Evaluation Guide

/ ForgeRock Access Management 7.0.2

Latest update: 7.0.2

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

Abstract

How to quickly install ForgeRock® Access Management (AM) for new users evaluating the product. ForgeRock Access Management provides intelligent authentication, authorization, federation, and single sign-on functionality.

Table of Contents

Overview

The Evaluation Guide covers the tasks you need to quickly get a test or demo AM instance running.

Evaluate Access Management

 About Access Management Learn about AM, and how it provides an infrastructure for managing users, roles, and access to resources.	 "*Step 1. Prepare Your Server*" Get your machine ready for hosting an AM instance.	 "*Step 2. Deploy AM*" Deploy the AM WAR file into Tomcat.
 "*Step 3. Configure AM*" Log in to your AM instance as an administrator, and create an authentication tree.	 "*Step 4. Authenticate to AM*" Log in to your AM instance for the first time as a user, using your authentication tree.	 Where To Go From Here Discover what else AM has to offer for a variety of different deployments.

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

About Access Management

AM provides a service called *access management*, which manages access to resources, such as a web page, an application, or a web service, that are 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.

Authentication trees provide fine-grained authentication by allowing multiple paths and decision points throughout the authentication flow. They are made up of authentication nodes, which define actions taken during authentication. Authentication nodes are more granular than modules, with each node performing a single task, such as collecting a username or making a simple decision. Authentication nodes can have multiple outcomes, rather than just success or failure. AM lets you create complex yet customer-friendly authentication experiences by linking nodes together, creating loops, and nesting nodes within a tree.

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.

Keep on reading to try AM's access management capabilities by installing AM and configuring an authentication tree.

Chapter 2

Step 1. Prepare Your Server

To install AM in a demo or test environment, you need to perform the following prerequisite tasks:

Check Disk Space

AM's distribution `.war` file includes an embedded DS server, which stores AM's configuration data and serves as an identity store.

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.

Prepare a Fully Qualified Domain Name (FQDN)

AM requires that you use fully qualified domain names. 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.

For information on preparing an FQDN, see "To Prepare a Fully Qualified Domain Name".

Install a Supported Java Development Kit (JDK)

AM is a Java web application, so you need to download and install a supported JDK. For a list of JDK versions that AM supports, refer to Java.

For information on installing a JDK, see "To Install a JDK and Apache Tomcat".

Important

Ensure that the JDK's default truststore, for example, `$JAVA_HOME/lib/security/cacerts`, has, at least, `644` permissions:

```
$ sudo chmod 644 $JAVA_HOME/lib/security/cacerts
```

+ *Why Is This Required?*

When evaluating AM, the installation process deploys an embedded DS instance that AM uses as configuration store, user store, and CTS store. To connect to the DS instance using LDAPS, AM requires access to the self-signed certificate that DS generates.

If you are installing AM for evaluation purposes, AM creates a copy of your JDK's default `lib/security/cacerts` truststore, names it `truststore`, and places it in `/path/to/openam/security/keystores/`.

AM then attempts to add the DS self-signed certificate to that store, with an alias of `ds-ca-cert`.

Important

If the `lib/security/cacerts` truststore does not have the default password of `changeit`, and/or if it does not have at least `644` permissions, then AM installation will fail, as it will not be able to open the truststore to add the DS certificate.

You can change the permissions back as they were originally after installing AM.

Install a Supported Web Container

Although AM can run in a number of application servers, download Apache Tomcat for the purposes of this guide.

For a list of versions that AM supports, see Application containers.

For information on installing Apache Tomcat, see "To Install a JDK and Apache Tomcat".

Download ForgeRock Access Management

The *ForgeRock BackStage download site* hosts downloadable versions of AM.

For a list of supported operating systems, refer to Operating systems.

Note

The procedures to set up the software are written for use on a UNIX-like system. If you are running Microsoft Windows, adapt these examples accordingly.

To Prepare a Fully Qualified Domain Name

Before deploying and installing AM, give your system a DNS alias, such as `openam.example.com`. You can add a DNS alias by editing your hosts file.

Tip

If you already have a DNS server set up, or use a service such as localtest.me, you can use those instead of editing 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
```

Proceed to install a JDK and Apache Tomcat.

To Install a JDK and 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 Apache Tomcat.

1. Extract the JDK download file:

   ```
   $ mkdir -p /path/to/JDK
   $ unzip ~/Downloads/openjdk-X_bin.zip -d /path/to/JDK
   ```

2. Extract the Apache Tomcat download file:

   ```
   $ mkdir -p /path/to/tomcat
   $ unzip ~/Downloads/apache-tomcat-X.X.XX.zip -d /path/to/tomcat
   ```

3. (UNIX-like systems only) Make the scripts in Apache Tomcat's `bin/` directory executable:

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

4. Create an Apache 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"
   ```

5. (Optional) If you have a custom installation that differs from the documented Apache Tomcat installation, make sure to set Apache 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.

6. Make sure that your system's firewall does not block the port that Apache Tomcat uses (`8080` by default).

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

7. Start Apache Tomcat:

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

   It might take Apache Tomcat several seconds to start. When it 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
   ```

8. Navigate to Apache Tomcat's home page; for example, http://openam.example.com:8080.

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

   Proceed to "*Step 2. Deploy AM*".

Chapter 3

Step 2. Deploy AM

Deploying AM creates a default configuration that you can access with AM's administrative user; `amAdmin`.

To Deploy ForgeRock Access Management

Deploy AM into Apache Tomcat and then configure it for use.

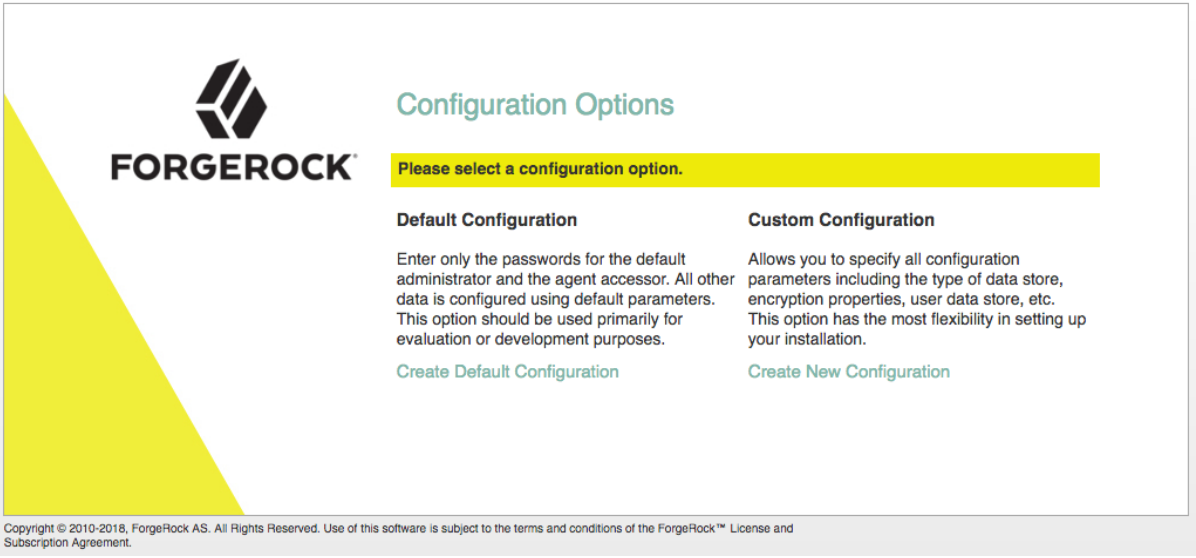
1. Copy the AM `.war` file to deploy in Apache Tomcat as `openam.war`:

   ```
   $ cp AM-7.0.2.war /path/to/tomcat/webapps/openam.war
   ```

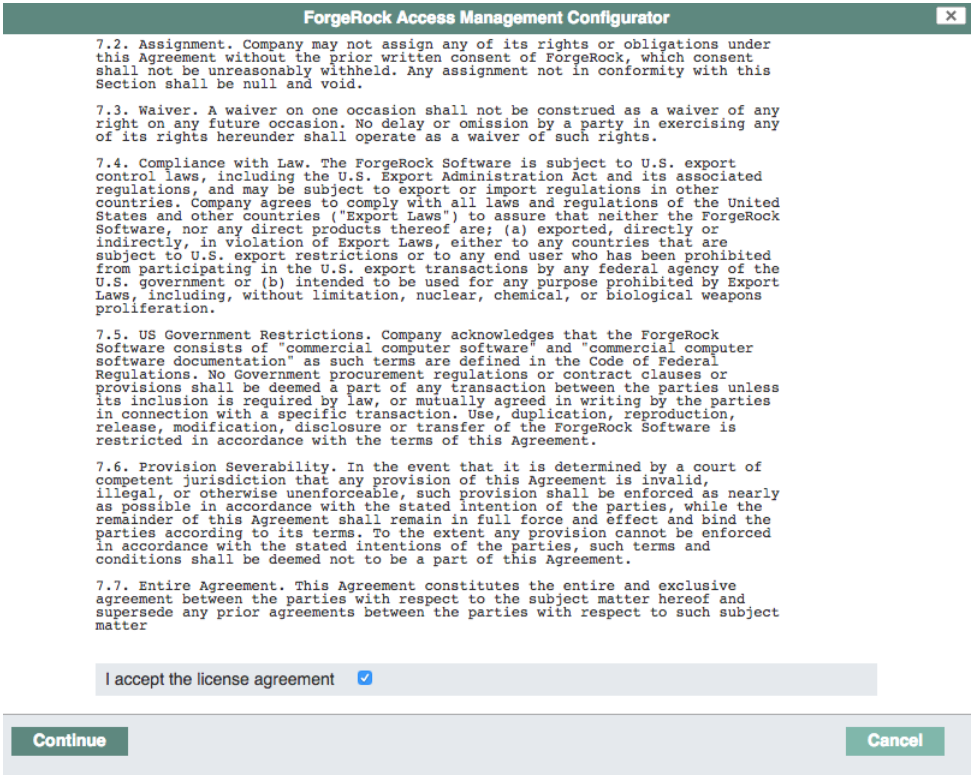
   It can take Apache Tomcat several seconds to deploy AM.

2. Navigate to the deployed AM application; for example, `http://openam.example.com:8080/openam/`.

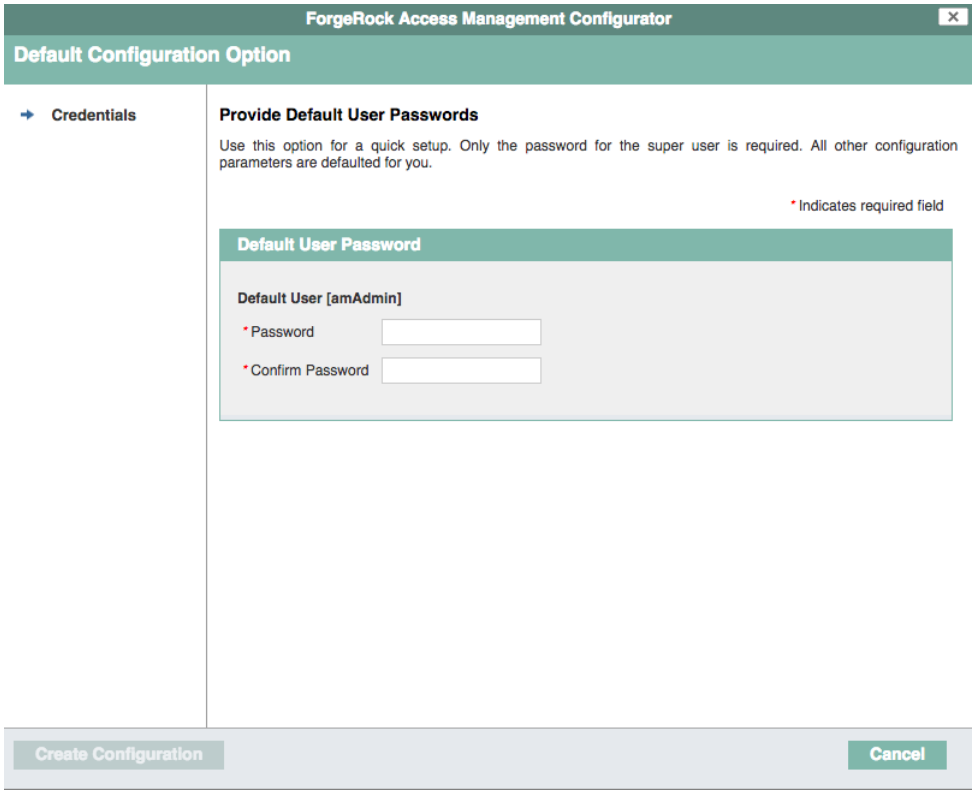
3. On the AM configuration screen, click Create Default Configuration.



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



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

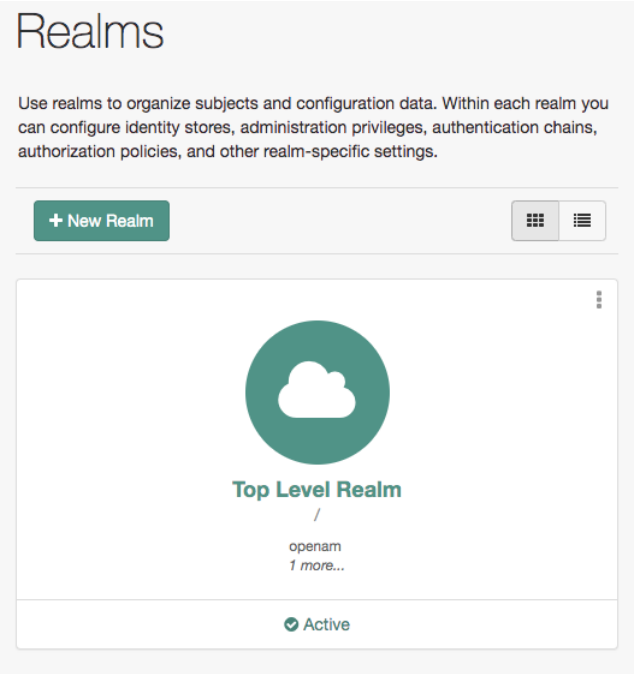


Note

When configuring AM for real-world use, do not use this password, it is only to get started with AM. The `amAdmin` user is the default AM administrator, that has full control over the AM configuration.

6. Click the Proceed to Login link, then log in as `amAdmin` with the password you configured in a previous step, `changeit`.

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



AM stores its configuration, including the embedded DS server, in a directory named after the deployment URI. In other words, if AM is deployed under `/openam`, then the configuration is saved under `$HOME/openam/.`

Tip

If you need to delete your configuration, the quickest way to start over is to stop Apache Tomcat, delete the AM configuration directory, and restart the AM web application to start the process from the beginning.

AM is now configured, and ready for use. Make sure you have successfully logged in to the AM console before configuring the authentication tree.

7. Make sure you have successfully logged in to the AM console, and then proceed to "*Step 3. Configure AM*".

Chapter 4

Step 3. Configure AM

Authentication trees provide fine-grained authentication by allowing multiple paths and decision points throughout the authentication flow.

Authentication trees are made up of authentication nodes, which define actions taken during authentication. Authentication nodes are granular, with each node performing a single task, such as collecting a username or making a simple decision. Authentication nodes can have multiple outcomes rather than just success or failure.

Tip

AM provides a number of ready-made sample authentication trees to demonstrate how they can be put together. For more information on setting up authentication trees, see "Configuring Authentication Trees" in the *Authentication and Single Sign-On Guide*.

To Configure an Authentication Tree

Follow these steps to create an authentication tree that you can use to log in to AM.

1. On the Realms page of the AM console, select the realm in which to create the authentication tree.
2. On the Realm Overview page, select Authentication in the menu on the left, and then select Trees.
3. On the Trees page, select Create Tree. Enter a tree name; for example, `myAuthTree`, and then select Create.

   The authentication tree designer is displayed, with the Start entry point connected to the Failure exit point.

   The authentication tree designer provides the following features on the toolbar:

Authentication Tree Designer Toolbar

Button	Usage
	Lay out and align nodes according to the order they are connected.
	Toggle the designer window between normal and full-screen layout.
	Remove the selected node. Note that the Start entry point cannot be deleted.

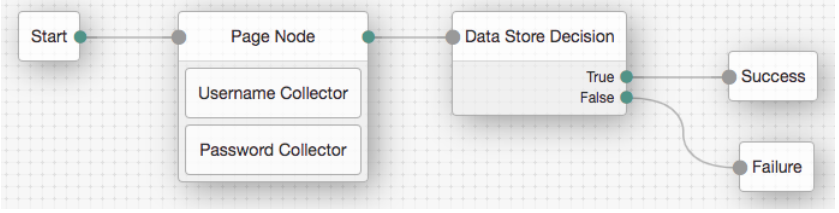
4. Drag the following nodes from the Components panel on the left-hand side and drop them into the designer area:

   - Page Node
   - Username Collector
   - Password Collector
   - Data Store Decision
   - Success

   The Data Store Decision authentication node uses the credentials to authenticate the user against the identity stores configured for the realm. In this example, the username and password are obtained by a combination of the Username Collector and Password Collector nodes.

5. Drag and drop the Username Collector and Password Collector onto the Page Node, so that they will both appear on the same page when logging in.

6. Connect the nodes as follows:



Tip

You can configure the node properties by using the panel on the right side of the page. For more information on the available properties for each node, see "Authentication Nodes Configuration Reference" in the *Authentication and Single Sign-On Guide*.

For more information on setting up more complex authentication trees, see "Configuring Authentication Trees" in the *Authentication and Single Sign-On Guide*.

7. You are now ready to authenticate your first user!

   Proceed to "*Step 4. Authenticate to AM*".

Chapter 5

Step 4. Authenticate to AM

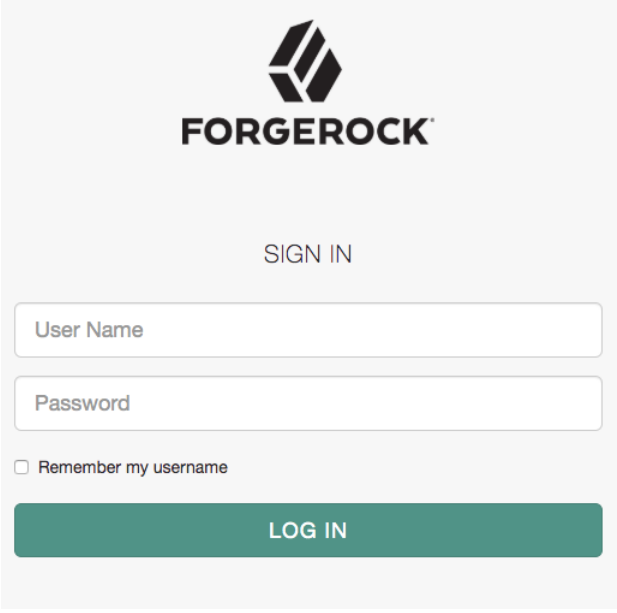
Now that you have completed "*Step 3. Configure AM*", you can use the `myAuthTree` tree you created to authenticate a user.

To test your authentication tree in a web browser, navigate to a URL similar to the following:

```
http://openam.example.com:8080/openam/XUI/?realm=/&service=myAuthTree#login
```

Use the correct FQDN, port number, and deployment path for your environment. Also ensure you use the correct authentication tree name, in the example above, the tree is named `myAuthTree`.

Log in as the built-in `demo` user, with the password `Ch4ng31t`.



On successful login, AM creates a cookie named `iPlanetDirectoryPro` in your browser for your domain; for example, `example.com`. That cookie is then available to all servers in the `example.com` domain, such as `openam.example.com`.

If you examine this cookie, you see that it has a value such as `AQI5wM2L...*AAJTS...`. 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).

Congratulations on authenticating your first user with AM!

See what else can AM do for you by reading "*Next Steps*".

Chapter 6

Next Steps

AM can do much more than authenticate users. 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. Find out more about them:

"User Self-Service Features"

Discover how end users can create their own accounts, reset their passwords, and others.

"Single Sign-On"

Create seamless end user journeys using AM, Web Agents and Java Agents, or IDM.

"SAML v2.0 Federation"

Create seamless end user journeys by federating identities using the SAML v2.0 standard.

"OAuth 2.0 and OAuth 2.0-based Standards Federation"

Protect applications and authorize clients to perform tasks on behalf of users with AM's OAuth 2.0, OpenID Connect 1.0, and User-Managed Access 2.0 standards.

"Policy Enforcement Points and Access Policies"

Control who access your applications by writing policies that AM will evaluate to grant or deny access to your resources. AM's policy enforcement points or IG will enforce AM decisions and protect your applications or endpoints.

"Modern APIs For Developers"

Discover the REST, RESTful, and Java APIs that AM exposes.

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 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 "*Dashboards*" in the *Setup 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 CDSSO" in the *Authentication and Single Sign-On Guide*.

SAML v2.0 Federation

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 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 OAuth 2.0-based Standards Federation

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. User-Managed Access (UMA) 2.0 takes OpenID Connect a step further, and lets the end user manage access to their resources.

AM can serve as the authorization server for your clients, or as a client to another authorization server. As an authorization server, AM supports capabilities such as:

- Dynamic client registration
- Using macaroons as access and refresh tokens
- Client-based access and refresh tokens
- Proof-of-possession
- Scripted OpenID Connect claims
- Authentication requirements for ID tokens.

For more information, see:

- OAuth 2.0 Guide
- OpenID Connect 1.0 Guide
- User-Managed Access (UMA) 2.0 Guide

Policy Enforcement Points and Access Policies

AM can handle large numbers of access policies, each of which gives you control over user provisioning and user entitlements. For details, see 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*.

AM also includes Web Agents and Java Agents, which are add-on components that operate as a policy enforcement point (PEP) for a website or application. For example, you can install a web agent to enforce AM's authorization decisions on Apache HTTP Server.

For details, see the *ForgeRock Web Agents User Guide* and the *ForgeRock Java Agents User Guide*.

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 Getting Started with REST.

- 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 Trees in the *Authentication and Single Sign-On Guide*
- Policy Condition Script API Functionality in the *Authorization Guide*
- Customizing Identity Stores in the *Setup Guide*
- Customizing OAuth 2.0 Scope Handling in the *OAuth 2.0 Guide*

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.