User Guide

ForgeRock Service Broker 2

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

Abstract

Guide for using the services provided by the ForgeRock® Service Broker.

Table of Contents

Preface

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

This guide is for anyone using the ForgeRock Service Broker to manage and federate access to Cloud Foundry applications, or to provide a route service to transform or process requests before they reach a Cloud Foundry application.

Chapter 1
Introducing the ForgeRock Service Broker

The ForgeRock Service Broker offers the following services to Cloud Foundry (CF) applications:

- **AM OAuth 2.0 Service**, to manage and federate access to web applications and web-based resources. CF applications can use this service to obtain OAuth 2.0 access tokens with the client credentials grant type, and validate OAuth 2.0 access tokens or OpenID Connect ID tokens passed to the application.

  The AM OAuth 2.0 Service 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 AM, 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 AM for ForgeRock Service Broker".

- **IG Route Service**, a fully brokered route service to filter traffic to and from CF applications, adapting requests to protect applications, and adapting responses to filter outgoing content.

  When an application is bound to an instance of the route service, requests to that application are directed to the route service before they are passed along to the application. The IG Route Service can be configured to require authentication or authorization before the request is passed to the application, or can throttle the number of requests that are allowed to access it in a given time. In fact, any of the features available in IG and described in the Gateway Guide can be configured in the route service and made available to CF applications.

  For more information about CF route services, see *Route Services* in the CF documentation.

Chapter 2

Preparing Services for the ForgeRock Service Broker

This chapter explains how to prepare installations of AM and IG for use with the ForgeRock Service Broker.

2.1. Preparing AM for ForgeRock Service Broker

This section describes how to prepare an AM installation for use with the ForgeRock Service Broker. Follow the procedure in this section if you plan to use the AM OAuth 2.0 Service.

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

The following procedure creates a new user in AM, 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 AM 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.

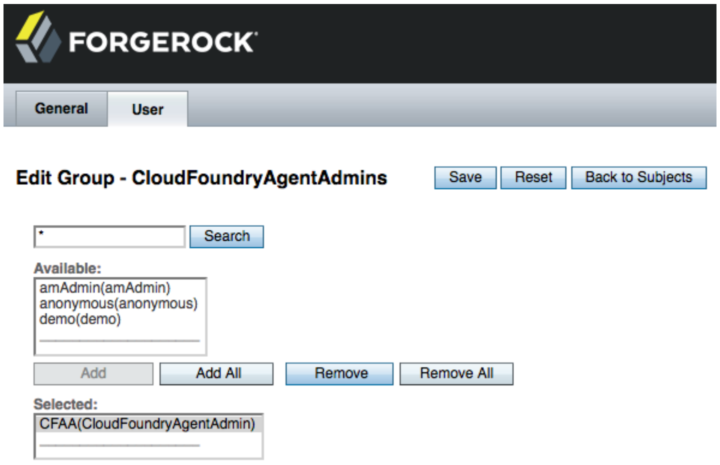
      You will need the user's ID and password to configure the service broker.

   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.



Edit Group - CloudFoundryAgentAdmins

Available:
amAdmin(amAdmin)
anonymous(anonymous)
demo(demo)

Selected:
CFAA(CloudFoundryAgentAdmin)

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. Preparing IG for ForgeRock Service Broker

This section describes how to prepare an IG installation to use with the ForgeRock Service Broker. Follow the procedures in this section if you plan to use the IG Route Service.

Consider the following points when preparing your installation:

- Install IG in a separate environment to CF, on an IP address that CF can access. For example, set up an instance of IG in Amazon Web Services, Google Cloud Platform, or Microsoft Azure.
- Configure IG for *HTTPS*. For information about using HTTPS with IG, see Configuring IG For HTTPS (Client-Side) the *IG Gateway Guide*.
- Use the `config.json` file described in Section 2.2.1, "Adding the Base Configuration of IG" as a template in your IG configuration. This base configuration file manages how requests are redirected from the IG Route Service back to the CF load balancer.

2.2.1. Adding the Base Configuration of IG

Add the following route to the IG configuration as `IG-base/config/config.json`:

```
{
  "heap": [
    {
      "name": "ClientHandler",
      "type": "ClientHandler",
      "config": {
        "hostnameVerifier": "ALLOW_ALL",
        "trustManager": {
          "type": "TrustAllManager"
        }
      }
    },
    {
      "name": "_router",
      "type": "Router",
      "config": {
        "defaultHandler": {
          "type": "StaticResponseHandler",
          "config": {
            "status": 404,
            "reason": "Not Found",
            "headers": {
              "Content-Type": [
                "application/json"
              ]
            },
            "entity": "{ \"error\": \"Something went wrong, contact the sys admin\"}"
          }
        }
      }
    },
    {
      "type": "Chain",
      "name": "CloudFoundryProxy",
      "config": {
        "filters": [
          {
            "type": "ScriptableFilter",
            "name": "CloudFoundryRequestRebaser",
            "comment": "Rebase the request based on the CloudFoundry provided headers",
            "config": {
              "type": "application/x-groovy",
              "source": [
                "Request newRequest = new Request(request);",
                "org.forgerock.util.Utils.closeSilently(request);",
                "newRequest.uri = URI.create(request.headers['X-CF-Forwarded-Url'].firstValue);",
                "newRequest.headers['Host'] = newRequest.uri.host;",
                "logger.info('Receive request : ' + request.uri + ' forwarding to ' + newRequest.uri);",
                "Context newRoutingContext =
 org.forgerock.http.routing.UriRouterContext.uriRouterContext(context).originalUri(newRequest.uri.asURI()).build();
                "return next.handle(newRoutingContext, newRequest);"
              ]
            }
          }
        ],
        "handler": "_router"
      },
      "capture": [
        "request",
        "response"
      ]
    }
  ],
  "handler": {
    "type": "DispatchHandler",
    "name": "Dispatcher",
    "config": {
      "bindings": [
        {
          "condition": "${not empty request.headers['X-CF-Forwarded-Url']}",
          "handler": "CloudFoundryProxy"
        },
        {
          "handler": {
            "type": "StaticResponseHandler",
            "config": {
              "status": 400,
              "entity": "Bad request : expecting a header X-CF-Forwarded-Url"
            }
          }
        }
      ]
    }
  }
}
```

Notice the following features of the route:

- For testing purposes, the ClientHandler is configured to accept all SSL certificates and to not verify host names. This configuration is not recommended for a production environment.
- If a request contains a `X-CF-Forwarded-Url` header, the DispatchHandler dispatches the request to the chain called `CloudFoundryProxy`.
- The ScriptableFilter in `CloudFoundryProxy` returns the request to the original URI for processing.

2.2.2. Adding Routes to Protect a CF Application

Configure the IG Route Service to protect applications by adding additional routes to the IG configuration, as described in the *IG Gateway Guide*.

Add the routes to the IG configuration in the `IG-base/config/routes` directory.

Consider adding routes for the following typical use cases:

- To require users to authenticate before requests are passed to the CF application, set up IG as an OpenID connect relying party, as described in About IG As an OpenID Connect 1.0 Relying Party .

  To require login with Google credentials, see Example Configuration For Multiple Identity Providers in the *IG Configuration Reference*.

- To throttle the number of requests that can access a CF application in a given time, set up an IG throttling filter, as described in Throttling the Rate of Requests to Protected Applications in the *IG Gateway Guide*.

Chapter 3

Implementing the ForgeRock Service Broker

This chapter describes two scenarios for installing the ForgeRock Service Broker for PCF, firstly with Pivotal Cloud Foundry (PCF), and secondly by using the CF command-line tool. Perform **one** of the following installations.

3.1. Installing ForgeRock Service Broker Into Pivotal Cloud Foundry

This section describes how to install the ForgeRock Service Broker into PCF. If you are not using PCF, perform the procedure in Procedure 3.2, "To Install into Cloud Foundry" instead of this procedure.

If you are using PCF, you can use Ops Manager to integrate AM and IG services by installing the ForgeRock Service Broker for PCF tile. To install, you import the ForgeRock Service Broker for PCF tile into your PCF installation, and then configure the ForgeRock Service Broker for PCF by providing the URL and other properties of an AM or IG instance.

When the tile is correctly configured and all changes have been applied, CF application developers can see the AM OAuth 2.0 Service and IG Route Service in the marketplace.

For information about how to use the services, see Chapter 4, "*Using the AM OAuth 2.0 Service*" and Chapter 5, "*Using the IG Route Service*".

Procedure 3.1. To Install into Pivotal Cloud Foundry

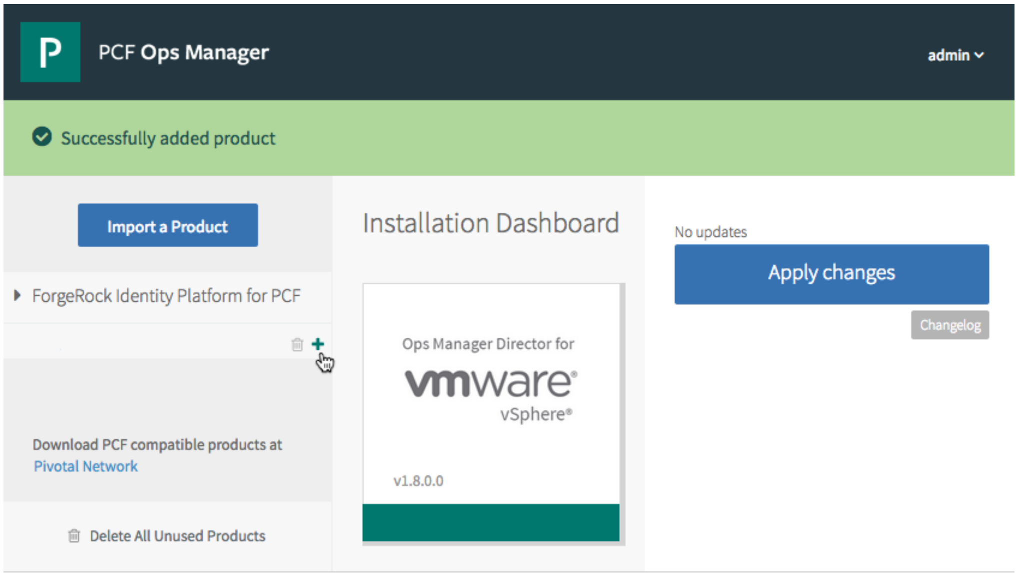
1. In a web browser, navigate to the Pivotal Network, click ForgeRock Identity Platform for PCF, and then download the latest version.

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

3. Browse to the downloaded file, for example `forgerock-broker-x.y.z.pivotal`, 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.

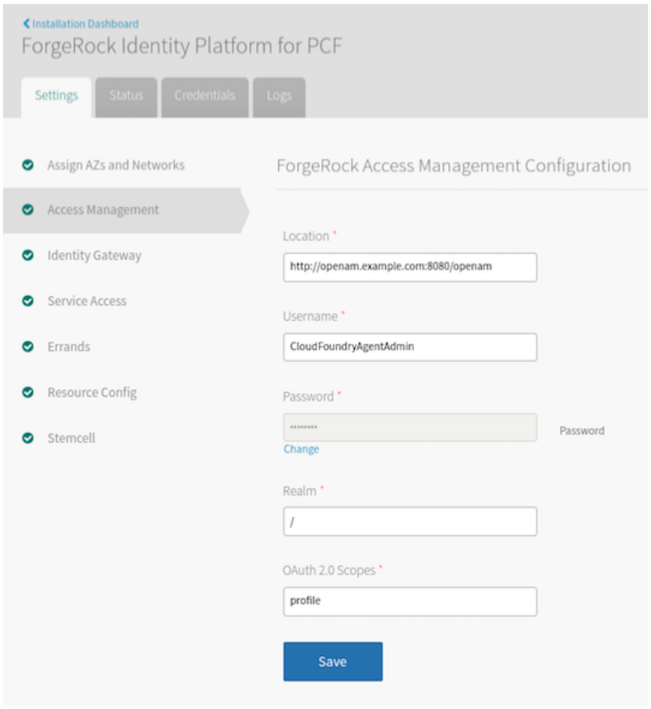


5. On the Installation Dashboard, select the tile ForgeRock Identity Platform for PCF.

6. If you plan to use the AM OAuth 2.0 Service, select the Access Management tab and configure the following properties:

   - `Location`: The URI to the AM instance, for example `http://openam.example.com:8080/openam/`.
   - `Username`: The username you created that the broker will use to authenticate with AM. See Procedure 2.1, "To Prepare AM for ForgeRock Service Broker Installation".
   - `Password`: The password you created that the broker will use to authenticate with AM. See Procedure 2.1, "To Prepare AM 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 is allowed to request when creating access tokens.

   See the following image for an example:



ForgeRock Access Management Configuration

Location *: http://openam.example.com:8080/openam

Username *: CloudFoundryAgentAdmin

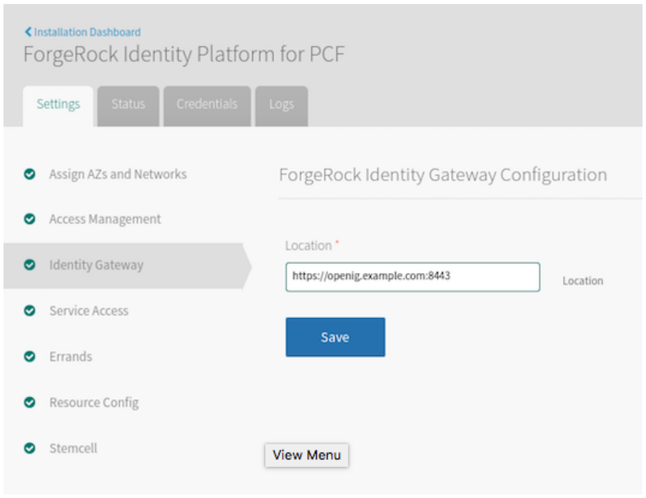
Password *: •••••••• Password
Change

Realm *: /

OAuth 2.0 Scopes *: profile

7. If you plan to use the IG Route Service, select the Identity Gateway tab and configure the URI to the IG instance.

   See the following image for an example:



ForgeRock Identity Gateway Configuration

Location *: https://openig.example.com:8443 Location

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

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

   A long sequence of operations reconfigures PCF to include ForgeRock Identity Platform for PCF.

3.2. Installing ForgeRock Service Broker Into CF

This section describes how to install the ForgeRock Service Broker into CF. If you are using Pivotal Cloud Foundry, see Section 3.1, "Installing ForgeRock Service Broker Into Pivotal Cloud Foundry" instead.

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

Procedure 3.2. To Install into Cloud Foundry

1. Download `service-broker-servlet-2.0.1.war` from the ForgeRock Maven repository .

   You need a ForgeRock BackStage account to access the ForgeRock Maven repositories. For more information, see the Knowledge Base.

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

   ```
   $ cf push forgerockbroker-app -p service-broker-servlet-2.0.1.war
   ```

   Where `forgerockbroker-app` is a name you assign to the new application.

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

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

   a. Configure the following variables for Cloud Foundry to access the broker:

      - `SECURITY_USER_NAME`: Username used by Cloud Foundry to access the broker. You should securely generate a random value.

      - `SECURITY_USER_PASSWORD`: Password used by Cloud Foundry to access the broker. You should securely generate a random value.

   b. To use the AM OAuth 2.0 Service, configure the following variables:

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

      - `OPENAM_USERNAME`: The username that the broker will use to authenticate with AM. See Procedure 2.1, "To Prepare AM for ForgeRock Service Broker Installation".

    - `OPENAM_PASSWORD`: The password that the broker will use to authenticate with AM. See Procedure 2.1, "To Prepare AM for ForgeRock Service Broker Installation".
    - `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.

   c. To use the IG Route Service, configure the following variable:

    - `OPENIG_BASE_URI`: The URI to the IG instance, for example, https://openig.example.com:8443.

      Note that the URL must be HTTPS.

   Example:

```
$ cf set-env forgerockbroker-app SECURITY_USER_NAME Qvd9a7ky
$ cf set-env forgerockbroker-app SECURITY_USER_PASSWORD yNUzCr3C
$ cf set-env forgerockbroker-app OPENAM_BASE_URI http://openam.example.com:8080/openam/
$ cf set-env forgerockbroker-app OPENAM_USERNAME CloudFoundryAgentAdmin
$ cf set-env forgerockbroker-app OPENAM_PASSWORD changeit
$ cf set-env forgerockbroker-app OAUTH2_SCOPES profile
$ cf set-env forgerockbroker-app OPENIG_BASE_URI https://openig.example.com:8443
```

4. Restage the application so that environment variables are applied to the configuration:

```
$ cf restage forgerockbroker-app
```

Tip

You can confirm the applied environment variables by using the **cf env** command. For example:

```
$ cf env forgerockbroker-app
User-Provided:
SECURITY_USER_NAME: Qvd9a7ky
SECURITY_USER_PASSWORD: yNUzCr3C
OAUTH2_SCOPES: profile
OPENAM_BASE_URI: http://openam.example.com:8080/openam/
OPENAM_PASSWORD: changeit
OPENAM_USERNAME: CloudFoundryAgentAdmin
OPENIG_BASE_URI https://openig.example.com:8443
```

5. Find the URN for the application:

```
$ cf app forgerockbroker-app

Showing health and status for app forgerockbroker-app 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}
```

Where:

- `forgerockbroker` is a name you assign to the service broker.
- `{Security User Name}` and `{Security User Password}` have the same values as the `SECURITY_USER_NAME` and `SECURITY_USER_PASSWORD` environment variables created when you restaged the application earlier.
- `{Application URL}` is constructed from the protocol used to access the broker. For example, use `http://`, and the application URN retrieved in the previous step.

Example:

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

7. Grant access to each ForgeRock service by using **cf enable-service-access** commands:

```
$ cf enable-service-access forgerock-am-oauth2
$ cf enable-service-access forgerock-ig-route-service
```

8. Make sure that the services are 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
forgerock-am-oauth2        shared  Uses ForgeRock Access Management to provide OAuth 2.0 authorization
forgerock-ig-route-service shared  Uses ForgeRock Identity Gateway as a Route Service

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

Chapter 4
Using the AM OAuth 2.0 Service

This chapter describes how to bind a CF application to the AM OAuth 2.0 Service. The application can then use AM for OAuth 2.0 functionality.

Binding a CF application to the AM OAuth 2.0 Service creates an OAuth 2.0 agent profile in AM for the application. The application is provided with the agent profile credentials, which can be used in subsequent calls to AM.

4.1. Binding a CF Application to the AM OAuth 2.0 Service

This section explains how to bind a CF application to the AM OAuth 2.0 Service, so that the CF application can use AM for OAuth 2.0 functionality.

Procedure 4.1. To Bind a CF Application to the AM OAuth 2.0 Service

1. Verify that `forgerock-am-oauth2` is available in the catalog:

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

   service                     plans    description
   forgerock-am-oauth2         shared   Uses ForgeRock Access Management to provide OAuth 2.0 authorization

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

   If the `forgerock-am-oauth2` service is not shown, you must install the ForgeRock Service Broker. See Chapter 2, "*Preparing Services for the ForgeRock Service Broker*".

2. Create a shared service instance that the Cloud Foundry application can bind to by using the **cf create-service** command:

   ```
   $ cf create-service forgerock-am-oauth2 shared myOpenAMService
   ```

3. Bind the CF application to the new service instance 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 AM are available to the application in the `VCAP_SERVICES` environment variable. For example:

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

4.2. Obtaining OAuth 2.0 Access Tokens

A CF application that is bound to an instance of the AM OAuth 2.0 Service can make calls to AM to acquire an OAuth 2.0 access token, for use with other services.

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

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

   ```
   VCAP_SERVICES=
   {
       "forgerock-am-oauth2": [
           {
               "name": "myOpenAMService",
               "label": "forgerock-am-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 4.3, "Validating OAuth 2.0 Access Tokens".

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

4.3. Validating OAuth 2.0 Access Tokens

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

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

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

   ```
   VCAP_SERVICES=
   {
       "forgerock-am-oauth2": [
           {
               "name": "myOpenAMService",
               "label": "forgerock-am-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 AM, see the *AM OAuth 2.0 Guide*.

Chapter 5 Using the IG Route Service

5.1. Binding a CF Application to the IG Route Service

This section describes how to bind a CF application to the IG Route Service. Requests to the application are routed through IG.

Procedure 5.1. To Bind a CF Application to the IG Route Service

1. Check that the IG Route Service is 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
   forgerock-ig-route-service shared  Uses ForgeRock Identity Gateway as a Route Service

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

   If `forgerock-ig-route-service` is not shown, install the ForgeRock Service Broker as described in Chapter 2, "*Preparing Services for the ForgeRock Service Broker*".

2. Create a shared instance of the service, to which the CF application can bind:

   ```
   $ cf create-service forgerock-ig-route-service shared myIGRouteService
   ```

   Where:

   - `forgerock-ig-route-service` is the name of the service
   - `myIGRouteService` is the name of the service instance

3. Note that the CF application is not yet bound to the service instance:

   ```
   $ cf routes
   space           host              domain            apps          service
   service-broker  myCFAppHostName   myCFAppDomain     myCFAppName
   ```

4. Bind the CF application to the service instance:

   ```
   $ cf bind-route-service myCFAppDomain myIGRouteService --hostname myCFAppHostName
   ```

   For example:

```
$ cf bind-route-service cfapps-0123.pivotal.io myIGRouteService --hostname spring-music-miototic-
meiosis
```

Where:

- `cfapps-0123.pivotal.io` is the domain of the CF application
- `myIGRouteService` is the name of the service instance
- `spring-music-miototic-meiosis` is the host name of the CF application

5. Check that the CF application is bound to the service instance:

```
$ cf routes
space          host            domain         apps         service
service-broker myCFAppHostName myCFAppDomain  myCFAppName  myIGRouteService
```

To unbind routes, delete services, and remove service brokers, use the **cf unbind-route-service**, **cf delete-service**, and **cf delete-service-broker** commands.

5.2. Testing the Setup

When a CF application is bound to an instance of the IG Route Service, requests to the application are routed through IG before they are passed along to the application.

Depending on the routes you configured in Section 2.2.2, "Adding Routes to Protect a CF Application", the IG Route Service filters and perhaps transforms requests before they are passed to the application, and filters and perhaps transforms responses from the application.

A typical use case could be to configure the IG Route Service to require authentication or authorization before the request is passed to the application. Another could be to throttle the number of requests that are allowed to access it in a given time. Any of the features available in IG and described in the Gateway Guide can be configured in the route service and made available to CF applications.

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.

  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.

A.2. Using the ForgeRock.org Site

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

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

A.3. How to Report Problems or Provide Feedback

If you have found issues or reproducible bugs within ForgeRock Service Broker 2, 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.

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.