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

Abstract

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

Table of Contents

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.

Example installation for this guide

Unless otherwise stated, the examples in this guide assume the following installation:

- ForgeRock Token Validation Microservice installed on `http://mstokval.example.com:9090`.
- One of the following:
  - ForgeRock Access Management 7.0.x, installed on `http://openam.example.com:8088`, with the default configuration.
  - ForgeRock Identity Cloud accessible on `https://myTenant.forgeblocks.com` with the default configuration.

If you use a different configuration, substitute the configuration in the procedures.

Chapter 1

About the ForgeRock Token Validation Microservice

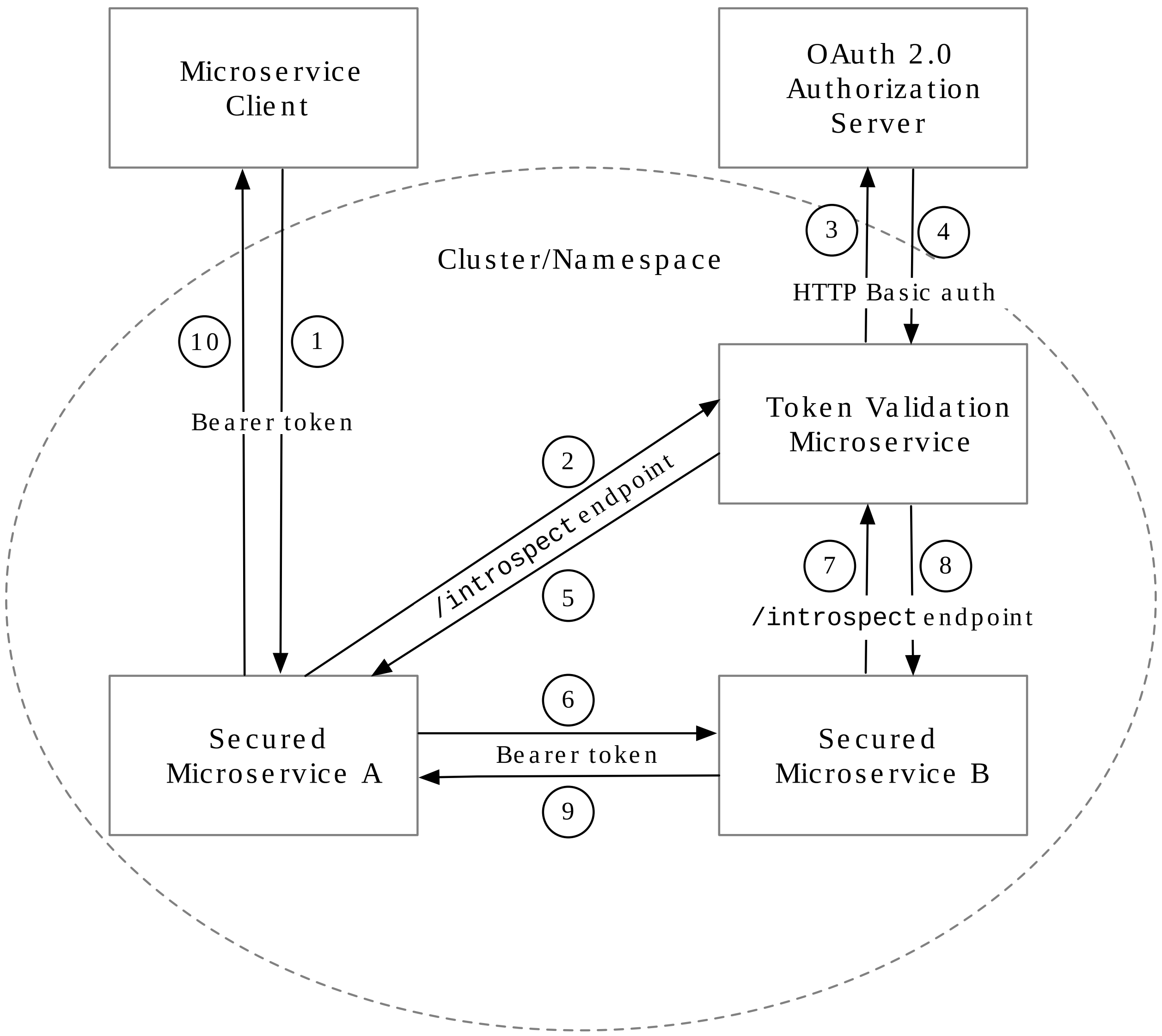
The ForgeRock Token Validation Microservice (TVMS) is delivered as part of the ForgeRock Identity Microservices to introspect and validate OAuth 2.0 access tokens that adhere to either of the following IETF specifications:

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

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

Use TVMS 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.

Example Deployment of the ForgeRock Token Validation 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.

2. Secured Microservice A passes the access_token for validation to TVMS, using the `/introspect` endpoint.
3. TVMS requests the Authorization Server to validate the token.
4. The Authorization Server introspects the token, and sends the introspection result to TVMS.
5. TVMS 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 TVMS for validation, using the `/introspect` endpoint.
8. TVMS 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.

TVMS can validate stateful and stateless OAuth 2.0 access tokens. For information about setting up TVMS for each type of access_token, see "*Introspecting Stateful Access Tokens*" and "*Introspecting Stateless Access Tokens*".

Chapter 2
Downloading and Installing TVMS

The following sections describe how to download and install TVMS:

- "Requirements"
- "Configuring the Network"
- "Downloading TVMS"

For information about starting and using TVMS, see "*Introspecting Stateful Access Tokens*" and "*Introspecting Stateless Access Tokens*".

Requirements

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

- An OAuth 2.0 authentication server, such as Identity Cloud or Access Management.

  For more information, see the *Identity Cloud docs* or *AM docs*.

- Oracle JDK 11 or later versions, or OpenJDK 11 or later versions.

Configuring the Network

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

- TVMS is reachable on `http://mstokval.example.com:9090`
- Identity Cloud is reachable on `https://myTenant.forgeblocks.com`
- AM is reachable on `http://openam.example.com:8088`

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:

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

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

Downloading TVMS

1. Create a local installation directory for TVMS. 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 TVMS, a log file is created in `/path/to/microservices/token-validation/logs`.

Chapter 3
Starting and Stopping TVMS

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 TVMS With Custom Settings".

The following sections describe options for starting and stopping the TVMS:

- "Starting TVMS With Default Settings"
- "Selecting Ports for TVMS"
- "Starting TVMS With Custom Settings"
- "Stopping TVMS"

Starting TVMS With Default Settings

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

TVMS starts up on the ports listed in `admin.json`, or by default on port `9090`.

1. Start TVMS, specifying the configuration directory as an argument. In the following example, TVMS 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
   ```

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

   - Ping TVMS 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.

Selecting Ports for TVMS

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

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

To run TVMS on multiple ports, add the ports to the array. In the following example, TVMS 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*.

Starting TVMS With Custom Settings

When TVMS 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:

```
# 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}"
```

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

2. Start TVMS, specifying the configuration directory as an argument. In the following example, TVMS 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
```

Stopping TVMS

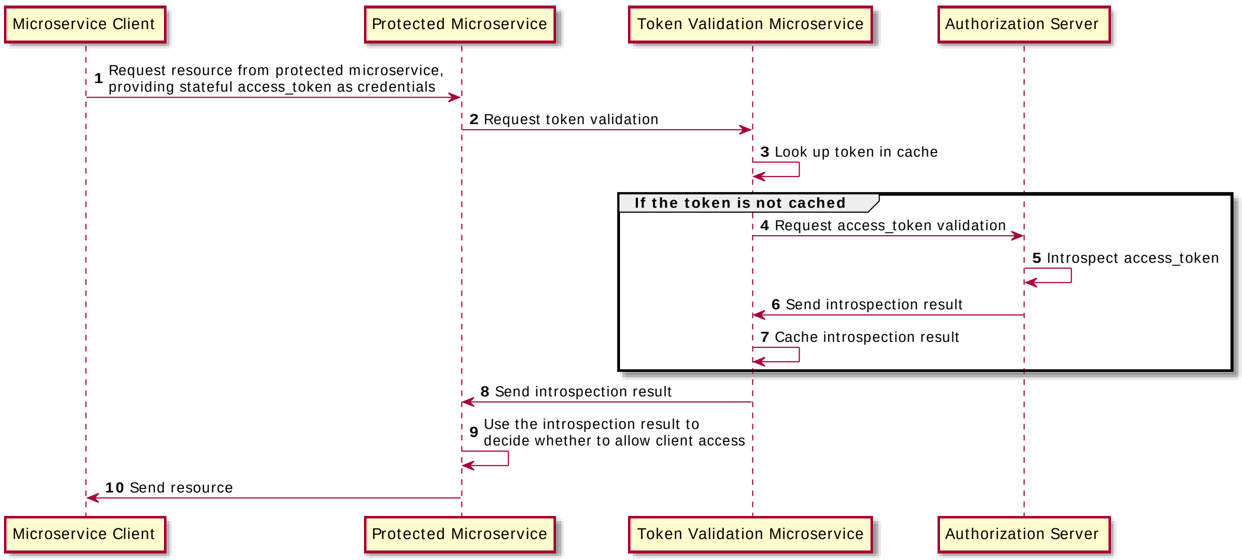
- In the terminal where TVMS is running, select CTRL-C to stop the service.

Chapter 4
Introspecting Stateful Access Tokens

This section describes how to set up TVMS to introspect stateful access tokens, using Identity Cloud or AM as the authorization server. For information about the architecture, see "Example Deployment of the ForgeRock Token Validation 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


Microservice Client
Protected Microservice
Token Validation Microservice
Authorization Server
1 Request resource from protected microservice, providing stateful access_token as credentials
2 Request token validation
3 Look up token in cache
If the token is not cached
4 Request access_token validation
5 Introspect access_token
6 Send introspection result
7 Cache introspection result
8 Send introspection result
9 Use the introspection result to decide whether to allow client access
10 Send resource

Introspect stateful access tokens from Identity Cloud

1. Set up Identity Cloud as an authorization server:
    a. Log in to Identity Cloud as an administrator.
    b. Make sure you are managing the alpha realm. If not, click the current realm at the top of the screen, and switch realm.

c. Configure an OAuth 2.0 Authorization Server:

   i. In the platform console, select Native Consoles > Access Management. The Access Management console is displayed.

   ii. Select Services, and update the OAuth2 Provider service with the following value:

      - Use Client-Side Access and Refresh Tokens: `off`

d. Add a web application to request OAuth 2.0 access tokens using client credentials for authentication:

   i. In the Identity Cloud console, click Applications > Add Application > Web, and add a web application with the following values:

      - Client ID: `microservice-client`
      - Client Secret: `password`

   ii. On the application page, add the following general settings:

      - Grant Types: `Client Credentials`
      - Scopes: `client-scope`

e. Add a web application authorized to examine (introspect) tokens:

   i. In the platform console, click Applications > Add Application > Web, and add a web application with the following values:

      - Client ID: `token-validation`
      - Client Secret: `password`

   ii. On the application page, add the following general settings:

      - Grant Types: `Authorization Code`
      - Scopes: `am-introspect-all-tokens`

2. Set up TVMS:

   a. Download and install TVMS as described in "*Downloading and Installing TVMS*".

   b. Prepare TVMS configuration and property files in `/path/to/microservices/token-validation/config`:

      i. Rename `config-template-stateful-cache.json` to `config.json`.

ii. In `config.json`, change the value of `openAmOAuth2Endpoint` to correspond to your authorization server. For example, use the following value in this example: `https://myTenant.forgeblocks.com/am/oauth2/alpha`

```
{
  "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"
              }
            }
          }
        }
      }
    }
  }
}
```

Notice the following features of the file:

- The `properties` section declares values used in the Identity Cloud configuration. If necessary, change these values to match your Identity Cloud 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 Identity Cloud 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*

iii. In `admin.json`, make sure that the ports for the TVMS are correct:

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

c. 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`.

d. Start TVMS, specifying the configuration directory as an argument. In the following example, TVMS 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
```

e. Make sure that TVMS is running, in the following ways:

- Ping TVMS 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. Test the setup:

a. Get an access token from Identity Cloud:

```
$ mytoken=$(curl -s \
--url https://myTenant.forgeblocks.com/am/oauth2/alpha/access_token \
--user "microservice-client:password" \
--data "grant_type=client_credentials" \
--data scope=client-scope | jq -r ".access_token")
```

b. View the access_token:

```
$ echo $mytoken
```

c. Call TVMS to introspect the access_token:

```
$ curl --data "token=${mytoken}" http://mstokval.example.com:9090/introspect
{
  "active": true,
  "scope": "client-scope",
  "realm": "/alpha",
  "client_id": "microservice-client",
  "user_id": "microservice-client",
  "username": "microservice-client",
  "token_type": "Bearer",
  "exp": 1661249008,
  "sub": "microservice-client",
  "subname": "microservice-client",
  "iss": "https://myTenant.forgeblocks.com/oauth2/alpha",
  "authGrantId": "T6F...mKw",
  "may_act": {
    "client_id": "serviceConfidentialClient",
    "sub": "(...)"
  },
  "auditTrackingId": "ef07c6b7-700f-4e72-add5-a9de0927831f-11139569"
}
```

Introspect stateful access tokens from Access Management

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*.

1. Set up AM as an authorization server:

   a. Configure an OAuth 2.0 Authorization Server:

      i. In the top level realm, select Services > Add a service > OAuth 2.0 Provider.

      ii. Accept all of the default values and select Create.

   b. Create an OAuth 2.0 client to request OAuth 2.0 access tokens, using client credentials for authentication:

      i. Select Applications > OAuth 2.0.

ii. Add a client with the following values:

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

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

- Grant Types: `Client Credentials`

c. Create an OAuth 2.0 Client authorized to examine (introspect) tokens:

i. In the top level realm, select Applications > OAuth 2.0.

ii. Add a client with the following values:

- Client ID: `token-validation`
- Client secret `password`
- Scope(s): `am-introspect-all-tokens`

2. Set up TVMS:

a. Download and install TVMS as described in "*Downloading and Installing TVMS*".

b. Prepare TVMS configuration and property files in `/path/to/microservices/token-validation/config`:

i. Rename `config-template-stateful-cache.json` to `config.json`.

ii. In `config.json`, change the value of `openAmOAuth2Endpoint` to correspond to your authorization server. For example, use the following value in this example: `http://openam.example.com:8088/openam/oauth2`

```
{
  "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"
              }
            }
          }
        }
      }
    }
  }
}
```

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*

iii. In `admin.json`, make sure that the ports for the TVMS are correct:

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

c. 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`.

d. Start TVMS, specifying the configuration directory as an argument. In the following example, TVMS 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
```

e. Make sure that TVMS is running, in the following ways:

- Ping TVMS 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. Test the setup:

a. Get an access_token from AM:

```
$ mytoken=$(curl \
--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)
```

b. View the access_token:

```
$ echo $mytoken
```

c. Call TVMS 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",
  "authGrantId": "Abc...1yz",
  "auditTrackingId": "12a...yz3
}
```

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.

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

   ```
   "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 TVMS, and then introspect an access_token.

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 TVMS 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.

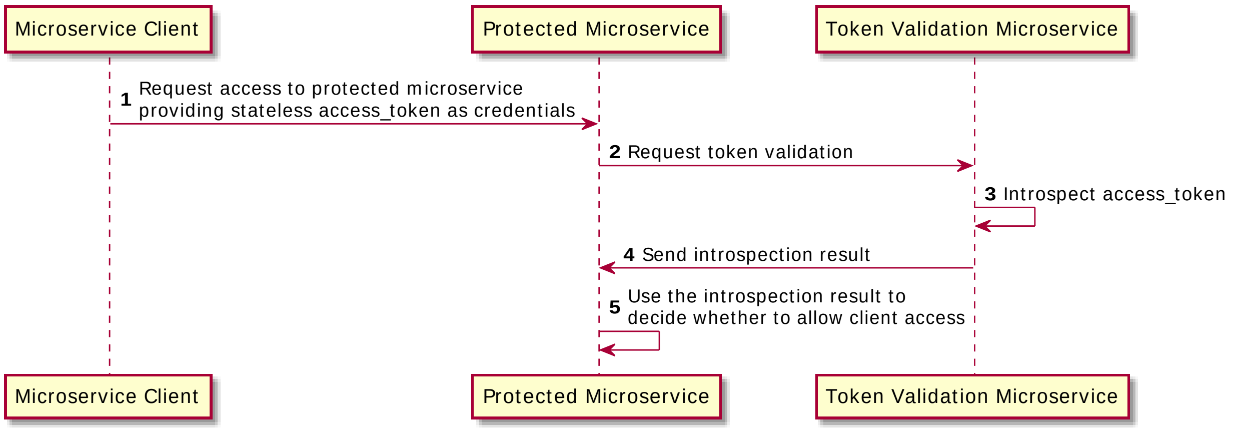
Chapter 5

Introspecting Stateless Access Tokens

This section describes how to set up TVMS to introspect stateless access tokens, using Identity Cloud or AM as the authorization server. For information about the architecture, see "Example Deployment of the ForgeRock Token Validation 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


Microservice Client
Protected Microservice
Token Validation Microservice
1 Request access to protected microservice providing stateless access_token as credentials
2 Request token validation
3 Introspect access_token
4 Send introspection result
5 Use the introspection result to decide whether to allow client access
Microservice Client
Protected Microservice
Token Validation Microservice

Introspect Stateless Access Tokens From Identity Cloud

1. Set up Identity Cloud as an authorization server:
    a. Log in to Identity Cloud as an administrator.
    b. Make sure you are managing the alpha realm. If not, click the current realm at the top of the screen, and switch realm.
    c. Configure an OAuth 2.0 Authorization Server:
        i. In the platform console, select Native Consoles > Access Management. The Access Management console is displayed.

ii. Select Services, and update the OAuth2 Provider service with the following value:

- Core > Use Client-Side Access and Refresh Tokens: `on`
- Advanced > OAuth2 Token Signing Algorithm : `RS256`

d. Add a web application to request OAuth 2.0 access tokens using client credentials for authentication:

i. In the Identity Cloud console, click Applications > Add Application > Web, and add a web application with the following values:

- Client ID: `microservice-client`
- Client Secret: `password`

ii. On the application page, add the following general settings:

- Grant Types: `Client Credentials`
- Scopes: `client-scope`

2. Set up TVMS:

a. Download and install TVMS as described in "*Downloading and Installing TVMS*".

b. Prepare TVMS configuration and property files in `/path/to/microservices/token-validation/config`:

i. Rename `config-template-stateless-cache.json` to `config.json`.

ii. In `config.json`, change the value of `openAmOAuth2Endpoint` to correspond to your authorization server. For example, use the following value in this example: `https://myTenant.forgeblocks.com:443/am/oauth2/alpha`

```
{
  "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 Identity Cloud configuration. If necessary, change these values to match your Identity Cloud 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 Identity Cloud 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*

iii. In `admin.json`, make sure that the ports for the TVMS are correct:

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

c. 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`.

d. Start TVMS, specifying the configuration directory as an argument. In the following example, TVMS 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
```

e. Make sure that TVMS is running, in the following ways:

- Ping TVMS 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. Test the setup:

a. Get an access token from Identity Cloud:

```
$ mytoken=$(curl -s \
--url https://myTenant.forgeblocks.com/am/oauth2/alpha/access_token \
--user "microservice-client:password" \
--data "grant_type=client_credentials" \
--data scope=client-scope | jq -r ".access_token")
```

b. View the access_token:

```
$ echo $mytoken
```

c. Call TVMS to introspect the access_token:

```
$ curl --data "token=${mytoken}" http://mstokval.example.com:9090/introspect
{
  "active": true,
  "sub": "microservice-client",
  "cts": "OAUTH2_STATELESS_GRANT",
  "auditTrackingId": "ef0...099",
  "subname": "microservice-client",
  "iss": "https://myTenant.forgeblocks.com/am/oauth2/alpha",
  "tokenName": "access_token",
  "token_type": "Bearer",
  "authGrantId": "hXz..._hM",
  "aud": "microservice-client",
  "nbf": 1661249395,
  "grant_type": "client_credentials",
  "scope": "client-scope",
  "auth_time": 1661249395,
  "realm": "/alpha",
  "exp": 1661252995,
  "iat": 1661249395,
  "expires_in": 3600,
  "jti": "hFL...PMY",
  "may_act": {
    "client_id": "serviceConfidentialClient",
    "sub": "(...)"
  }
}
```

Introspect stateless access tokens from Access Management

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*.

1. Set up AM as an authorization server:

   a. Configure an OAuth 2.0 Authorization Provider to provide signed stateless access tokens:

      i. In the top level realm, select Services > Add a service > OAuth 2.0 Provider.

      ii. Accept all of the default values and select Create.

      iii. On the Core tab, select the following option, and save the configuration:

         - Use Client-Based Access & Refresh Tokens: `on`

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

         - OAuth2 Token Signing Algorithm: `RS256`

   b. Create an OAuth 2.0 client to request OAuth 2.0 access tokens, using client credentials for authentication:

i. Select Applications > OAuth 2.0.

ii. Add a client with the following values:

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

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

- Grant Types: `Client Credentials`

2. Set up Token Validation Microservice Documentation:

a. Download and install TVMS as described in "*Downloading and Installing TVMS*".

b. Prepare the TVMS configuration and property files in `/path/to/microservices/token-validation/config`:

i. 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.

ii. In `admin.json`, make sure that the ports for the TVMS are correct:

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

c. Start TVMS, specifying the configuration directory as an argument. In the following example, TVMS 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
```

d. Make sure that TVMS is running, in the following ways:

- Ping TVMS 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. Test the setup:

a. Get an access_token from AM:

```
$ mytoken=$(curl \
--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)
```

b. View the access_token:

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

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

c. Call TVMS to introspect the access_token:

```
$ 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": 123....456,
  "grant_type": "client_credentials",
  "scope": client-scope,
  "auth_time": 155...523,
  "realm": "/",
  "exp": 155...123,
  "iat": 155...523,
  "expires_in": 3600,
  "jti": "cXQ...Lko"
}
```

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*" or "*Introspecting Stateless Access Tokens*".

2. Add the following script to the TVMS 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*.

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 TVMS to reload the configuration.

5. Test the setup

   a. Get an access_token from your OAuth 2.0 Authorization Server as described previously.

   b. View the access_token:

   ```
   $ echo $mytoken
   ```

   c. Call TVMS 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
   ...
   ```

   d. 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
   ...
   ```

Chapter 7
Monitoring TVMS

All ForgeRock products expose monitoring endpoints where metrics are exposed. When TVMS 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 TVMS, and protect the monitoring endpoints:

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

To Query the Monitoring Endpoints

1. Set up TVMS as described in "*Introspecting Stateful Access Tokens*" or "*Introspecting Stateless Access Tokens*".
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"
  },
...
```

To Protect the Monitoring Endpoints

1. Add the following script to the TVMS 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"
    }
  }
}]
```

3. Restart TVMS 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
   ```

Chapter 8 Querying and Disabling the Information Endpoint

When TVMS 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 TVMS to reload the configuration, and then query the version information endpoint again. Note this time that no information is returned.