User Guide

/ ForgeRock Microgateway 1.0.2

Latest update: 1.0.2

,
, ,

Abstract

Guide to using the ForgeRock® Microgateway for new users and readers evaluating the product.

Table of Contents

Preface iv
1. About the Microgateway 1
2. Downloading and Installing the Microgateway 4
    2.1. Requirements 4
    2.2. Configuring the Network 4
    2.3. Downloading the Microgateway 5
3. Starting and Stopping the Microgateway 6
    3.1. Starting the Microgateway With Default Settings 6
    3.2. Selecting Ports for the Microgateway 7
    3.3. Starting the Microgateway With Custom Settings 7
    3.4. Stopping the Microgateway 8
4. Protecting a Microservice With the Microgateway 9

Preface

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.

This guide introduces the Microgateway, and describes how to set up and run the product.

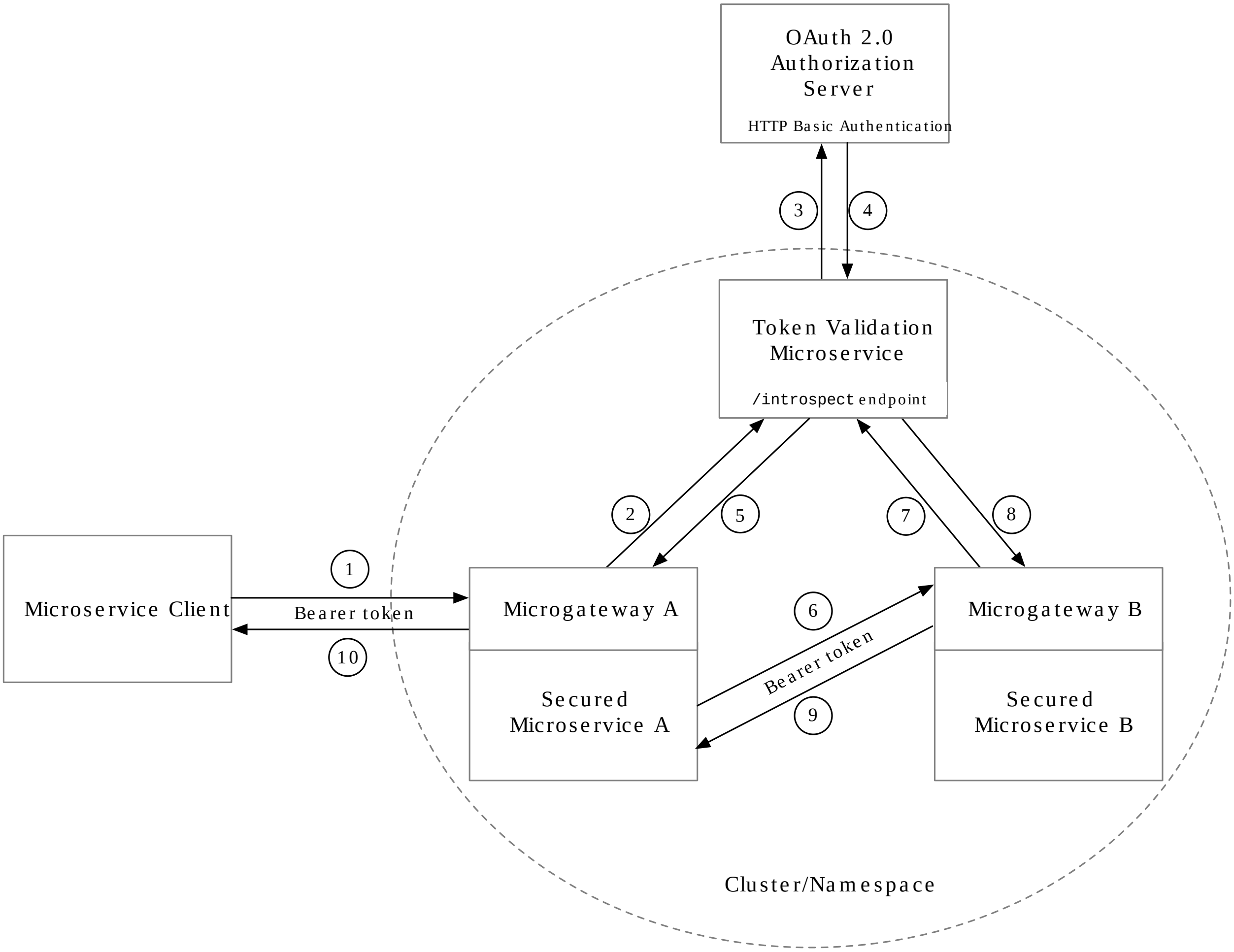
Chapter 1
About the Microgateway

The ForgeRock Microgateway is a standalone Identity Gateway optimized to run in containerized environments. Microgateway is part of the ForgeRock Identity Platform, and shares a common core with ForgeRock Identity Gateway.

Use Microgateway with business microservices to separate the security concerns of your applications from their business logic. For example, use Microgateway with the ForgeRock Token Validation Microservice to provide access_token validation at the edge of your namespace.

For information about features that are shared with IG, see the *IG Documentation*. For a list of IG features that are not provided in this release of the Microgateway, see "*What's New*" in the *Release Notes*.

Example Deployment of the Microgateway



The request is processed in the following sequence:

1. A client requests access to Secured Microservice A, providing a stateful OAuth 2.0 access_token as credentials.

2. Microgateway A intercepts the request, and passes the access_token for validation to the Token Validation Microservice, using the `/introspect` endpoint.

3. The Token Validation Microservice requests the Authorization Server to validate the token.

4. The Authorization Server introspects the token, and sends the introspection result to the Token Validation Microservice.
5. The Token Validation Microservice caches the introspection result, and sends it to Microgateway A, which forwards the result to Secured Microservice A.
6. Secured Microservice A uses the introspection result to decide how to process the request. In this case, it continues processing the request. Secured Microservice A asks for additional information from Secured Microservice B, providing the validated token as credentials.
7. Microgateway B intercepts the request, and passes the access_token to the Token Validation Microservice for validation, using the `/introspect` endpoint.
8. The Token Validation Microservice retrieves the introspection result from the cache, and sends it back to Microgateway B, which forwards the result to Secured Microservice B.
9. Secured Microservice B uses the introspection result to decide how to process the request. In this case it passes its response to Secured Microservice A, through Microgateway B.
10. Secured Microservice A passes its response to the client, through Microgateway A.

Chapter 2

Downloading and Installing the Microgateway

The following sections describe how to download and install the Microgateway:

- "Requirements"
- "Configuring the Network"
- "Downloading the Microgateway"

2.1. Requirements

For detailed information about the requirements for running the Microgateway, see "*Before You Install*" in the *Release Notes*. The following software is required:

- An OAuth 2.0 authentication server, such as ForgeRock Access Management. For information about downloading and using AM, see AM's *Release Notes*.
- Oracle JDK 11 or later versions, or OpenJDK 11 or later versions.

The examples in this guide use the sample application provided with IG. Before you try the procedures, download and run the sample application as described in *Installing the Sample Application*, in the IG *Getting Started*.

2.2. Configuring the Network

Configure the network to route network traffic to the Microgateway and sample application. The examples in the guide assume that:

- The Microgateway is reachable on `http://microgateway.example.com:8080`
- The sample application is reachable on `http://app.example.com:8081/home`
- AM is reachable on `http://openam.example.com:8088/openam`

Before you try out the examples, configure the network to include the hosts.

To Configure the Network

- Add the following additional entry to your `/etc/hosts` file on UNIX systems or `%SystemRoot%\system32\drivers\etc\hosts` on Windows:

```
127.0.0.1  localhost microgateway.example.com app.example.com openam.example.com
```

For more information about host files, see the Wikipedia entry, *Hosts (file)*.

2.3. Downloading the Microgateway

To Download the Microgateway

1. Create a local installation directory for the Microgateway. The examples in this section use `/path/to/microservices`.
2. Download `Microgateway-1.0.2.zip` from the ForgeRock BackStage download site into your local installation directory.
3. Unzip the file:

   ```
   $ unzip Microgateway-1.0.2.zip
   ```

   The directory `/path/to/microservices/identity-gateway` is created.

Chapter 3

Starting and Stopping the Microgateway

> **Important**
>
> In JVM, the default ephemeral Diffie-Hellman (DH) key size is 1024 bits. To support stronger ephemeral DH keys, and protect against weak keys, increase the key size as described in "Starting the Microgateway With Custom Settings".

The following sections describe options for starting and stopping the Microgateway and its sample application:

- "Starting the Microgateway With Default Settings"
- "Selecting Ports for the Microgateway"
- "Starting the Microgateway With Custom Settings"
- "Stopping the Microgateway"

3.1. Starting the Microgateway With Default Settings

Use the following step to start the instance of Microgateway, specifying the configuration directory where the Microgateway looks for configuration files. The Microgateway starts up by default on port `8080`, and `route-system.log` is created in the `logs` directory.

To Start the Microgateway With Default Settings

1. Start the Microgateway, specifying the configuration directory as an argument. In the following example, the Microgateway looks for configuration files in the installation directory.

```
$ /path/to/microservices/identity-gateway/bin/start.sh /path/to/microservices/identity-gateway
...
[vert.x-eventloop-thread-11] INFO  org.forgerock.openig.vertx.Main @system - Server listening on port
 8080
[vert.x-eventloop-thread-11] INFO  org.forgerock.openig.vertx.Main @system - Started in ....ms
```

2. Make sure that the Microgateway is running, in the following ways:

   - Ping Microgateway at http://microgateway.example.com:8080/openig/ping, and make sure an `HTTP 200` is returned.

- Access the Microgateway welcome page at http://microgateway.example.com:8080.
- Display the product version and build information at http://microgateway.example.com:8080/openig/api/info.

3.2. Selecting Ports for the Microgateway

By default the Microgateway runs on a single port, `8080`. To run the Microgateway on a different port, add the configuration file `/path/to/microservices/identity-gateway/config/admin.json`, and restart the Microgateway. The following example runs the Microgateway on port `9090`:

```
{
  "connectors": [
    {
      "port": 9090
    }
  ]
}
```

To run the Microgateway on multiple ports, edit `/path/to/microservices/identity-gateway/config/admin.json` to add the ports to the array, and restart the Microgateway. In the following example, the Microgateway listens on port `9090` and `9091`:

```
{
  "connectors": [
    {
      "port": 9090
    },
    {
      "port": 9091
    }
  ]
}
```

For information about the configuration of `connectors`, see "*AdminHttpApplication (`admin.json`)*" in the *Configuration Reference*.

3.3. Starting the Microgateway With Custom Settings

When the Microgateway starts up, it searches for the file `/path/to/microservices/identity-gateway/bin/env.sh` to configure environment variables, JVM options, and other settings.

Configure `/path/to/microservices/identity-gateway/bin/env.sh` to customize the settings.

The following example specifies environment variables for a secret and JVM options:

```
# Specify JVM options
JVM_OPTS="-Xms256m -Xmx2048m"

# Specify the DH key size for stronger ephemeral DH keys, and to protect against weak keys
JSSE_OPTS="-Djdk.tls.ephemeralDHKeySize=2048"

# Wrap them up into the JAVA_OPTS environment variable
export JAVA_OPTS="${JAVA_OPTS} ${JVM_OPTS} ${JSSE_OPTS}"
```

To Start the Microgateway With Custom Settings

1. Add a file `/path/to/microservices/identity-gateway/bin/env.sh` to define environment variables.

2. Start the Microgateway, specifying the configuration directory as an argument:

   ```
   $ /path/to/microservices/identity-gateway/bin/start.sh /path/to/microservices/identity-gateway
   ...
   ...Server listening on port 8080
   ...Microgateway started in 1106ms
   ```

3.4. Stopping the Microgateway

To Stop the Microgateway

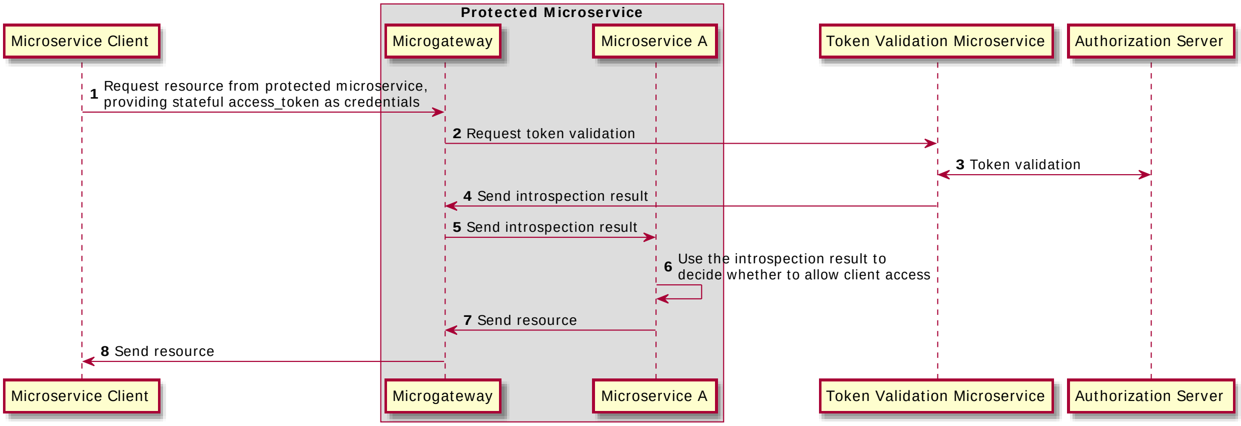
- In the terminal where the Microgateway is running, select CTRL+C to stop the service.

Chapter 4
Protecting a Microservice With the Microgateway

This section describes how to set up the Microgateway to protect a microservice. The section is based on the example in *Introspecting Stateful Access_Tokens With the Token Validation Microservice*, in the Token Validation Microservice *User Guide*.

For information about the architecture, see "Example Deployment of the Microgateway". The following figure illustrates the flow of information when a client requests access to a protected microservice, providing a stateful access_token as credentials:

Request Flow When a Client Requests Access to a Protected Microservice



Protected Microservice
Microservice Client
Microgateway
Microservice A
Token Validation Microservice
Authorization Server
1 Request resource from protected microservice, providing stateful access_token as credentials
2 Request token validation
3 Token validation
4 Send introspection result
5 Send introspection result
6 Use the introspection result to decide whether to allow client access
7 Send resource
8 Send resource
Microservice Client
Microgateway
Microservice A
Token Validation Microservice
Authorization Server

To Protect a Microservice With the Microgateway

Before you start, download and run the sample application as described in *Installing the Sample Application*, in the IG *Getting Started*. The sample application acts as Microservice A.

1. Set up the example in *Introspecting Stateful Access_Tokens With the Token Validation Microservice*, in the Token Validation Microservice *User Guide*.
2. In AM, edit the microservice client to add a scope to access the protected microservice:
   a. Select Applications > OAuth 2.0 > Clients.

   b. Select `microservice-client`, and add the following scope: `microservice-A`

3. Add the following route to the Microgateway configuration as `/path/to/microservices/identity-gateway/config/routes/mgw.json`:

   ```
   {
     "properties": {
       "introspectOAuth2Endpoint": "http://mstokval.example.com:9090"
     },
     "capture": "all",
     "name": "mgw",
     "baseURI": "http://app.example.com:8081",
     "condition": "${matches(request.uri.path, '^/home/mgw')}",
     "handler": {
       "type": "Chain",
       "config": {
         "filters": [
           {
             "name": "OAuth2ResourceServerFilter-1",
             "type": "OAuth2ResourceServerFilter",
             "config": {
               "requireHttps": false,
               "accessTokenResolver": {
                 "name": "TokenIntrospectionAccessTokenResolver-1",
                 "type": "TokenIntrospectionAccessTokenResolver",
                 "config": {
                   "endpoint": "&{introspectOAuth2Endpoint}/introspect",
                   "providerHandler": "ForgeRockClientHandler"
                 }
               },
               "scopes": ["microservice-A"]
             }
           }
         ],
         "handler": "ReverseProxyHandler"
       }
     }
   }
   ```

   Notice the following features of the route:

   - The route matches requests to the Microgateway on `http://microgateway.example.com:8080/home/mgw`, and rebases them to the sample application, on `http://app.example.com:8081`.
   - The OAuth2ResourceServerFilter expects an OAuth 2.0 access_token in the header of the incoming authorization request, with the scope `microservice-A`.
   - If the filter successfully validates the access_token, the ReverseProxyHandler passes the request to the sample application.

To Test the Setup

1. With the Microgateway, Token Validation Microservice, and sample application running, get an access_token from AM, using the scope `microservice-A`:

```
$ mytoken=$(curl \
  --request POST \
  --url http://openam.example.com:8088/openam/oauth2/access_token \
  --user microservice-client:password \
  --data grant_type=client_credentials \
  --data scope=microservice-A --silent | jq -r .access_token)
```

2. View the access_token:

   ```
   $ echo $mytoken
   eyJ...FPg
   ```

3. Call the Microgateway to access microservice A:

   ```
   $ curl -v --header "Authorization: Bearer ${mytoken}" http://microgateway.example.com:8080/home/mgw
   ```

   The home page of the sample application is displayed.