FORGEROCK®

User Guide

ForgeRock Service Broker 1

Copyright © 2016 ForgeRock AS.

Abstract

Guide for using the ForgeRock Service Broker to manage and federate access to web applications and web-based resources.

Table of Contents

Preface

This guide covers concepts, configuration, and usage procedures for working with the ForgeRock Service Broker.

This guide is written for anyone using the ForgeRock Service Broker to manage and federate access to web applications and web-based resources.

Chapter 1
Introducing the ForgeRock Service Broker

The ForgeRock Service Broker allows Cloud Foundry applications to access OAuth 2.0 features provided by the ForgeRock Identity Platform.

Cloud Foundry is an open platform as a service (PaaS), providing a choice of clouds, developer frameworks, and application services.

Integrate with the ForgeRock Identity Platform to::

• Obtain OAuth 2.0 access tokens using the client credentials grant type.

• Validate OAuth 2.0 access tokens or OpenID Connect ID tokens passed your application.

The ForgeRock Service Broker automates the process of creating OAuth 2.0 client profiles, and requires a set of credentials with privileges for adding and removing OAuth 2.0 clients. The recommended approach is to create a new user in OpenAM, add the user to a new group, and give that group the `AgentAdmin` privilege, allowing members to create and remove OAuth 2.0 clients. For information on creating a user and delegating privileges, see Section 2.1, "Preparing OpenAM".

Chapter 2

Implementing the ForgeRock Service Broker

This chapter explains how to prepare OpenAM for use with the ForgeRock Service Broker, and how to install the broker into both Pivotal Cloud Foundry (PCF), and Cloud Foundry environments.

2.1. Preparing OpenAM

This section describes how to prepare an OpenAM installation for use with the ForgeRock Service Broker.

Creating OAuth 2.0 agent profiles in OpenAM requires a set of OpenAM credentials with the correct privileges.

The procedure below creates a new user in OpenAM, adds the user to a new group, and gives that group the `AgentAdmin` privilege, allowing members to create and remove OAuth 2.0 agent profiles.
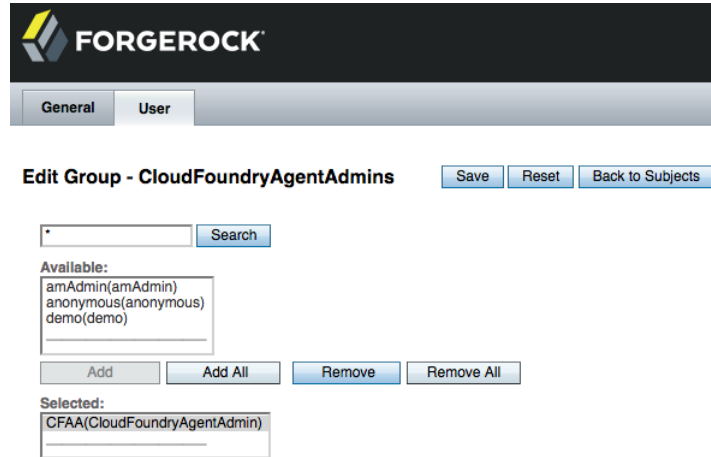
Procedure 2.1. To Prepare OpenAM for ForgeRock Service Broker Installation

1. On the Realms page, click the realm where the ForgeRock Service Broker will create OAuth 2.0 agent profiles, and then click Subjects.

2. On the Subjects page:

   a. Click the User tab, and then click New.

   b. Enter values for all required fields, and then click OK.

      In this example, specify the value `CloudFoundryAgentAdmin` as the ID.

      You will need the user's ID and password to configure the service broker.

   c. Click the Group tab, and then click New.

   d. Enter an ID for the new group, for example `CloudFoundryAgentAdmins`, and then click OK.

   e. Click the name of the group you just created, and then click the User tab.

   f. In the list of available users, double-click the `CloudFoundryAgentAdmin` user you created earlier to add it to the Selected list, and then click Save.

3. Click Back to Subjects, click the Privileges tab, and then click the `CloudFoundryAgentAdmins` group you created earlier.

4. In the list of privileges, enable the Read and write access to all configured Agents privilege, and then click Save.

   Users in the selected group are now able to create and configure OAuth 2.0 agent profiles.

2.2. Installing into Pivotal Cloud Foundry

This section describes how to install the ForgeRock Service Broker into Pivotal Cloud Foundry (PCF).

If you are using Pivotal Cloud Foundry, you can use Ops Manager to add OpenAM integration by installing the ForgeRock Service Broker for PCF tile.

Installation requires that you import the tile into your PCF installation, and then configure the ForgeRock Service Broker for PCF by providing the URL of an OpenAM instance and the credentials of the user you created for the broker. You will also need to provide the realm that you want the OAuth 2.0 clients to be created under, and the OAuth 2.0 scopes that the OAuth 2.0 clients may request.

Procedure 2.2. To Install into Pivotal Cloud Foundry

1. In a web browser, navigate to the Pivotal Network, click ForgeRock Identity Platform for PCF, and then download the latest version.

2. Log in to PCF Ops Manager as an administrative user, and then click Import a Product.

3. Browse to the downloaded `forgerock-broker-1.0.10.pivotal` file, and then click Open.

The ForgeRock Identity Platform for PCF item is uploaded and appears in the left-hand menu in Pivotal Ops Manager.

4. Click the Plus icon to add ForgeRock Identity Platform for PCF to the Installation Dashboard.
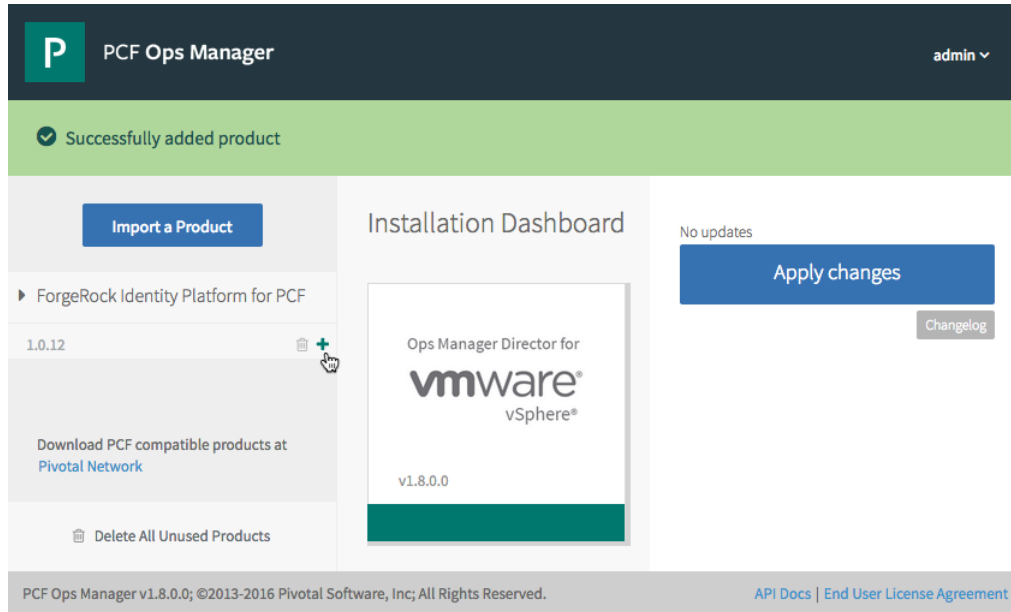
5. On the Installation Dashboard, click ForgeRock Identity Platform for PCF, and then click the OpenAM tab.

6. On the OpenAM Configuration page, complete the form as follows:

Location

The URI to the OpenAM instance, for example `http://openam.example.com:8080/openam/`.

Username

Specify the username you created that the broker will use to authenticate with OpenAM. See Procedure 2.1, "To Prepare OpenAM for ForgeRock Service Broker Installation".

Password

Specify the password you created that the broker will use to authenticate with OpenAM. See Procedure 2.1, "To Prepare OpenAM for ForgeRock Service Broker Installation".
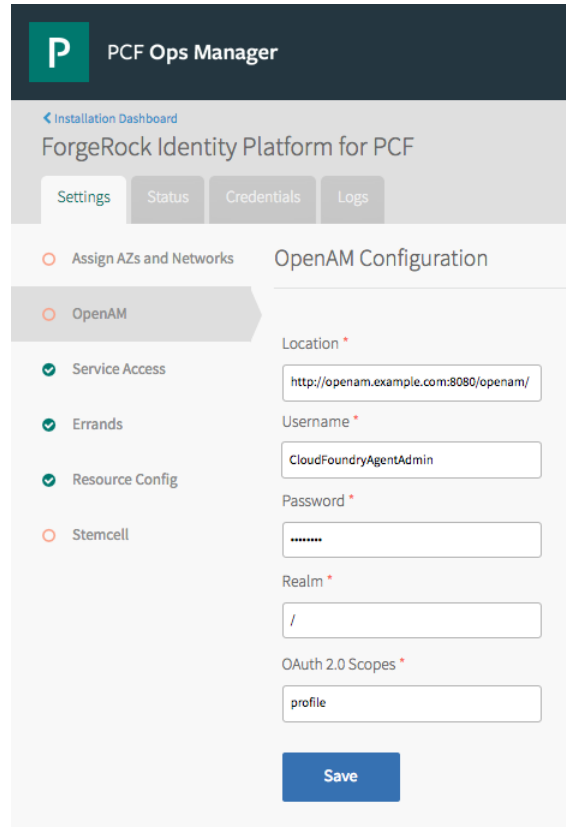
Realm

The realm to use to authenticate and create the OAuth 2.0 clients.

OAuth 2.0 Scopes

The scopes that applications will be allowed to request when creating access tokens.

See the following image for an example:

7. Make any other Pivotal-specific configuration changes as necessary, and then save your work.

8. Return to the Installation Dashboard, and then click Apply changes to configure the ForgeRock Identity Platform for PCF tile.

Once the tile is correctly configured and all changes have been applied, Cloud Foundry application developers will be able to see the OpenAM OAuth 2.0 service in the marketplace.

For details on how to make use of the OpenAM OAuth 2.0 broker service, see Chapter 3, "*Using the ForgeRock Service Broker*".

2.3. Installing into Cloud Foundry

This section describes how to install the ForgeRock Service Broker into Cloud Foundry.

Deploying the service broker into Cloud Foundry installations without Ops Manager requires use of the Cloud Foundry **cf** command-line tool.

Procedure 2.3. To Install into Cloud Foundry

1. Download the latest version of the OpenAM Cloud Foundry Broker WAR file, `service-broker-servlet-1.0.10.war`, from the ForgeRock Maven repository.

2. Push the OpenAM Cloud Foundry Broker WAR file to Cloud Foundry as a new application by using the **cf push** command:

   ```
   $ cf push forgerockbroker-1.0.10 -p service-broker-servlet-1.0.10.war
   ```

3. Set environment variables for the broker by using multiple **cf set-env** commands:

   ```
   $ cf set-env forgerockbroker-1.0.10 {variable} {value}
   ```

   The list of supported variables is as follows:

   OPENAM_BASE_URI

   The URI to the OpenAM instance, for example `http://openam.example.com:8080/openam/`.

   OPENAM_USERNAME

   Specify the username you created that the broker will use to authenticate with OpenAM. See Procedure 2.1, "To Prepare OpenAM for ForgeRock Service Broker Installation".

   OPENAM_PASSWORD

   Specify the password you created that the broker will use to authenticate with OpenAM. See Procedure 2.1, "To Prepare OpenAM for ForgeRock Service Broker Installation".

   SECURITY_USER_NAME

   Username that will be used by Cloud Foundry when accessing the broker. You should securely generate a random value.

   SECURITY_USER_PASSWORD

   Password that will be used by Cloud Foundry when accessing the broker. You should securely generate a random value.

   OPENAM_REALM

   Optional realm to use to authenticate and create the OAuth 2.0 clients. The top-level realm is used by default if not specified.

OAUTH2_SCOPES

> The scopes that applications will be allowed to request when creating access tokens.

Example:

```
$ cf set-env forgerockbroker-1.0.10 OPENAM_BASE_URI http://openam.example.com:8080/openam/
$ cf set-env forgerockbroker-1.0.10 OPENAM_USERNAME CloudFoundryAgentAdmin
$ cf set-env forgerockbroker-1.0.10 OPENAM_PASSWORD changeit
$ cf set-env forgerockbroker-1.0.10 SECURITY_USER_NAME Qvd9a7ky
$ cf set-env forgerockbroker-1.0.10 SECURITY_USER_PASSWORD yNUzCr3C
$ cf set-env forgerockbroker-1.0.10 OAUTH2_SCOPES profile
```

4. Restage the application to apply changes to the environment variables by using the **cf restage** command:

```
$ cf restage forgerockbroker-1.0.10
```

> **Tip**
>
> You can confirm the applied environment variables by using the **cf env** command. For example:
>
> ```
> $ cf env forgerockbroker-1.0.10
>
>  User-Provided:
>  OAUTH2_SCOPES: profile
>  OPENAM_BASE_URI: http://openam.example.com:8080/openam/
>  OPENAM_PASSWORD: changeit
>  OPENAM_USERNAME: CloudFoundryAgentAdmin
>  SECURITY_USER_NAME: Qvd9a7ky
>  SECURITY_USER_PASSWORD: yNUzCr3C
> ```

5. Find the URN for the application by using the **cf app** command:

```
$ cf app forgerockbroker-1.0.10

 Showing health and status for app forgerockbroker-1.0.10 in org forgerock / space development as
 admin...
 OK

 requested state: started
 instances: 1/1
 usage: 1G x 1 instances
 urls: forgerockbroker-101.cfapps-0123.pivotal.io
 last uploaded: Wed Sep 28 10:35:02 UTC 2016
```

In the above example, the application URN is `forgerockbroker-101.cfapps-0123.pivotal.io`.

6. Create the service broker by using the **cf create-service-broker** command:

```
$ cf create-service-broker forgerockbroker {Security User Name} {Security User Password} {Application
URL}
```

The values used for *{Security User Name}* and *{Security User Password}* must be the same as those used for the `SECURITY_USER_NAME` and `SECURITY_USER_PASSWORD` environment variables created earlier.

Combine the protocol used to access the broker, for example `http://`, with the application URN retrieved in the previous step to determine the value for the *{Application URL}* parameter.

Example:

```
$ cf create-service-broker forgerockbroker Qvd9a7ky yNUzCr3C http://forgerockbroker-101.cfapps-0123
.pivotal.io
```

7. After the service broker has been created, grant access to its service plans by using the **cf enable-service-access** command:

```
$ cf enable-service-access openam-oauth2
```

8. Verify that the `openam-oauth2` service is now visible in the marketplace by using the **cf marketplace** command:

```
$ cf marketplace

Getting services from marketplace in org forgerock / space documentation as admin...
OK

service          plans           description
hello            hi              Friendly service that greets you
openam-oauth2    shared          Uses ForgeRock OpenAM to provide OAuth 2.0 authorization

TIP:  Use 'cf marketplace -s SERVICE' to view descriptions of individual plans of a given service.
```

For details on how to bind to the OpenAM OAuth 2.0 service, see Chapter 3, "*Using the ForgeRock Service Broker*".

Chapter 3
Using the ForgeRock Service Broker

This chapter describes how to bind a Cloud Foundry application to the ForgeRock Service Broker so that it can use OpenAM for OAuth 2.0 functionality.

Binding a Cloud Foundry application to the ForgeRock Service Broker creates an OAuth 2.0 agent profile in OpenAM for the application. The application is provided with the agent profile credentials, which can be used in subsequent calls to OpenAM.

3.1. Binding an Application to the ForgeRock Service Broker

This section explains how to bind a Cloud Foundry application to the ForgeRock Service Broker, so that it can use OpenAM for OAuth 2.0 functionality.

Procedure 3.1. To Bind an Application to the ForgeRock Service Broker

1. Verify that the OpenAM OAuth 2.0 service is available in the catalog:

    ```
    $ cf marketplace

     Getting services from marketplace in org forgerock / space documentation as admin...
     OK

     service         plans           description
     hello           hi              Friendly service that greets you
     openam-oauth2   shared          Uses ForgeRock OpenAM to provide OAuth 2.0 authorization

     TIP:  Use 'cf marketplace -s SERVICE' to view descriptions of individual plans of a given service.
    ```

    If the `openam-oauth2` service is not shown, you must install the ForgeRock Service Broker. See Chapter 2, "*Implementing the ForgeRock Service Broker*".

2. If the `openam-oauth2` service is available, create a shared service instance that the Cloud Foundry application can bind to by using the **cf create-service** command:

    ```
    $ cf create-service openam-oauth2 shared myOpenAMService
    ```

3. Bind the Cloud Foundry application to the new service by using the **cf bind-service** command:

    ```
    $ cf bind-service myCloudFoundryApp myOpenAMService
    ```

4. After the bind is complete, restage the application to see the changes.

```
$ cf restage myCloudFoundryApp
```

The OAuth 2.0 agent profile credentials created by OpenAM are available to the application in the
`VCAP_SERVICES` environment variable. For example:

```
VCAP_SERVICES=
{
    "openam-oauth2": [
        {
            "name": "myOpenAMService",
            "label": "openam-oauth2",
            "plan": "shared",
            "credentials": {
                "uri": "http://openam.example.com:8080/openam/oauth2/",
                "username": "c031e0f7-18fa",
                "password": "ce8e565d-a4a1"
            }
        }
    ]
}
```

3.2. Obtaining OAuth 2.0 Access Tokens

A Cloud Foundry application that is bound to an instance of the ForgeRock Service Broker can make
calls to OpenAM to acquire an OAuth 2.0 access token, for use with other services.

To obtain an access token, send a request to the OpenAM OAuth 2.0 access token endpoint, as
demonstrated in the procedure below:

1. Check the credentials for accessing the OpenAM agent profile by reading the `VCAP_SERVICES`
   environment variable. For example:

```
VCAP_SERVICES=
{
    "openam-oauth2": [
        {
            "name": "myOpenAMService",
            "label": "openam-oauth2",
            "plan": "shared",
            "credentials": {
                "uri": "http://openam.example.com:8080/openam/oauth2/",
                "username": "c031e0f7-18fa",
                "password": "ce8e565d-a4a1"
            }
        }
    ]
}
```

2. From the values provided in the `VCAP_SERVICES` environment variable, create a string for the basic
   `authorization` header as follows:

   a. Concatenate the value of the `username` attribute, a colon character (`:`), and the value of the
      `password` attribute.

For example:

```
c031e0f7-18fa:ce8e565d-a4a1
```

b. Base64-encode the result.

For example:

```
YzAzMWUwZjctMThmYTpjZThlNTY1ZC1hNGEx
```

3. From the values provided in the `VCAP_SERVICES` environment variable, create the URL for the access token endpoint by appending `access_token` to the value of the `uri` attribute.

For example:

```
http://openam.example.com:8080/openam/oauth2/access_token
```

4. Using the values from the previous steps, perform an HTTP POST call to obtain the access token, for example:

```
$ curl \
--request POST
 \
--header "Authorization: Basic YzAzMWUwZjctMThmYTpjZThlNTY1ZC1hNGEx"
 \
--header "Content-Type: application/x-www-form-urlencoded"
 \
--data "grant_type=client_credentials"
 \
--data "scope=profile" \
"http://openam.example.com:8080/openam/oauth2/access_token"

 {
     "access_token": "78069524-f679-48ff-bb59-47ab574846ea",
     "scope": "profile",
     "token_type": "Bearer",
     "expires_in": 3599
 }
```

For information on validating the provided access token, see Section 3.3, "Validating OAuth 2.0 Access Tokens".

For information on the OAuth 2.0 functionality provided by OpenAM, see the *OpenAM OAuth 2.0 Guide*.

3.3. Validating OAuth 2.0 Access Tokens

If the Cloud Foundry application receives OAuth 2.0 access tokens, you can use OpenAM to validate those tokens.

To validate an access token, send a request to the OpenAM OAuth 2.0 access token introspection endpoint, as demonstrated in the procedure below:

1. Check the credentials for accessing the OpenAM agent profile by reading the `VCAP_SERVICES` environment variable. For example:

```
VCAP_SERVICES=
{
    "openam-oauth2": [
        {
            "name": "myOpenAMService",
            "label": "openam-oauth2",
            "plan": "shared",
            "credentials": {
                "uri": "http://openam.example.com:8080/openam/oauth2/",
                "username": "c031e0f7-18fa",
                "password": "ce8e565d-a4a1"
            }
        }
    ]
}
```

2. From the values provided in the `VCAP_SERVICES` environment variable, create a string for the basic `authorization` header as follows:

    a. Concatenate the value of the `username` attribute, a colon character (`:`), and the value of the `password` attribute.

    For example:

    ```
    c031e0f7-18fa:ce8e565d-a4a1
    ```

    b. Base64-encode the result.

    For example:

    ```
    YzAzMWUwZjctMThmYTpjZThlNTY1ZC1hNGEx
    ```

3. From the values provided in the `VCAP_SERVICES` environment variable, create the URL for the token introspection endpoint by appending `access_token` to the value of the `uri` attribute.

    For example:

    ```
    http://openam.example.com:8080/openam/oauth2/introspect
    ```

4. Using the values from the previous, perform an REST POST call to validate an access token, for example:

```
$ curl \
--request POST
 \
--header "Authorization: Basic YzAzMWUwZjctMThmYTpjZThlNTY1ZC1hNGEx" \
"http://openam.example.com:8080/openam/oauth2/introspect?token=78069524-f679-48ff-bb59-47ab574846ea"

 {
     "active": true,
     "scope": "profile",
     "client_id": "c031e0f7-18fa",
     "user_id": "c031e0f7-18fa",
     "token_type": "access_token",
     "exp": 1475666232,
     "sub": "c031e0f7-18fa",
     "iss": "http://openam.example.com:8080/openam/oauth2"
 }
```

For information on the OAuth 2.0 functionality provided by OpenAM, see the *OpenAM OAuth 2.0 Guide*.

Appendix A. Getting Support

For more information or resources about OpenAM and ForgeRock Support, see the following sections:

A.1. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

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

- ForgeRock core 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.

Core documentation therefore follows a three-phase review process designed to eliminate errors:

- Product managers and software architects review project documentation design with respect to the readers' software lifecycle needs.

- Subject matter experts review proposed documentation changes for technical accuracy and completeness with respect to the corresponding software.

- Quality experts validate implemented documentation changes for technical accuracy, completeness in scope, and usability for the readership.

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

Fully reviewed, published core documentation is available at http://backstage.forgerock.com/. Use this documentation when working with a ForgeRock Identity Platform release.

A.2. Joining the ForgeRock Community

Visit the Community resource center where you can find information about each project, download trial builds, browse the resource catalog, ask and answer questions on the forums, find community events near you, and find the source code for open source software.

A.3. How to Report Problems or Provide Feedback

If you have found issues or reproducible bugs within ForgeRock Service Broker 1, report them in https://bugster.forgerock.org.

When requesting help with a problem, include the following information:

- Description of the problem, including when the problem occurs and its impact on your operation

- Description of the environment, including the following information:

  - Machine type

  - Operating system and version

  - Web server or container and version

  - Java version

  - OpenAM version

  - Any patches or other software that might be affecting the problem

- Steps to reproduce the problem

- Any relevant access and error logs, stack traces, or core dumps

A.4. Getting Support and Contacting ForgeRock

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

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

OpenAM 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 subjects 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 policy agent profile configuration information, typically created to delegate policy agent profile creation to the user installing a policy 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 OpenAM 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	OpenAM 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. OpenAM 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. OpenAM can play this role in the OAuth 2.0 authorization framework.
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 OpenAM configuration data.
Cross-domain single sign-on (CDSSO)	OpenAM capability allowing single sign-on across different DNS domains.
Delegation	Granting users administrative privileges with OpenAM.
Entitlement	Decision that defines which resource names can and cannot be accessed for a given subject 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 OpenAM.
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 OpenAM on the service provider side; OpenAM lets you create both .NET and Java Fedlets.
Hot swappable	Refers to configuration properties for which changes can take effect without restarting the container where OpenAM 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 EE policy agent	Java web application installed in a web container that acts as a policy agent, 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	Agent that intercepts requests for resources, directs principals to OpenAM for authentication, and enforces policy decisions from OpenAM.
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, OpenAM associates the Subject with the Principal.
Privilege	In the context of delegated administration, a set of administrative tasks that can be performed by specified subjects in a given realm.
Provider federation	Agreement among providers to participate in a circle of trust.
Realm	OpenAM unit for organizing configuration and identity information. Realms can be used for example when different parts of an organization have different applications and user data stores, and when different organizations use the same OpenAM 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 OpenAM 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).
Session	The interval that starts with the user authenticating through OpenAM and ends when the user logs out, or when their session is terminated. For browser-based clients, OpenAM manages user sessions across one or more applications by setting a session cookie. See also Stateful session and Stateless session.
Session failover (SFO)	Capability to allow another OpenAM server to manage a session when the OpenAM server that initially authenticated the principal goes offline.
Session token	Unique identifier issued by OpenAM after successful authentication. For a Stateful session, 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 OpenAM servers configured the same way, accessed through a load balancer layer.
	The load balancer handles failover to provide service-level availability. Use sticky load balancing based on `amlbcookie` values to minimize cross-talk in the site.
	The load balancer can also be used to protect OpenAM services.
Standard metadata	Standard federation configuration information that you can share with other access management software.
Stateful session	An OpenAM session that resides in the OpenAM server's memory and, if session failover is enabled, is also persisted in the Core Token

	Service's token store. OpenAM tracks stateful 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.
Stateless session	An OpenAM session for which state information is encoded in OpenAM and stored on the client. The information from the session is not retained in OpenAM's memory. For browser-based clients, OpenAM sets a cookie in the browser that contains the session information.
Subject	Entity that requests access to a resource
	When a subject successfully authenticates, OpenAM associates the subject with the Principal that distinguishes it from other subjects. A subject can be associated with multiple principals.
User data store	Data storage service holding principals' profiles; underlying storage can be an LDAP directory service, a relational database, or a custom `IdRepo` implementation.
Web policy agent	Native library installed in a web server that acts as a policy agent with policies based on web page URLs.