User Guide

/ ForgeRock Token Validation Microservice 1.0.2

Latest update: 1.0.2

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

Copyright © 2019 ForgeRock AS.

Abstract

Guide to using the ForgeRock® Token Validation Microservice for new users and readers evaluating the product.

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

To view a copy of this license, visit https://creativecommons.org/licenses/by-nc-nd/3.0/ or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

ForgeRock® and ForgeRock Identity Platform™ are trademarks of ForgeRock Inc. or its subsidiaries in the U.S. and in other countries. Trademarks are the property of their respective owners.

UNLESS OTHERWISE MUTUALLY AGREED BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

DejaVu Fonts

Bitstream Vera Fonts Copyright

Copyright (c) 2003 by Bitstream, Inc. All Rights Reserved. Bitstream Vera is a trademark of Bitstream, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Bitstream" or the word "Vera".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Bitstream Vera" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL BITSTREAM OR THE GNOME FOUNDATION BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the names of Gnome, the Gnome Foundation, and Bitstream Inc., shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from the Gnome Foundation or Bitstream Inc., respectively. For further information, contact: fonts at gnome dot org.

Arev Fonts Copyright

Copyright (c) 2006 by Tavmjong Bah. All Rights Reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the modifications to the Bitstream Vera Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Tavmjong Bah" or the word "Arev".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Tavmjong Bah Arev" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL TAVMJONG BAH BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the name of Tavmjong Bah shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from Tavmjong Bah. For further information, contact: tavmjong @ free . fr.

FontAwesome Copyright

Copyright (c) 2017 by Dave Gandy, http://fontawesome.io.

This Font Software is licensed under the SIL Open Font License, Version 1.1. See https://opensource.org/licenses/OFL-1.1.

Table of Contents

Copyright © 2019 ForgeRock AS. All rights reserved.

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.

Copyright © 2019 ForgeRock AS. All rights reserved.

Chapter 1

About the Token Validation Microservice

The Token Validation Microservice is delivered as part of the ForgeRock Identity Microservices to introspect and validate OAuth 2.0 access_tokens that adhere to either of the following IETF specifications:

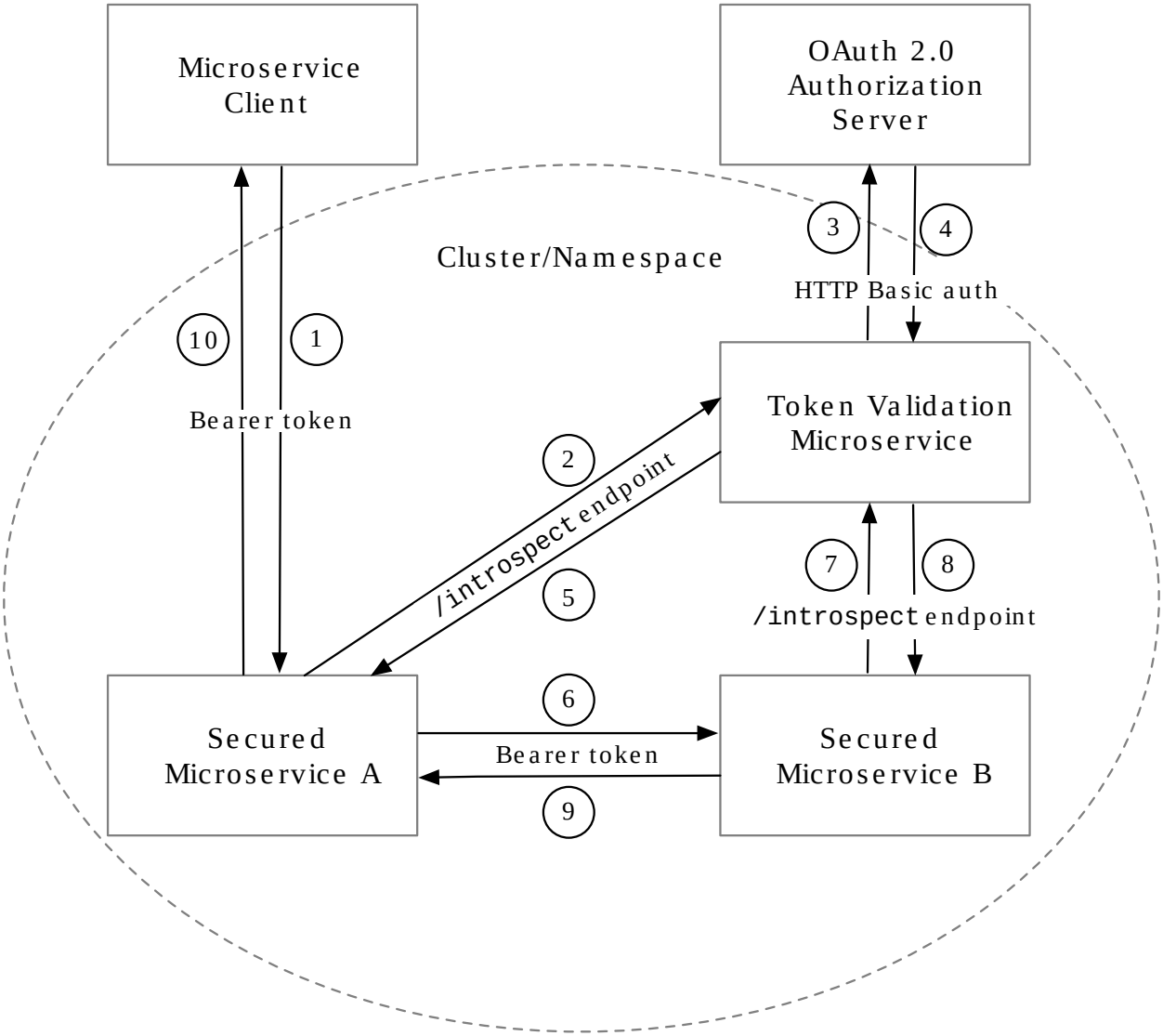
- *OAuth 2.0 Bearer Token Usage*
- *JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants*

The Token Validation Microservice uses the introspection endpoint defined in RFC-7662, *OAuth 2.0 Token Introspection*.

Use the Token Validation Microservice in service-to-service deployments to validate OAuth 2.0 access_tokens. The following figure illustrates a group of Secured Microservices in a container, representing a business service such as ordering, billing, or registration.

Copyright © 2019 ForgeRock AS. All rights reserved.

Example Deployment of the Token Resolution Microservice



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.

Copyright © 2019 ForgeRock AS. All rights reserved.

2. Secured Microservice A 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 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. Secured Microservice B 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 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.
10. Secured Microservice A passes its response to the client.

The Token Validation Microservice can validate stateful and stateless OAuth 2.0 access_tokens. For information about setting up the Token Validation Microservice for each type of access_token, see "*Introspecting Stateful Access_Tokens With the Token Validation Microservice*" and "*Introspecting Stateless Access_Tokens With the Token Validation Microservice*".

Copyright © 2019 ForgeRock AS. All rights reserved.

Chapter 2

Downloading and Installing the Token Validation Microservice

The following sections describe how to download and install the Token Validation Microservice:

- "Requirements"
- "Configuring the Network"
- "Downloading the Token Validation Microservice"

For information about starting and using the Token Validation Microservice, see "*Introspecting Stateful Access_Tokens With the Token Validation Microservice*" and "*Introspecting Stateless Access_Tokens With the Token Validation Microservice*".

2.1. Requirements

For detailed information about the requirements for running the Token Validation Microservice, 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.

2.2. Configuring the Network

Configure the network to route network traffic through the Token Validation Microservice. The examples used in the guide assume that:

- The Token Validation Microservice is reachable on `http://mstokval.example.com:9090`
- AM is reachable on `http://openam.example.com:8088/openam`

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

Copyright © 2019 ForgeRock AS. All rights reserved.

To Configure the Network

- Add the following additional entry to your `/etc/hosts` file on UNIX systems:

  ```
  127.0.0.1  localhost mstokval.example.com openam.example.com
  ```

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

2.3. Downloading the Token Validation Microservice

To Download the Token Validation Microservice Software

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

   ```
   $ unzip MicroserviceTokenValidation-1.0.2.zip
   ```

   The directory `/path/to/microservices/token-validation` is created for the configuration files and startup scripts. When you start the Token Validation Microservice, a log file is created in `/path/to/microservices/token-validation/logs`.

Copyright © 2019 ForgeRock AS. All rights reserved.

Chapter 3

Starting and Stopping the Token Validation Microservice

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 Token Validation Microservice With Custom Settings".

The following sections describe options for starting and stopping the Token Validation Microservice:

- "Starting the Token Validation Microservice With Default Settings"
- "Selecting Ports for the Token Validation Microservice"
- "Starting the Token Validation Microservice With Custom Settings"
- "Stopping the Token Validation Microservice"

3.1. Starting the Token Validation Microservice With Default Settings

This section describes how to start the Token Validation Microservice, specifying the configuration directory where the Token Validation Microservice looks for configuration files. An error is produced if a `config.json` is not available in the configuration directory.

The Token Validation Microservice starts up on the ports listed in `admin.json`, or by default on port `9090`.

To Start the Token Validation Microservice With Default Settings

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

```
$ /path/to/microservices/token-validation/bin/start.sh /path/to/microservices/token-validation
...
...Server listening on port 9090
...Token Validation Microservice started in 1106ms
```


Copyright © 2019 ForgeRock AS. All rights reserved.

2. Make sure that the Token Validation Microservice is running, in the following ways:

   - Ping the Token Validation Microservice at http://mstokval.example.com:9090/ping, and make sure an `HTTP 200` is returned.
   - Display the product version and build information at http://mstokval.example.com:9090/info.

3.2. Selecting Ports for the Token Validation Microservice

By default the Token Validation Microservice runs on a single port, `9090`. To run the Token Validation Microservice on a different port, edit the configuration file `/path/to/microservices/token-validation/config/admin.json`. The following example runs the Token Validation Microservice on port `9091`:

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

To run the Token Validation Microservice on multiple ports, add the ports to the array. In the following example, the Token Validation Microservice listens on ports `9091` and `9092`

```
{
  "connectors": [
    {
      "port": 9091
    },
    {
      "port": 9092
    }
  ]
}
```

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

3.3. Starting the Token Validation Microservice With Custom Settings

When the Token Validation Microservice starts up, it searches for the file `/path/to/microservices/token-validation/bin/env.sh` to configure environment variables, JVM options, and other settings.

Configure `/path/to/microservices/token-validation/bin/env.sh` to customize the settings.

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

Copyright © 2019 ForgeRock AS. All rights reserved.

```
# 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 Token Validation Microservice With Custom Settings

1. Add a file `/path/to/microservices/token-validation/bin/env.sh` to define environment variables.
2. Start the Token Validation Microservice, specifying the configuration directory as an argument. In the following example, the Token Validation Microservice looks for configuration files in the installation directory.

```
$ /path/to/microservices/token-validation/bin/start.sh /path/to/microservices/token-validation
...
...Server listening on port 9090
...Token Validation Microservice started in 1106ms
```

3.4. Stopping the Token Validation Microservice

To Stop the Token Validation Microservice

- In the terminal where the Token Validation Microservice is running, select CTRL-C to stop the service.

Copyright © 2019 ForgeRock AS. All rights reserved.

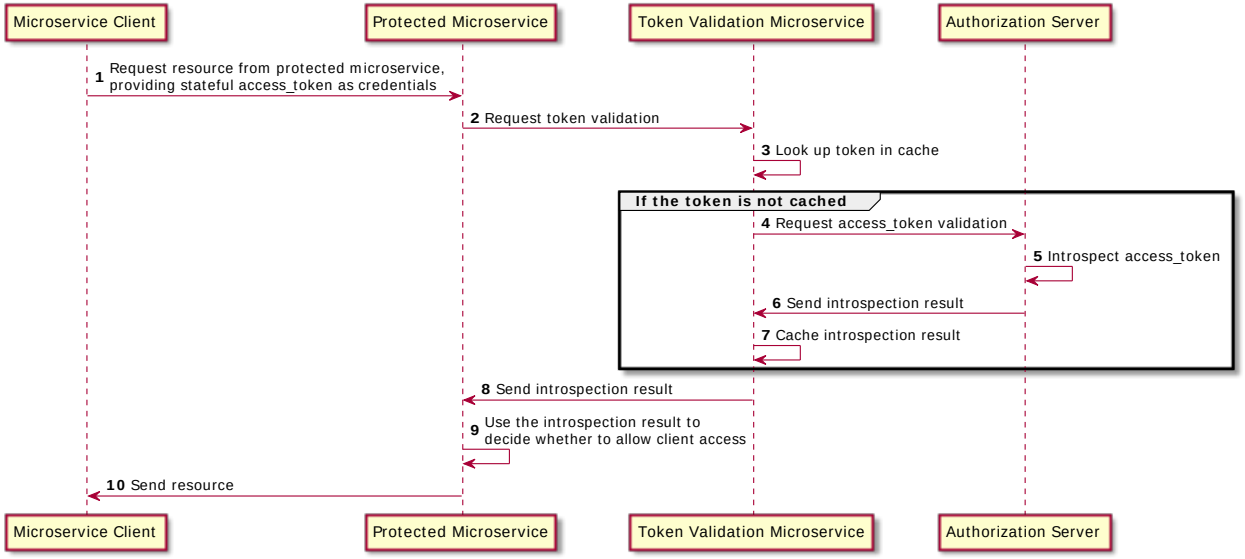
Chapter 4

Introspecting Stateful Access_Tokens With the Token Validation Microservice

This section describes how to set up the Token Validation Microservice to introspect stateful access_tokens, using AM as the authorization server. For information about the architecture, see "Example Deployment of the Token Resolution Microservice".

The following figure illustrates the flow of information when a client requests access to a protected microservice, providing a stateful access_token as credentials:

Validating Stateful OAuth 2.0 Tokens



The following sections describe how to set up the Token Validation Microservice to introspect stateful access_tokens:

- "Setting Up AM as An Authorization Server for Stateful Access_Tokens"
- "Setting Up the Token Validation Microservice for Stateful Access_Token Introspection"
- "Testing Stateful Access_Token Introspection"

Copyright © 2019 ForgeRock AS. All rights reserved.

- "Capturing Calls to AM to Introspect Access_Tokens"

4.1. Setting Up AM as An Authorization Server for Stateful Access_Tokens

Before you start, install and configure AM on http://openam.example.com:8088/openam, with the default configuration. If you use a different configuration, substitute in the tutorial accordingly.

For information about downloading and using AM, see AM's *Release Notes*. For more information about configuring AM as an OAuth 2.0 authorization service, see Configuring the OAuth 2.0 Provider Service in the AM *OAuth 2.0 Guide*.

To Set Up AM as An Authorization Server for Stateful Access_Tokens

1. Configure an OAuth 2.0 Authorization Server:
   a. In the top level realm, select Configure OAuth Provider > Configure OAuth 2.0.
   b. Accept all of the default values and select Create.
2. Create an OAuth 2.0 client to request OAuth 2.0 access_tokens, using client credentials for authentication:
   a. Select Applications > OAuth 2.0.
   b. Add a client with the following values:
      - Client ID: `microservice-client`
      - Client secret: `password`
      - Scope(s): `client-scope`
   c. (From AM 6.5) On the Advanced tab, select the following option:
      - Grant Types: `Client Credentials`
3. Create an OAuth 2.0 Client authorized to examine (introspect) tokens:
   a. In the top level realm, select Applications > OAuth 2.0.
   b. Add a client with the following values:
      - Client ID: `token-validation`
      - Client secret `password`
      - Scope(s): `am-introspect-all-tokens`

Copyright © 2019 ForgeRock AS. All rights reserved.

4.2. Setting Up the Token Validation Microservice for Stateful Access_Token Introspection

This section describes steps to set up the Token Validation Microservice to introspect stateful access_tokens, at the introspection endpoint.

Before you start, download and install the Token Validation Microservice as described in "*Downloading and Installing the Token Validation Microservice*".

To Set Up the Token Validation Microservice for Stateful Access_Token Introspection

1. Prepare the Token Validation Microservice configuration and property files in `/path/to/microservices/token-validation/config`:

   a. Rename `config-template-stateful-cache.json` to `config.json`:

```
{
  "properties": {
    "openAmOAuth2Endpoint": "http://openam.example.com:8088/openam/oauth2",
    "oauth2ClientId": "token-validation"
  },
  "introspectionConfig": {
    "accessTokenResolver": {
      "type": "CacheAccessTokenResolver",
      "config": {
        "maximumTimeToCache": "2 minutes",
        "delegate": {
          "type": "TokenIntrospectionAccessTokenResolver",
          "config": {
            "endpoint": "&{openAmOAuth2Endpoint}/introspect",
            "providerHandler": {
              "type": "Chain",
              "config": {
                "filters": [
                  {
                    "type": "HttpBasicAuthenticationClientFilter",
                    "config": {
                      "urlEncodeCredentials": true,
                      "username": "${oauth2ClientId}",
                      "passwordSecretId": "oauth2.client.secret",
                      "secretsProvider": {
                        "type": "FileSystemSecretStore",
                        "config": {
                          "directory": "${environment.configDirectory}/secrets",
                          "format": "BASE64"
                        }
                      }
                    }
                  }
                ],
                "handler": "ForgeRockClientHandler"
              }
            }
          }
        }
```


Copyright © 2019 ForgeRock AS. All rights reserved.

```
      }
    }
  }
}
```

Notice the following features of the file:

- The `properties` section declares values used in the AM configuration. If necessary, change these values to match your AM configuration.
- The `introspectionConfig` object calls a CacheAccessTokenResolver to look in the cache for the presented access_token:
  - If the access_token is cached, the filter sends the introspection result back to the client microservice.
  - If the token is not cached, the filter delegates access_token introspection to the TokenIntrospectionAccessTokenResolver and then caches the result.
- The TokenIntrospectionAccessTokenResolver uses the HttpBasicAuthenticationClientFilter to authenticate itself to AM as the OAuth 2.0 client `token-validation`.

  The password is provided by a FileSystemSecretStore. For more information, see *FileSystemSecretStore*, in the IG *Configuration Reference*

b. In `admin.json`, make sure that the ports for the Token Validation Microservice are correct:

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

2. Verify that the file `/path/to/microservices/token-validation/config/secrets/oauth2.client.secret` has the following content:

```
cGFzc3dvcmQ=
```

This file is called by the FileSystemSecretStore, and contains the base64-encoded password for the OAuth 2.0 client `token-validation`.

3. Start the Token Validation Microservice, specifying the configuration directory as an argument. In the following example, the Token Validation Microservice looks for configuration files in the installation directory.

```
$ /path/to/microservices/token-validation/bin/start.sh /path/to/microservices/token-validation
...
...Server listening on port 9090
...Token Validation Microservice started in 1106ms
```

4. Make sure that the Token Validation Microservice is running, in the following ways:

Copyright © 2019 ForgeRock AS. All rights reserved.

- Ping the Token Validation Microservice at http://mstokval.example.com:9090/ping, and make sure an `HTTP 200` is returned.
- Display the product version and build information at http://mstokval.example.com:9090/info.

4.3. Testing Stateful Access_Token Introspection

To Test Stateful Access_Token Introspection

1. Get an access_token from AM:

```
$ 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=client-scope --silent | jq -r .access_token)
```

2. View the access_token:

```
$ echo $mytoken
Abc...xyz
```

3. Call the Token Validation Microservice to introspect the access_token:

```
$ curl --data "token=${mytoken}" http://mstokval.example.com:9090/introspect
{
  "active": true,
  "scope": "client-scope",
  "client_id": "microservice-client",
  "user_id": "microservice-client",
  "token_type": "Bearer",
  "exp": 155...587,
  "sub": "microservice-client",
  "iss": "http://openam.example.com:8088/openam/oauth2"
}
```

4.4. Capturing Calls to AM to Introspect Access_Tokens

When an access_token is cached, IG can reuse the token information without repeatedly asking the authorization server to verify the access_token. The following procedure gives an example of how to capture the call to AM to introspect the access_token.

To Capture Calls to AM to Introspect Access_Tokens

1. Set up AM and the Token Validation Microservice as described in this chapter, and then change the ForgeRockClientHandler in `config.json` from:

Copyright © 2019 ForgeRock AS. All rights reserved.

```
"handler": "ForgeRockClientHandler"
```

to

```
"handler": {
  "type": "Delegate",
  "config": {
    "delegate": "ForgeRockClientHandler"
  },
  "capture": "all"
}
```

The capture logs the call to AM to introspect the access_token.

2. Restart the Token Validation Microservice, and then introspect an access_token as described in "Testing Stateful Access_Token Introspection".

3. In `/path/to/microservices/token-validation/logs/token-validation.log`, note the call to AM to validate the access_token:

```
POST http://openam.example.com:8088/openam/oauth2/introspect HTTP/1.1
```

4. Within the duration set by `maximumTimeToCache`, call the Token Validation Microservice to introspect the access_token again.

   Note that this time, because the access_token is cached, the log does not include a call to AM.

Copyright © 2019 ForgeRock AS. All rights reserved.

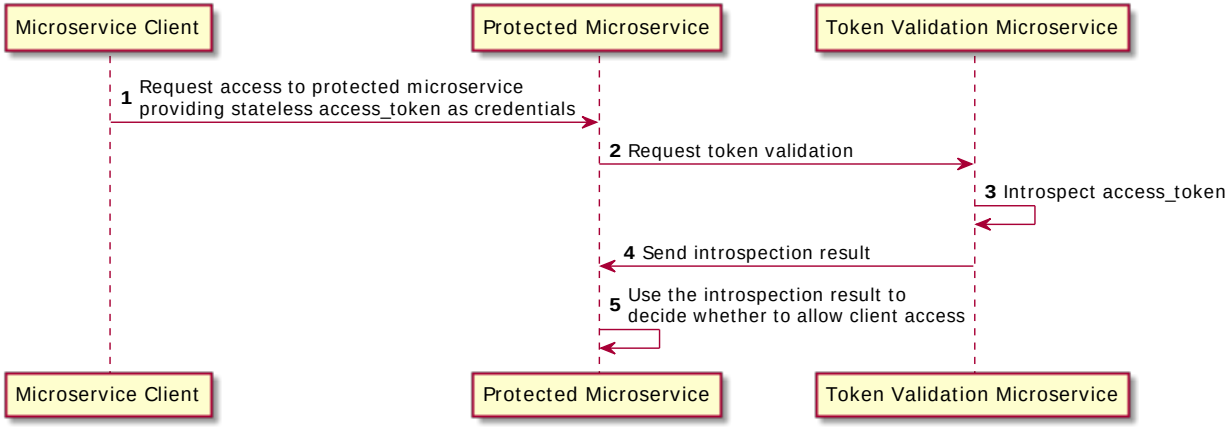
Chapter 5

Introspecting Stateless Access_Tokens With the Token Validation Microservice

This section describes how to set up the Token Validation Microservice to introspect stateless access_tokens locally, without referring to an authorization server. For information about the architecture, see "Example Deployment of the Token Resolution Microservice".

The following figure illustrates the flow of information when a client requests access to a protected microservice, providing a stateless access_token as credentials:

Request Flow When Introspecting Stateless Access_Tokens



The following sections describe how to set up the Token Validation Microservice to introspect stateless access_tokens:

- "Setting Up AM as An Authorization Server for Stateless Access_Tokens"
- "Setting Up the Token Validation Microservice for Stateless Access_Token Introspection"
- "Testing Stateless Access_Token Introspection"

Copyright © 2019 ForgeRock AS. All rights reserved.

5.1. Setting Up AM as An Authorization Server for Stateless Access_Tokens

Before you start, install and configure AM on http://openam.example.com:8088/openam, with the default configuration. If you use a different configuration, substitute in the tutorial accordingly.

For information about downloading and using AM, see AM's *Release Notes*. For more information about configuring AM as an OAuth 2.0 authorization service, see Configuring the OAuth 2.0 Provider Service in the AM *OAuth 2.0 Guide*.

To Set Up AM as An Authorization Server for Stateless Access_Tokens

1. Configure an OAuth 2.0 Authorization Provider to provide signed stateless access_tokens:

    a. Select Configure OAuth Provider > Configure OAuth 2.0, accept the default values, and select Create.

    b. Select Services > OAuth2 Provider.

    c. On the Core tab, select the following option:

    - (From AM 6.5) Use Client-Based Access & Refresh Tokens: `on`
    - (From AM 6) Use Stateless Access & Refresh Tokens: `on`

    Save the configuration.

    d. On the Advanced tab, select the following option:

    - OAuth2 Token Signing Algorithm: `RS256`

2. Create an OAuth 2.0 client to request OAuth 2.0 access_tokens, using client credentials for authentication:

    a. Select Applications > OAuth 2.0.

    b. Add a client with the following values:

    - Client ID: `microservice-client`
    - Client secret: `password`
    - Scope(s): `client-scope`

    c. (From AM 6.5) On the Advanced tab, select the following option:

    - Grant Types: `Client Credentials`

Copyright © 2019 ForgeRock AS. All rights reserved.

5.2. Setting Up the Token Validation Microservice for Stateless Access_Token Introspection

This section describes steps to set up Token Validation Microservice to introspect stateless access_tokens, at the introspection endpoint.

Before you start, download and install the Token Validation Microservice as described in "*Downloading and Installing the Token Validation Microservice*".

To Set Up the Token Validation Microservice for Stateless Access_Token Introspection

1. Prepare the Token Validation Microservice configuration and property files in `/path/to/microservices/token-validation/config`:

   a. Rename `config-template-stateless.json` to `config.json`:

   ```
   {
     "properties": {
       "openAmOAuth2Endpoint": "http://openam.example.com:8088/openam/oauth2"
     },
     "introspectionConfig": {
       "accessTokenResolver": {
         "type": "StatelessAccessTokenResolver",
         "config": {
           "issuer": "&{openAmOAuth2Endpoint}",
           "secretsProvider": {
             "type": "JwkSetSecretStore",
             "config": {
               "jwkUrl": "&{openAmOAuth2Endpoint}/connect/jwk_uri"
             }
           },
           "verificationSecretId": "stateless.access.token.verification.key"
         }
       }
     }
   }
   ```

   Notice the following features of the file:

   - The `properties` section declares values used in the AM configuration.
   - The `introspectionConfig` object delegates access_token introspection to the StatelessAccessTokenResolver.
   - The StatelessAccessTokenResolver uses a JwkSetSecretStore, which specifies the URL to a JWK set on AM that contains signing keys identified by a `kid`.

     The value of `verificationSecretId` must be specified, but it is not used unless the signed access_token doesn't contain a `kid`, or the JWK set doesn't contain a matching key.

   b. In `admin.json`, make sure that the ports for the Token Validation Microservice are correct:

Copyright © 2019 ForgeRock AS. All rights reserved.

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

2. Start the Token Validation Microservice, specifying the configuration directory as an argument. In the following example, the Token Validation Microservice looks for configuration files in the installation directory.

```
$ /path/to/microservices/token-validation/bin/start.sh /path/to/microservices/token-validation
...
...Server listening on port 9090
...Token Validation Microservice started in 1106ms
```

3. Make sure that the Token Validation Microservice is running, in the following ways:

   - Ping the Token Validation Microservice at http://mstokval.example.com:9090/ping, and make sure an `HTTP 200` is returned.
   - Display the product version and build information at http://mstokval.example.com:9090/info.

5.3. Testing Stateless Access_Token Introspection

To Test Stateless Access_Token Introspection

1. Get an access_token from AM:

```
$ 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=client-scope --silent | jq -r .access_token)
```

2. View the access_token:

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

   Note that the token has the structure of a stateless access_token.

3. Call the Token Validation Microservice to introspect the access_token:

Copyright © 2019 ForgeRock AS. All rights reserved.

```
$ curl --data "token=${mytoken}" http://mstokval.example.com:9090/introspect
{
 "active": true,
 "sub": "microservice-client",
 "cts": "OAUTH2_STATELESS_GRANT",
 "auditTrackingId": "3b5...083",
 "iss": "http://openam.example.com:8088/openam/oauth2",
 "tokenName": "access_token",
 "token_type": "Bearer",
 "authGrantId": "26V..._oc",
 "aud": "microservice-client",
 "nbf": 1559895523,
 "grant_type": "client_credentials",
 "scope": ["client-scope"],
 "auth_time": 155...523,
 "realm": "/",
 "exp": 155...123,
 "iat": 155...523,
 "expires_in": 3600,
 "jti": "cXQ...Lko"
}
```


Copyright © 2019 ForgeRock AS. All rights reserved.

Chapter 6
Protecting the `/introspect` Endpoint

By default, no special credentials or privileges are required to access the `/introspect` endpoint. The following procedure gives an example of how to manage access to the `/introspect` endpoint:

To Protect the `/introspect` Endpoint

1. Set up token introspection, as described in "*Introspecting Stateful Access_Tokens With the Token Validation Microservice*" or "*Introspecting Stateless Access_Tokens With the Token Validation Microservice*".

2. Add the following script to the Token Validation Microservice configuration as `/path/to/microservices/token-validation/scripts/groovy/BasicAuthResourceServerFilter.groovy`

```
/**
 * This scripts is a simple implementation of HTTP Basic Authentication on
 * server side.
 *
 * It expects the following arguments:
 *  - realm: the realm to display when the user-agent prompts for
 *    username and password if none were provided.
 *  - username: the expected username
 *  - password: the expected password
 */

import static org.forgerock.util.promise.Promises.newResultPromise;

import java.nio.charset.Charset;
import org.forgerock.util.encode.Base64;

String authorizationHeader = request.getHeaders().getFirst("Authorization");
if (authorizationHeader == null) {
    // No credentials provided, reply that they are needed.
    Response response = new Response(Status.UNAUTHORIZED);
    response.getHeaders().put("WWW-Authenticate", "Basic realm=\"" + realm + "\"");
    return newResultPromise(response);
}

String expectedAuthorization = "Basic " + Base64.encode((username + ":" +
 password).getBytes(Charset.defaultCharset()))
if (!expectedAuthorization.equals(authorizationHeader)) {
    return newResultPromise(new Response(Status.FORBIDDEN));
}
// Credentials are as expected, let's continue
return next.handle(context, request);
```


Copyright © 2019 ForgeRock AS. All rights reserved.

The script is a simple implementation of the HTTP Basic Authentication mechanism. For information about scripting filters and handlers, see *Extending IG*, in the IG *Gateway Guide*.

3. Add the following heap object to `/path/to/microservices/token-validation/config/config.json`:

```
"heap": [{
  "name": "ProtectionFilter",
  "type": "ScriptableFilter",
  "config": {
    "type": "application/x-groovy",
    "file": "BasicAuthResourceServerFilter.groovy",
    "args": {
      "realm": "/",
      "username": "introspect",
      "password": "password"
    }
  }
}]
```

4. Restart the Token Validation Microservice to reload the configuration.

To Test the Setup

1. Get an access_token from AM:

```
$ 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=client-scope --silent | jq -r .access_token)
```

2. View the access_token:

```
$ echo $mytoken
```

3. Call the Token Validation Microservice to introspect the access_token without using credentials, and note that access is denied:

```
$ curl -v --data "token=${mytoken}" http://mstokval.example.com:9090/introspect
...
HTTP/1.1 401 Unauthorized
...
```

4. Introspect the access_token again, providing the credentials required in the ScriptableFilter, and note that the access_token information is returned:

```
$ curl -v --data "token=${mytoken}" http://mstokval.example.com:9090/introspect --user
 introspect:password
...
HTTP/1.1 200 OK
...
```


Copyright © 2019 ForgeRock AS. All rights reserved.

Chapter 7

Monitoring the Token Validation Microservice

All ForgeRock products expose monitoring endpoints where metrics are exposed. When the Token Validation Microservice is running, metrics are exposed at the Prometheus Scrape Endpoint and Common REST Monitoring Endpoint. By default, no authentication is required to access the monitoring endpoints.

The following procedures describe how to monitor the Token Validation Microservice, and protect the monitoring endpoints:

- "To Query the Monitoring Endpoints"
- "To Protect the Monitoring Endpoints"

To Query the Monitoring Endpoints

1. Set up the the Token Validation Microservice as described in "*Introspecting Stateful Access_Tokens With the Token Validation Microservice*" or "*Introspecting Stateless Access_Tokens With the Token Validation Microservice*".

2. Query the Prometheus Scrape Endpoint, and note that monitoring information is returned:

```
$ curl "http://mstokval.example.com:9090/metrics/prometheus"
# HELP tv_request_active Generated from Dropwizard metric import (metric=request.active, type=gauge)
# TYPE tv_request_active gauge
tv_request_active 0.0
# HELP tv_request_total Generated from Dropwizard metric import (metric=request, type=counter)
# TYPE tv_request_total counter
tv_request_total 1.0
...
```

3. Query the Common REST Monitoring Endpoint, and note that monitoring information is returned:

```
$ curl "http://mstokval.example.com:9090/metrics/api?
_prettyPrint=true&_sortKeys=_id&_queryFilter=true"
{
  "result" : [ {
    "_id" : "request",
    "count" : 1,
    "_type" : "counter"
  }, {
    "_id" : "request.active",
    "value" : 0.0,
    "_type" : "gauge"
  },
...
```


Copyright © 2019 ForgeRock AS. All rights reserved.

To Protect the Monitoring Endpoints

1. Add the following script to the Token Validation Microservice configuration as `/path/to/microservices/token-validation/scripts/groovy/BasicAuthResourceServerFilter.groovy`

```
/**
 * This scripts is a simple implementation of HTTP Basic Authentication on
 * server side.
 *
 * It expects the following arguments:
 *  - realm: the realm to display when the user-agent prompts for
 *    username and password if none were provided.
 *  - username: the expected username
 *  - password: the expected password
 */

import static org.forgerock.util.promise.Promises.newResultPromise;

import java.nio.charset.Charset;
import org.forgerock.util.encode.Base64;

String authorizationHeader = request.getHeaders().getFirst("Authorization");
if (authorizationHeader == null) {
    // No credentials provided, reply that they are needed.
    Response response = new Response(Status.UNAUTHORIZED);
    response.getHeaders().put("WWW-Authenticate", "Basic realm=\"" + realm + "\"");
    return newResultPromise(response);
}

String expectedAuthorization = "Basic " + Base64.encode((username + ":" +
 password).getBytes(Charset.defaultCharset()))
if (!expectedAuthorization.equals(authorizationHeader)) {
    return newResultPromise(new Response(Status.FORBIDDEN));
}
// Credentials are as expected, let's continue
return next.handle(context, request);
```

   The script is a simple implementation of the HTTP Basic Authentication mechanism. For information about scripting filters and handlers, see *Extending IG*, in the IG *Gateway Guide*.

2. Add the following heap object to `/path/to/microservices/token-validation/config/config.json`:

```
"heap": [{
  "name": "MetricsProtectionFilter",
  "type": "ScriptableFilter",
  "config": {
    "type": "application/x-groovy",
    "file": "BasicAuthResourceServerFilter.groovy",
    "args": {
      "realm": "/",
      "username": "metric",
      "password": "password"
    }
  }
}]
```


Copyright © 2019 ForgeRock AS. All rights reserved.

3. Restart the Token Validation Microservice to reload the configuration.

4. Query the monitoring endpoints again, as described in "To Query the Monitoring Endpoints", and note that access is denied:

```
$ curl "http://mstokval.example.com:9090/metrics/prometheus" -v
...
HTTP/1.1 401 Unauthorized
...
```

```
$ curl "http://mstokval.example.com:9090/metrics/api?
_prettyPrint=true&_sortKeys=_id&_queryFilter=true" -v
...
HTTP/1.1 401 Unauthorized
...
```

5. Query the monitoring endpoints again, providing the credentials required in the ScriptableFilter, and note that the monitoring information is returned:

```
$ curl "http://mstokval.example.com:9090/metrics/api?
_prettyPrint=true&_sortKeys=_id&_queryFilter=true" --user metric:password
```


Copyright © 2019 ForgeRock AS. All rights reserved.

Chapter 8

Querying and Disabling the Information Endpoint

When the Token Validation Microservice is running, version information is exposed at the information endpoint. The following sections describe how to query and disable the information endpoint:

- "To Query the Version Information Endpoint"
- "To Disable the Version Information Endpoint"

To Query the Version Information Endpoint

- Query the version information endpoint, and note that the following information is returned:

```
$ curl "http://mstokval.example.com:9090/info"
{"version":"...","revision":"...","branch":"...","timestamp":...}
...
```

To Disable the Version Information Endpoint

1. Add the following attribute to the `admin.json` file:

   ```
   "infoEndpointEnabled": false
   ```

   For example:

   ```
   {
     "connectors": [
       {
         "port": 9090
       }
     ],
     "infoEndpointEnabled": false
   }
   ```

   For more information, see "*Service Configuration (`admin.json`)*" in the *Configuration Reference*.

2. Restart the Token Validation Microservice to reload the configuration, and then query the version information endpoint again. Note this time that no information is returned.

Copyright © 2019 ForgeRock AS. All rights reserved.