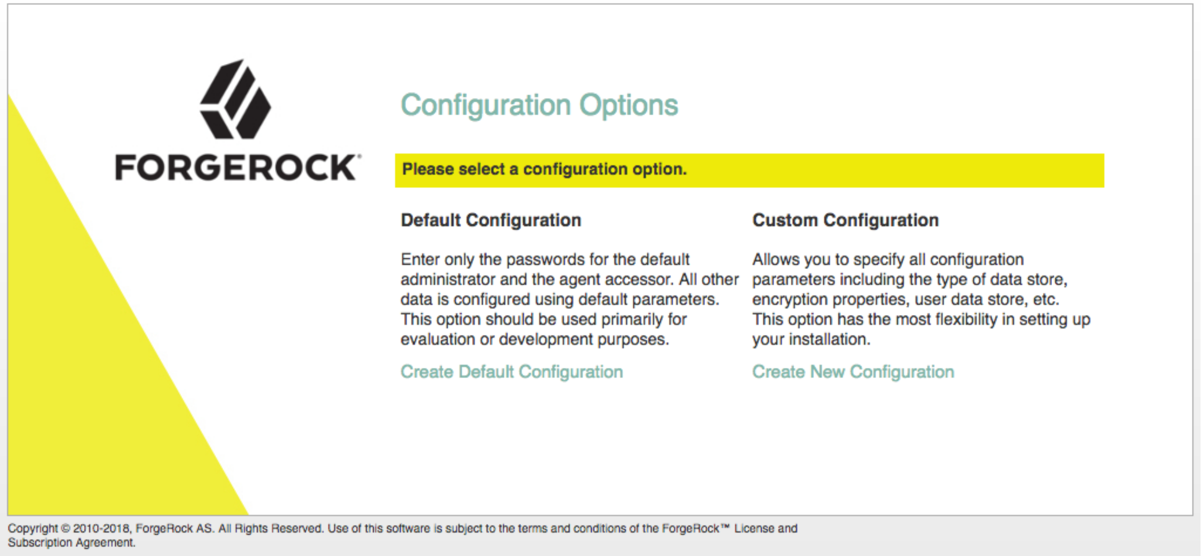
2. Deploy the `.war` file in Tomcat as `openam.war`.

   ```
   $ mv ~/Downloads/AM-6.5.5.war /path/to/tomcat/webapps/openam.war
   ```

   Tomcat deploys AM under the `/path/to/tomcat/webapps/openam/` directory. You can access the web application in a browser at `https://openam.example.com:8443/openam/`.

3. Browse to AM where it is deployed in Tomcat, in this example, `https://openam.example.com:8443/openam/`, to configure the application.

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



Configuration Options

Please select a configuration option.

Default Configuration

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

Create Default Configuration

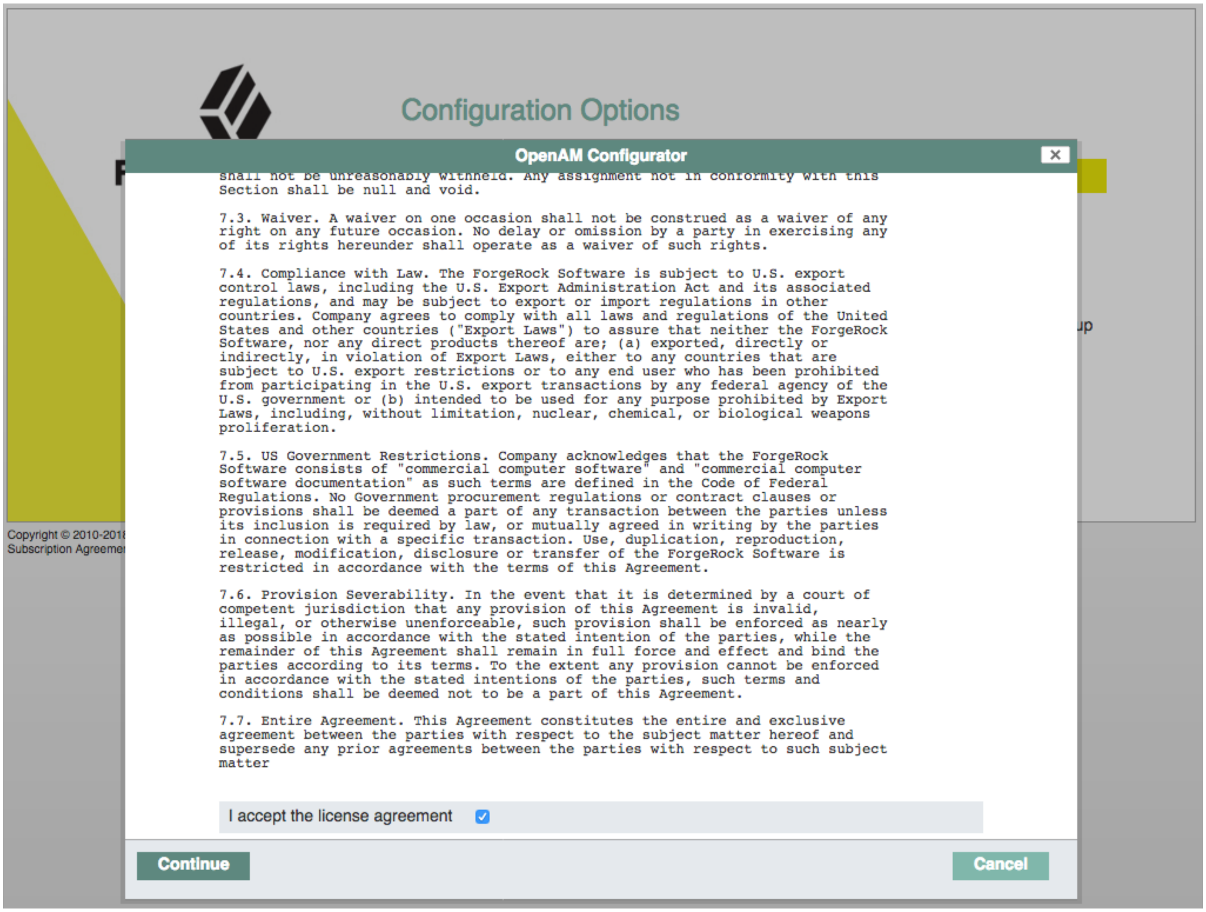
Custom Configuration

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

Create New Configuration

Copyright © 2010-2018, ForgeRock AS. All Rights Reserved. Use of this software is subject to the terms and conditions of the ForgeRock™ License and Subscription Agreement.

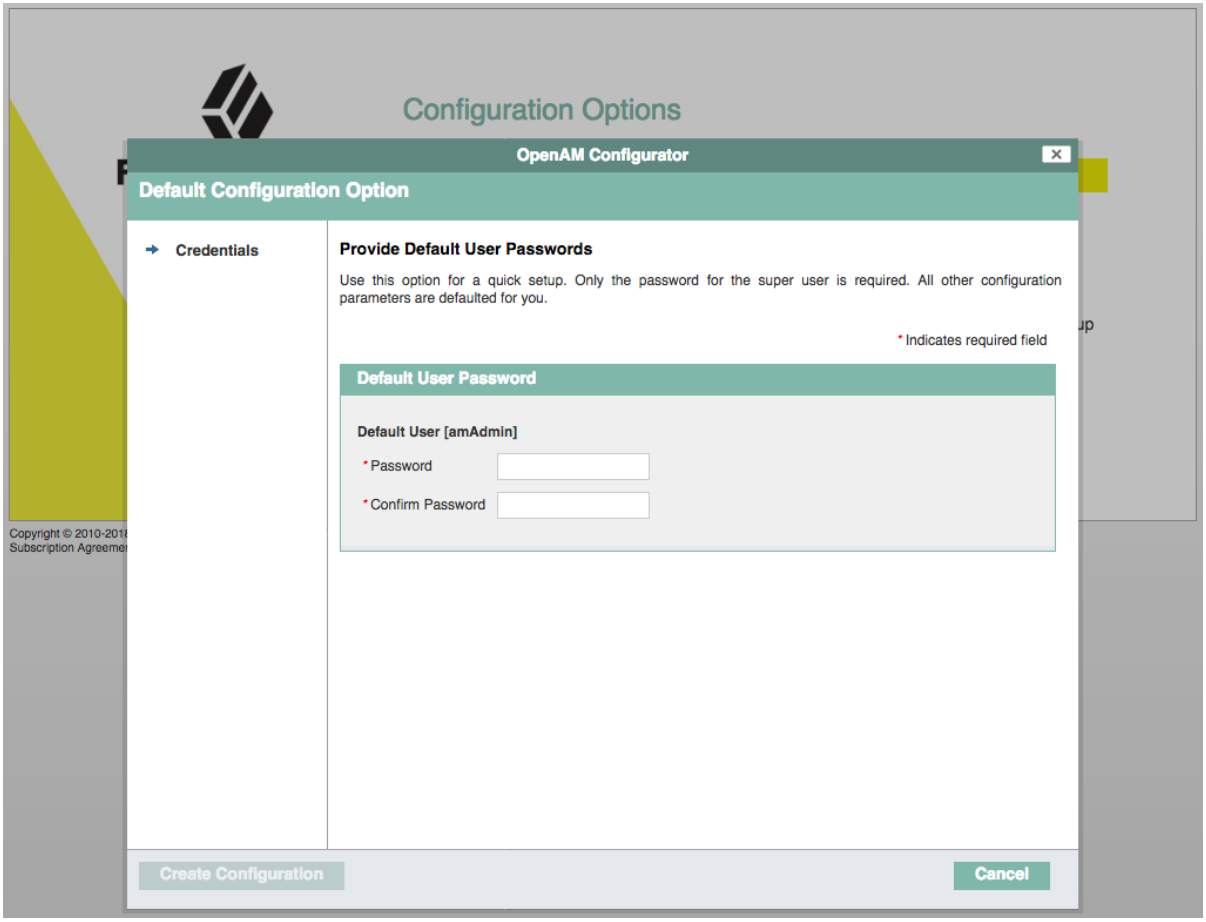
5. Review the software license agreement. If you agree to the license, click "I accept the license agreement", and then click Continue.



```
shall not be unreasonably withheld. Any assignment not in conformity with this
Section shall be null and void.

7.3. Waiver. A waiver on one occasion shall not be construed as a waiver of any
right on any future occasion. No delay or omission by a party in exercising any
of its rights hereunder shall operate as a waiver of such rights.

7.4. Compliance with Law. The ForgeRock Software is subject to U.S. export
control laws, including the U.S. Export Administration Act and its associated
regulations, and may be subject to export or import regulations in other
countries. Company agrees to comply with all laws and regulations of the United
States and other countries ("Export Laws") to assure that neither the ForgeRock
Software, nor any direct products thereof are; (a) exported, directly or
indirectly, in violation of Export Laws, either to any countries that are
subject to U.S. export restrictions or to any end user who has been prohibited
from participating in the U.S. export transactions by any federal agency of the
U.S. government or (b) intended to be used for any purpose prohibited by Export
Laws, including, without limitation, nuclear, chemical, or biological weapons
proliferation.

7.5. US Government Restrictions. Company acknowledges that the ForgeRock
Software consists of "commercial computer software" and "commercial computer
software documentation" as such terms are defined in the Code of Federal
Regulations. No Government procurement regulations or contract clauses or
provisions shall be deemed a part of any transaction between the parties unless
its inclusion is required by law, or mutually agreed in writing by the parties
in connection with a specific transaction. Use, duplication, reproduction,
release, modification, disclosure or transfer of the ForgeRock Software is
restricted in accordance with the terms of this Agreement.

7.6. Provision Severability. In the event that it is determined by a court of
competent jurisdiction that any provision of this Agreement is invalid,
illegal, or otherwise unenforceable, such provision shall be enforced as nearly
as possible in accordance with the stated intention of the parties, while the
remainder of this Agreement shall remain in full force and effect and bind the
parties according to its terms. To the extent any provision cannot be enforced
in accordance with the stated intentions of the parties, such terms and
conditions shall be deemed not to be a part of this Agreement.

7.7. Entire Agreement. This Agreement constitutes the entire and exclusive
agreement between the parties with respect to the subject matter hereof and
supersede any prior agreements between the parties with respect to such subject
matter
```

I accept the license agreement ☑

6. Set the Default User [amAdmin] password to `changeit`, and then click Create Configuration to configure AM.



Provide Default User Passwords

Use this option for a quick setup. Only the password for the super user is required. All other configuration parameters are defaulted for you.

* Indicates required field

Default User Password

Default User [amAdmin]

* Password

* Confirm Password

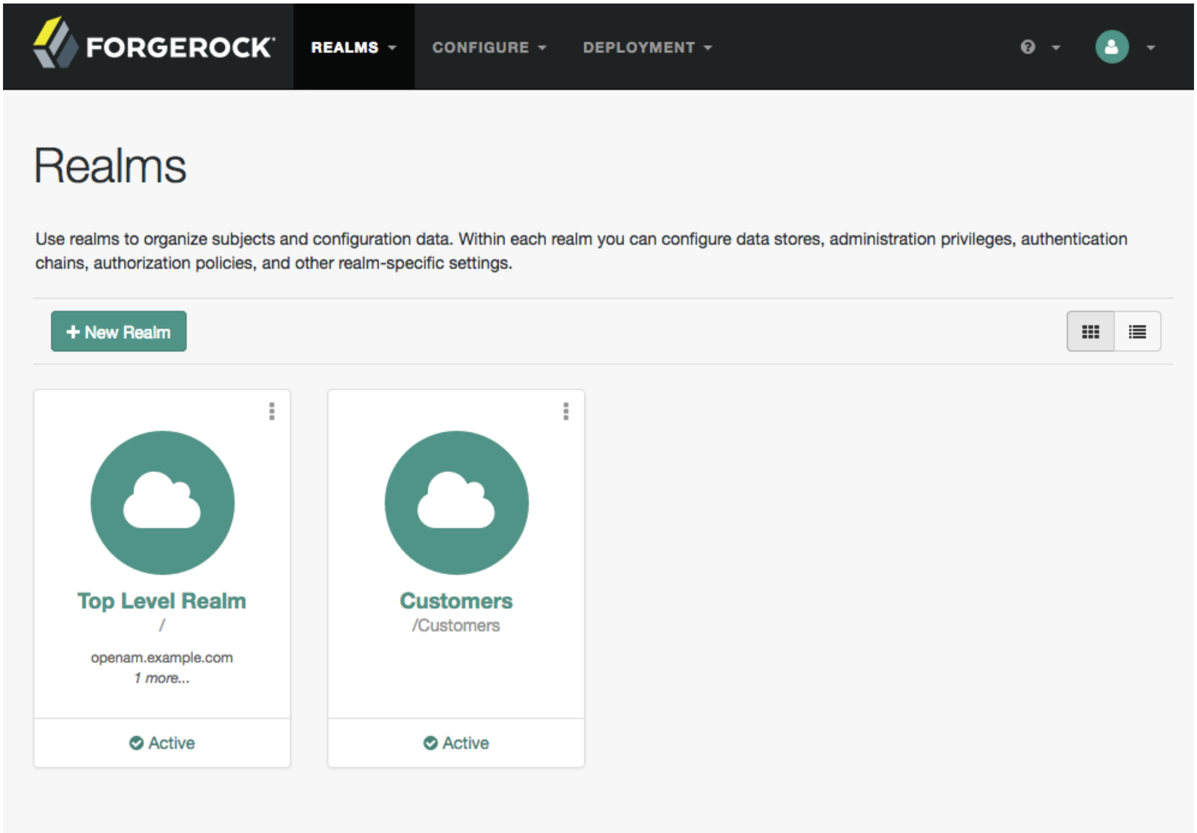
Copyright © 2010-201
Subscription Agreeme

Note

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

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

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



Realms

Use realms to organize subjects and configuration data. Within each realm you can configure data stores, administration privileges, authentication chains, authorization policies, and other realm-specific settings.

AM stores its configuration, including the embedded DS 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 AM when it starts up. If you ruin your configuration of AM somehow, the quickest way to start over is to stop Tomcat, delete these two folders, and configure AM again.

AM core server and the AM console are now configured. Make sure you have successfully logged in to the AM console before you proceed.

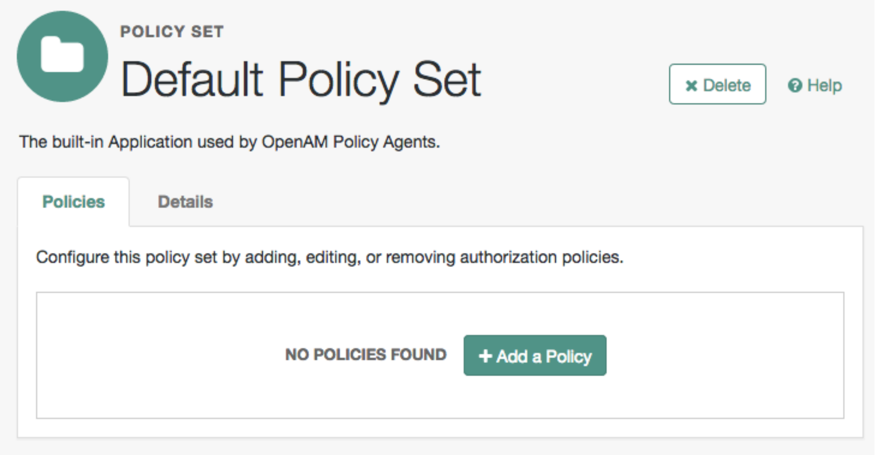
To Configure a Policy

AM 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 AM console, select the Top Level Realm on the Realms page.

AM allows you to organize identities, policies, and agent profiles into realms as described in "*Setting Up Realms*" in the *Setup and Maintenance 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.



The built-in Application used by OpenAM Policy Agents.

Configure this policy set by adding, editing, or removing authorization policies.

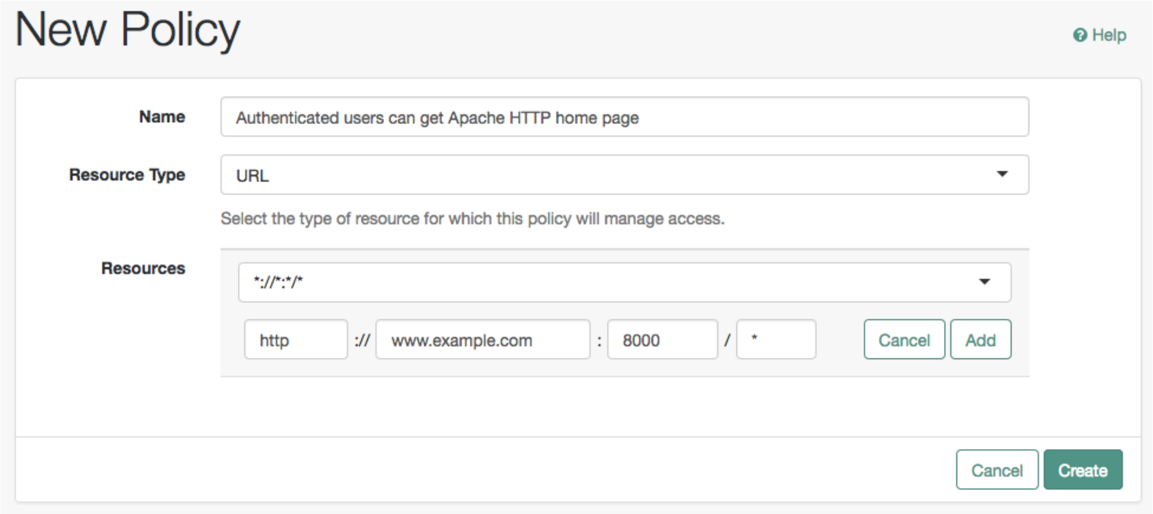
For more information on the relationship between realms, policy sets, and policies, see "Resource Types, Policy Sets, and Policies" in the *Authorization Guide*.

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

   c. From 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.



New Policy

Name: Authenticated users can get Apache HTTP home page

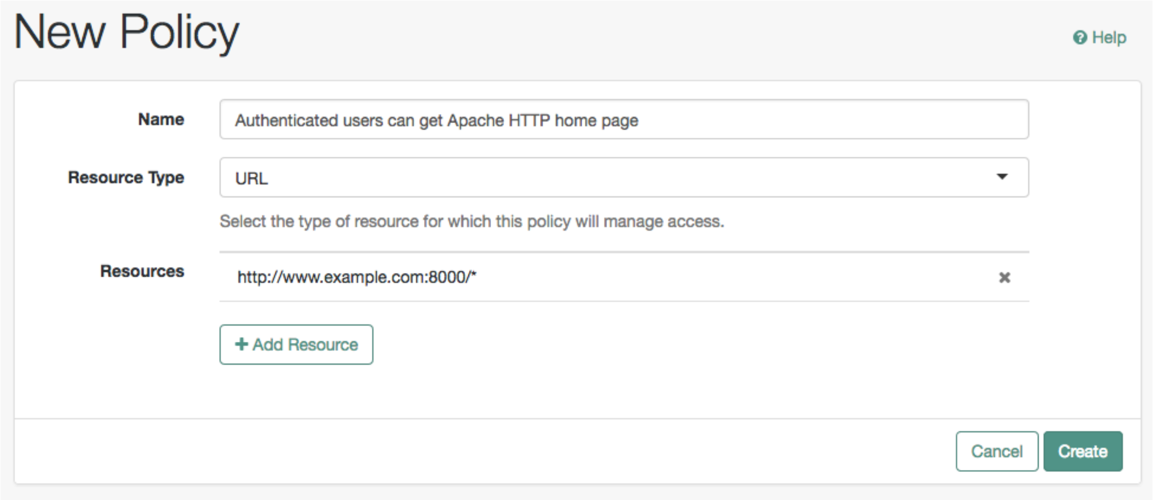
Resource Type: URL

Select the type of resource for which this policy will manage access.

Resources: *://*:*/*

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

d. Click Create to save your settings.



New Policy

Name: Authenticated users can get Apache HTTP home page

Resource Type: URL

Select the type of resource for which this policy will manage access.

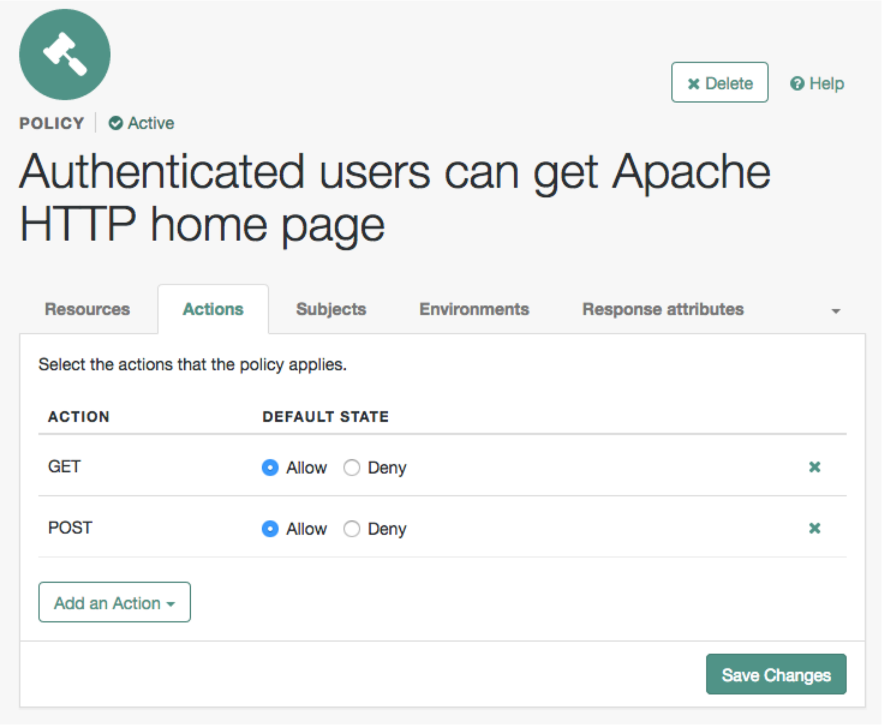
Resources: http://www.example.com:8000/*

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

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

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

   c. Save your changes.



POLICY | Active

Authenticated users can get Apache HTTP home page

Select the actions that the policy applies.

ACTION	DEFAULT STATE
GET	● Allow ○ Deny
POST	● Allow ○ Deny

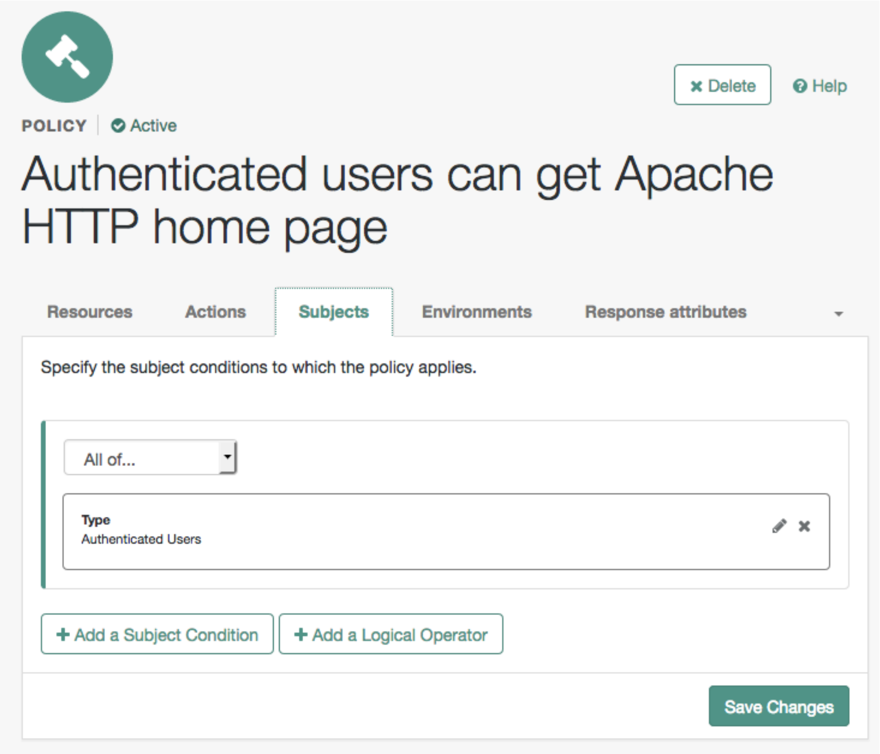
5. On your policy page, navigate to Subjects and enter the following data:

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

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

      Note that a subject condition that applies to all authenticated users includes the anonymous user.

   c. Save your changes.



POLICY | Active

Authenticated users can get Apache HTTP home page

Specify the subject conditions to which the policy applies.

All of...

Type
Authenticated Users

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

To Create a Web Agent Profile

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

1. In the AM console, navigate to Realms > *Realm Name* > Applications > Agents > Web, and select New in the Agents table.

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

   Name

   `WebAgent`

Password

`password`

Configuration

Keep the default, `Centralized`

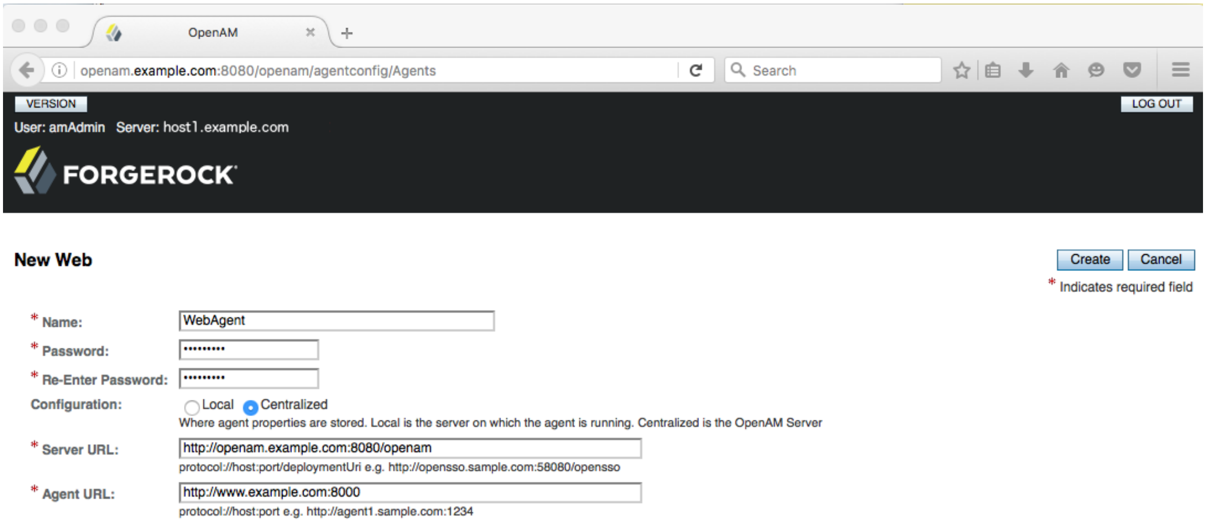
Server URL

`https://openam.example.com:8443/openam`

Agent URL

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

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


OpenAM
openam.example.com:8080/openam/agentconfig/Agents
Search
VERSION
LOG OUT
User: amAdmin Server: host1.example.com
FORGEROCK
Create
Cancel

New Web

* Indicates required field

* Name: WebAgent

* Password: •••••••••

* Re-Enter Password: •••••••••

Configuration: ○ Local ● Centralized

Where agent properties are stored. Local is the server on which the agent is running. Centralized is the OpenAM Server

* Server URL: http://openam.example.com:8080/openam

protocol://host:port/deploymentUri e.g. http://opensso.sample.com:58080/opensso

* Agent URL: http://www.example.com:8000

protocol://host:port e.g. http://agent1.sample.com:1234

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

Adjust the Cookie Domain

By default, AM installer sets the cookie domain based on the fully-qualified hostname on the server on which it installs AM. During your installation, it sets the cookie domain to `.openam.example.com`.

To allow the agent, which runs on `www.example.com`, to accept AM's `iPlanetDirectoryPro` cookie, you must change the AM cookie domain from `.openam.example.com` to `example.com`.

1. In the AM console, navigate to Configure > Global Services > Platform > Cookie Domain.
2. In the Cookie Domain field, change the domain to `example.com`.
3. Click Save Changes to apply your changes.

To Install a Web Agent

Web agents enforce policies defined in ForgeRock Access Management. While the web agent's job is to verify that users have the appropriate privileges to the resources they request, web agents do not make policy decisions. They call on AM 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 AM uses to manage user sessions. A web or Java agent is, in essence, a gatekeeper for AM.

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 AM for authentication and calls on AM for policy decisions as necessary.

1. Download the AM web agent for your version of Apache HTTP Server from the *ForgeRock BackStage* website.
2. Create a password file, for example `$HOME/.pwd.txt`, that the agent installer reads when first connecting to AM 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 web agent.

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

   The password is stored encrypted after installation.
3. Make sure AM is running.

   You can verify this by logging in to the AM console.
4. Stop Apache HTTP Server while you install the web agent.

   ```
   $ /path/to/apache/bin/apachectl stop
   ```
5. Extract the download.

   ```
   $ cd /path/to
   $ unzip ~/Downloads/Apache-v22-Linux-64-Agent-5.10.0.zip
   ```
6. Install the web 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 : https://openam.example.com:8443/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 agent is configured correctly.

   ```
   $ /path/to/apache/bin/apachectl -k start
   $ tail /path/to/apache/logs/error_log
   ...[notice] Apache/2.2.25 (Unix) AM WPA/5.10.0 configured -- resuming
    normal operations
   ```

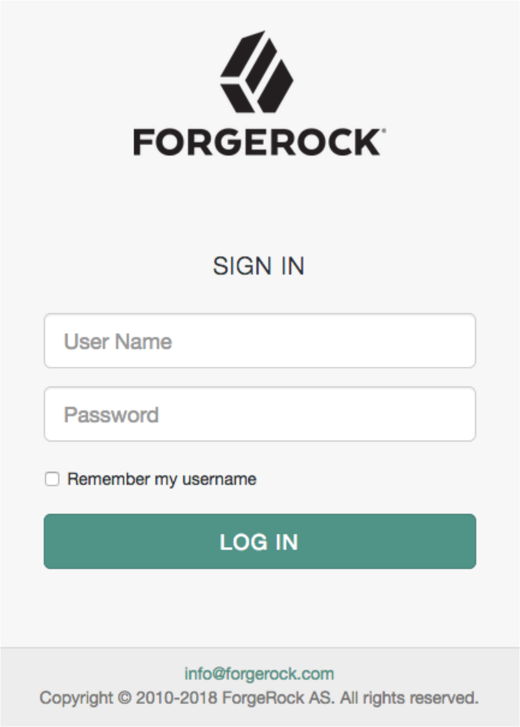
   You can now try your installation to see AM in action.

Trying It Out

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

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

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



info@forgerock.com
Copyright © 2010-2018 ForgeRock AS. All rights reserved.

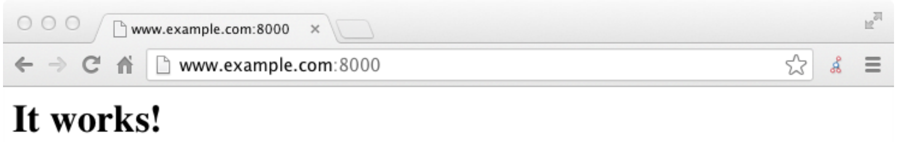
On successful login, AM sets a session cookie named `iPlanetDirectoryPro` in your browser for the domain `example.com`. The cookie is then returned to servers in the `example.com` domain, such as, `openam.example.com` and `www.example.com`.

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 AM. So, only AM can determine whether you are actually logged in, or instead, that the session is no longer valid and you need to authenticate again.

The AM 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 AM using the SSO Token as a reference to the session. This lets AM 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 AM explicitly. Sessions can also expire. AM 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.



It works!

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

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

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

Trying Out Client-Based Sessions

In the "Trying It Out" section, you successfully configured AM and viewed the `iPlanetDirectoryPro` session cookie. The session cookie contains information for AM or a web or Java agent to locate the session data object on the server. By default, sessions are stored in the Core Token Service (CTS) token store.

AM also supports *client-based* sessions, in which the authenticated user's session is stored on the client (for example, in an HTTP browser cookie), not on the server. The session cookie cannot be updated until the session ends, when the user logs out or the session expires.

To try out client-based sessions, see "Implementing Client-Based Sessions" in the *Authentication and Single Sign-On Guide*.

Chapter 2
Where To Go From Here

AM can do much more than protect web pages. In addition to being the right foundation for building highly available, Internet-scale access management services, AM 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 AM and indicates where in the documentation you can find out more about them.

User Self-Service Features

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

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

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

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

  For details on building your own self-registration application using the REST API, see "Registering Users" in the *User Self-Service Guide*.

- **Password Reset**. With AM's self-service password reset, users can help reset passwords, as well as update their existing passwords. AM 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 "Configuring the Forgotten Password Reset Feature" in the *User Self-Service Guide*.

  For details on building your own application to handle password reset using the REST API, see "Retrieving Forgotten Usernames" in the *User Self-Service 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 AM's dashboard service.

The AM cloud dashboard service makes this relatively easy to set up. For basic information on using the service, see "*Setting Up the Dashboard Service*" in the *Setup and Maintenance Guide*.

AM'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.

Single Sign-On

Single sign-on (SSO) and cross-domain single sign-on (CDSSO) are core features of AM. Once you have set up AM, you protect as many applications in the network domain as you want. Simply install web or Java 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.

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. AM's CDSSO provides a safe mechanism for your AM servers in one domain to work with web or Java 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.

For details on how to configure web and Java agents for CDSSO, see "Implementing Cross-Domain Single Sign-On" in the *Authentication and Single Sign-On Guide*.

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, AM has the capability to integrate well in federated access management scenarios.

AM 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. AM supports SAML v2.0 and earlier standards, and can function as a hub in deployments where different standards are used. For details on AM's SAML v2.0 capabilities, see the SAML v2.0 Guide.

When your deployment serves as an identity provider for a SAML federation, AM makes it easy to develop applications called Fedlets that your service providers can easily deploy to participate in the federation. For details see "*Implementing SAML v2.0 Service Providers by Using Fedlets*" in the *SAML v2.0 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.

  AM offers support for both OAuth 2.0 and OpenID Connect 1.0. AM 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, AM web and Java agents can be used on resource servers to enforce authorization. For details, see the OAuth 2.0 Guide.

  AM 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 the OpenID Connect 1.0 Guide.

Access Policies

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

AM 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 AM, see "Importing and Exporting Policies" in the *Authorization Guide*.

Protect Any Web Application

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

For details about web agents, see the Web Policy Agents documentation.

For details about Java agents, see the Java Policy Agents documentation.

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

Modern APIs For Developers

For client application developers, AM offers REST and Java APIs.

- AM 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 "*Developing with the REST API*" in the *Development Guide*.

- AM Java APIs let your Java and Java applications call on AM for authentication and authorization in both AM and federated environments.

  For details, see the *ForgeRock Access Management Java API Specification*.

AM 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 AM's capabilities. For such cases, AM defines Service Provider Interfaces (SPIs) where you can integrate your own plugins. For information about extension points, and some examples, see the following:

- Customizing Authentication in the *Authentication and Single Sign-On Guide*
- Customizing Policy Evaluation With a Plug-In in the *Authorization Guide*
- Customizing Identity Data Stores in the *Setup and Maintenance Guide*
- Customizing OAuth 2.0 Scope Handling in the *OAuth 2.0 Guide*

Appendix A. Getting Support

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 on ForgeRock's support offering, including support plans and service level agreements (SLAs), visit https://www.forgerock.com/support.

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.

Glossary

Access control

Control to grant or to deny access to a resource.

Account lockout

The act of making an account temporarily or permanently inactive after successive authentication failures.

Actions

Defined as part of policies, these verbs indicate what authorized identities can do to resources.

Advice

In the context of a policy decision denying access, a hint to the policy enforcement point about remedial action to take that could result in a decision allowing access.

Agent administrator

User having privileges only to read and write agent profile configuration information, typically created to delegate agent profile creation to the user installing a web or Java agent.

Agent authenticator

Entity with read-only access to multiple agent profiles defined in the same realm; allows an agent to read web service profiles.

Application

In general terms, a service exposing protected resources.

In the context of AM policies, the application is a template that constrains the policies that govern access to protected resources. An application can have zero or more policies.

Application type

Application types act as templates for creating policy applications.

Application types define a preset list of actions and functional logic, such as policy lookup and resource comparator logic.

Application types also define the internal normalization, indexing logic, and comparator logic for applications.

Attribute-based access control (ABAC)

Access control that is based on attributes of a user, such as how old a user is or whether the user is a paying customer.

Authentication

The act of confirming the identity of a principal.

Authentication chaining

A series of authentication modules configured together which a principal must negotiate as configured in order to authenticate successfully.

Authentication level

Positive integer associated with an authentication module, usually used to require success with more stringent authentication measures when requesting resources requiring special protection.

Authentication module

AM authentication unit that handles one way of obtaining and verifying credentials.

Authorization

The act of determining whether to grant or to deny a principal access to a resource.

Authorization Server

In OAuth 2.0, issues access tokens to the client after authenticating a resource owner and confirming that the owner authorizes the client to access the protected resource. AM can play this role in the OAuth 2.0 authorization framework.

Auto-federation

Arrangement to federate a principal's identity automatically based on a common attribute value shared across the principal's profiles at different providers.

Bulk federation

Batch job permanently federating user profiles between a service provider and an identity provider based on a list of matched user identifiers that exist on both providers.

Circle of trust

Group of providers, including at least one identity provider, who have agreed to trust each other to participate in a SAML v2.0 provider federation.

Client

In OAuth 2.0, requests protected web resources on behalf of the resource owner given the owner's authorization. AM can play this role in the OAuth 2.0 authorization framework.

Client-based OAuth 2.0 tokens

After a successful OAuth 2.0 grant flow, AM returns a token to the client. This differs from CTS-based OAuth 2.0 tokens, where AM returns a *reference* to token to the client.

Client-based sessions

AM sessions for which AM returns session state to the client after each request, and require it to be passed in with the subsequent request. For browser-based clients, AM sets a cookie in the browser that contains the session information.

For browser-based clients, AM sets a cookie in the browser that contains the session state. When the browser transmits the cookie back to AM, AM decodes the session state from the cookie.

Conditions

Defined as part of policies, these determine the circumstances under which which a policy applies.

Environmental conditions reflect circumstances like the client IP address, time of day, how the subject authenticated, or the authentication level achieved.

Subject conditions reflect characteristics of the subject like whether the subject authenticated, the identity of the subject, or claims in the subject's JWT.

Configuration datastore

LDAP directory service holding AM configuration data.

Cross-domain single sign-on (CDSSO)

AM capability allowing single sign-on across different DNS domains.

CTS-based OAuth 2.0 tokens

After a successful OAuth 2.0 grant flow, AM returns a *reference* to the token to the client, rather than the token itself. This differs from client-based OAuth 2.0 tokens, where AM returns the entire token to the client.

CTS-based sessions

AM sessions that reside in the Core Token Service's token store. CTS-based sessions might also be cached in memory on one or more AM servers. AM tracks these sessions in order to handle events like logout and timeout, to permit session constraints, and to notify applications involved in SSO when a session ends.

Delegation

Granting users administrative privileges with AM.

Entitlement

Decision that defines which resource names can and cannot be accessed for a given identity in the context of a particular application, which actions are allowed and which are denied, and any related advice and attributes.

Extended metadata

Federation configuration information specific to AM.

Extensible Access Control Markup Language (XACML)

Standard, XML-based access control policy language, including a processing model for making authorization decisions based on policies.

Federation

Standardized means for aggregating identities, sharing authentication and authorization data information between trusted providers, and

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

When a Subject successfully authenticates, AM associates the Subject with the Principal.

Privilege

In the context of delegated administration, a set of administrative tasks that can be performed by specified identities in a given realm.

Provider federation

Agreement among providers to participate in a circle of trust.

Realm

AM unit for organizing configuration and identity information.

Realms can be used for example when different parts of an organization have different applications and identity stores, and when different organizations use the same AM deployment.

Administrators can delegate realm administration. The administrator assigns administrative privileges to users, allowing them to perform administrative tasks within the realm.

Resource

Something a user can access over the network such as a web page.

Defined as part of policies, these can include wildcards in order to match multiple actual resources.

Resource owner

In OAuth 2.0, entity who can authorize access to protected web resources, such as an end user.

Resource server

In OAuth 2.0, server hosting protected web resources, capable of handling access tokens to respond to requests for such resources.

Response attributes

Defined as part of policies, these allow AM to return additional information in the form of "attributes" with the response to a policy decision.

Role based access control (RBAC)

Access control that is based on whether a user has been granted a set of permissions (a role).

Security Assertion Markup Language (SAML)

Standard, XML-based language for exchanging authentication and authorization data between identity providers and service providers.

Service provider (SP)

Entity that consumes assertions about a principal (and provides a service that the principal is trying to access).

Authentication Session

The interval while the user or entity is authenticating to AM.

Session

The interval that starts after the user has authenticated and ends when the user logs out, or when their session is terminated. For browser-based clients, AM manages user sessions across one or more applications by setting a session cookie. See also CTS-based sessions and Client-based sessions.

Session high availability

Capability that lets any AM server in a clustered deployment access shared, persistent information about users' sessions from the CTS token store. The user does not need to log in again unless the entire deployment goes down.

Session token

Unique identifier issued by AM after successful authentication. For a CTS-based sessions, the session token is used to track a principal's session.

Single log out (SLO)

Capability allowing a principal to end a session once, thereby ending her session across multiple applications.

Single sign-on (SSO)

Capability allowing a principal to authenticate once and gain access to multiple applications without authenticating again.

Site

Group of AM servers configured the same way, accessed through a load balancer layer. The load balancer handles failover to provide service-level availability.

The load balancer can also be used to protect AM services.

Standard metadata

Standard federation configuration information that you can share with other access management software.

Stateless Service

Stateless services do not store any data locally to the service. When the service requires data to perform any action, it requests it from a data store. For example, a stateless authentication service stores session state for logged-in users in a database. This way, any server in the deployment can recover the session from the database and service requests for any user.

All AM services are stateless unless otherwise specified. See also Client-based sessions and CTS-based sessions.

Subject

Entity that requests access to a resource

When an identity successfully authenticates, AM associates the identity with the Principal that distinguishes it from other identities. An identity can be associated with multiple principals.

Identity store

Data storage service holding principals' profiles; underlying storage can be an LDAP directory service or a custom `IdRepo` implementation.

Web Agent

Native library installed in a web server that acts as a policy enforcement point with policies based on web page URLs.