Operation and configuration reference for Edge Microgateway

Edge Microgateway v. 2.3.x

Overview

This topic discusses how to manage and configure Edge Microgateway, including monitoring, logging, and debugging.

Making configuration changes

The configuration files you need to know about include:

- Default system configuration file
- Default config file for a newly initialized Edge Microgateway instance
- Dynamic configuration file for running instances

This section discusses these files and what you need to know about changing them. For details about configuration file settings, see Edge Microgateway configuration reference (#Edge%20Microgateway%20configuration%20reference).

Default system configuration file

When you install Edge Microgateway, a default system configuration file is placed here:

`[prefix]/lib/node_modules/edgemicro/config/default.yaml`

where **`[prefix]`** is the **`npm`** prefix directory. See Where is Edge Microgateway installed (https://docs.apigee.com/api-platform/microgateway/2.3.x/installing-edge-microgateway-v2.3.x.html#whereisedgemicrogatewayinstalled)
.

If you change the system config file, you must reinitialize, reconfigure, and restart Edge Microgateway:

1. Call **`edgemicro init`**
2. Call **`edgemicro configure [params]`**
3. Call **`edgemicro start [params]`**

Default config file for newly initialized Edge Microgateway instances

When you run **`edgemicro init`**, the system config file (described above), **`default.yaml`**, is placed in this directory: **`~/.edgemicro`**

If you change the config file in **`~/.edgemicro`**, you must reconfigure and restart Edge Microgateway:

1. **`edgemicro stop`**
2. **`edgemicro configure [params]`**
3. **`edgemicro start [params]`**

Dynamic configuration file for running instances

When you run **`edgemicro configure [params]`**, a dynamic configuration file is created in **`~/.edgemicro`**. The file is named according to this pattern: **`[org]-[env]-config.yaml`**, where **`org`** and **`env`** are your Apigee Edge organization and environment names. You can use this file to make configuration changes, and then reload them with zero-downtime. For example, if you add and configure a plugin, you can reload the configuration without incurring any downtime, as explained below.

If Edge Microgateway is running (zero-downtime option):

1. Reload the Edge Microgateway configuration:

   ```
   edgemicro reload -o [org] -e [env] -k [key] -s [secret]
   ```

   Where:

   - **`org`** is your Edge organization name (you must be an org administrator).
   - **`env`** is an environment in your org (such as test or prod).
   - **`key`** is the key returned previously by the configure command.
   - **`secret`** is the key returned previously by the configure command.

Example

```
edgemicro reload -o docs -e test -k 701e70ee718ce6dc188016b3c39177d64a
```

If Edge Microgateway is stopped:

1. Restart Edge Microgateway:

   ```
   edgemicro start -o [org] -e [env] -k [key] -s [secret]
   ```

   Where:

   - **org** is your Edge organization name (you must be an org administrator).
   - **env** is an environment in your org (such as test or prod).
   - **key** is the key returned previously by the configure command.
   - **secret** is the key returned previously by the configure command.

   Example

   ```
   edgemicro start -o docs -e test -k 701e70ee718ce6dc188016b3c39177d64a8
   ```

Here is an example config file. For details about configuration file settings, see Edge Microgateway configuration reference (#Edge%20Microgateway%20configuration%20reference).

```
edge_config:
  bootstrap: >-
    https://edgemicroservices-us-east-1.apigee.net/edgemicro/bootstrap/organizat
  jwt_public_key: 'https://docs-test.apigee.net/edgemicro-auth/publicKey'
  managementUri: 'https://api.enterprise.apigee.com'
  vaultName: microgateway
  authUri: 'https://%s-%s.apigee.net/edgemicro-auth'
  baseUri: >-
    https://edgemicroservices.apigee.net/edgemicro/%s/organization/%s/environmen
  bootstrapMessage: Please copy the following property to the edge micro agent c
  keySecretMessage: The following credentials are required to start edge micro
  products: 'https://docs-test.apigee.net/edgemicro-auth/products'
edgemicro:
  port: 8000
  max_connections: 1000
  max_connections_hard: 5000
  config_change_poll_interval: 600
```

```
  logging:
    level: error
    dir: /var/tmp
    stats_log_interval: 60
    rotate_interval: 24
  plugins:
    sequence:
      - oauth
headers:
  x-forwarded-for: true
  x-forwarded-host: true
  x-request-id: true
  x-response-time: true
  via: true
oauth:
  allowNoAuthorization: false
  allowInvalidAuthorization: false
  verify_api_key_url: 'https://docs-test.apigee.net/edgemicro-auth/verifyApiKey'
analytics:
  uri: >-
    https://edgemicroservices-us-east-1.apigee.net/edgemicro/axpublisher/organiz
```

Setting environment variables

The command-line interface commands that require values for your Edge organization and environment, and the key and secret needed for starting Edge Microgateway can be stored in these environment variables:

- **EDGEMICRO_ORG**
- **EDGEMICRO_ENV**
- **EDGEMICRO_KEY**
- **EDGEMICRO_SECRET**

Setting these variables is optional. If you set them, you do not have to specify their values when you use the Command-Line Interface (CLI) to configure and start Edge Microgateway.

Configuring SSL on the Edge Microgateway server

You can configure the Microgateway server to use SSL. For example, with SSL configured, you can call APIs through Edge Microgateway with the "https" protocol, like this:

```
https://localhost:8000/myapi
```

To configure SSL on the Microgateway server, follow these steps:

1. Generate or obtain an SSL certificate and key using the openssl (https://www.openssl.org) utility or whichever method you prefer.
2. Add the **`edgemicro:ssl`** attribute to the Edge Microgateway configuration file. For a complete list of options, see the table below. For details on modifying the Edge Microgateway config, see Making configuration changes (#Making%20configuration%20changes). For example:

```
edgemicro:
    ssl:
        key: <absolute path to the SSL key file>
        cert: <absolute path to the SSL cert file>
        passphrase: admin123 #option added in v2.2.2
        rejectUnauthorized: true #option added in v2.2.2
```

★ **Note:** The key specified in the referenced SSL key file **must not be encrypted**.

3. Restart Edge Microgateway. Follow the steps outlined in Making configuration changes (#Making%20configuration%20changes) depending on which configuration file you edited: the default file or the runtime config file.

Here's an example of the edgemicro section of the config file, with SSL configured:

```
edgemicro:
  port: 8000
  max_connections: 1000
  max_connections_hard: 5000
  logging:
    level: error
    dir: /var/tmp
    stats_log_interval: 60
    rotate_interval: 24
  plugins:
    sequence:
```

```
      - oauth
  ssl:
    key: /MyHome/SSL/em-ssl-keys/server.key
    cert: /MyHome/SSL/em-ssl-keys/server.crt
    passphrase: admin123 #option added in v2.2.2
    rejectUnauthorized: true #option added in v2.2.2
```

Here is a list of all of the supported server options:

Option	Description
key	Path to a **ca.key** file (in PEM format).
cert	Path to a **ca.cert** file (in PEM format).
pfx	Path to a **pfx** file containing the private key, certificate, and CA certs of the client in PFX format.
passphrase	A string containing the passphrase for the private key or PFX.
ca	Path to a file containing a list of trusted certificates in PEM format.
ciphers	A string describing the ciphers to use separated by a ":".
rejectUnauthorized	If true, the server certificate is verified against the list of supplied CAs. If verification fails, an error is returned.
secureProtocol	The SSL method to use. For example, SSLv3_method to force SSL to version 3.
servername	The server name for the SNI (Server Name Indication) TLS extension.

Using client SSL/TLS options

You can configure Edge Microgateway to be a TLS or SSL client when connecting to target endpoints. In the Microgateway configuration file, use the targets element to set SSL/TLS options.

This example provides settings that will be applied to all hosts:

```
targets:
   ssl:
     client:
       key: /Users/jdoe/nodecellar/twowayssl/ssl/client.key
```

```
      cert: /Users/jdoe/nodecellar/twowayssl/ssl/ca.crt
      passphrase: admin123
      rejectUnauthorized: true
```

In this example, the settings are applied only to the specified host:

```
targets:
   host: 'myserver.example.com'
   ssl:
     client:
       key: /Users/myname/twowayssl/ssl/client.key
       cert: /Users/myname/twowayssl/ssl/ca.crt
       passphrase: admin123
       rejectUnauthorized: true
```

Here is an example for TLS:

```
targets:
   host: 'myserver.example.com'
   tls:
     client:
       pfx: /Users/myname/twowayssl/ssl/client.pfx
       passphrase: admin123
       rejectUnauthorized: true
```

Here is a list of all of the supported client options:

Option	Description
`pfx`	Path to a **`pfx`** file containing the private key, certificate, and CA certs of the client in PFX format.
`key`	Path to a **`ca.key`** file (in PEM format).
`passphrase`	A string containing the passphrase for the private key or PFX.
`cert`	Path to a **`ca.cert`** file (in PEM format).
`ca`	Path to a file containing a list of trusted certificates in PEM format.
`ciphers`	A string describing the ciphers to use separated by a ":".
`rejectUnauthorized`	If true, the server certificate is verified against the list of supplied CAs. If verification fails, an error is returned.

secureProtocol	The SSL method to use. For example, SSLv3_method to force SSL to version 3.
servername	The server name for the SNI (Server Name Indication) TLS extension.

Using a custom auth service

By default, Edge Microgateway uses a proxy deployed on Apigee Edge for OAuth2 authentication. This proxy is deployed when you initially run **edgemicro configure**. By default, this proxy's URL is specified in the Edge Microgateway config file as follows:

```
authUri: https://myorg-myenv.apigee.net/edgemicro-auth
```

If you want to use your own custom service to handle authentication, change the **authUri** value in the config file to point to your service. For example, you may have a service that uses LDAP to verify identity.

Note: Important: If you change the auth service URI, you also need to configure these attributes in your Edge Microgateway config file:

- **edgeconfig:verify_api_key_url** - Set this URL to point to the API key endpoint on your custom auth service. In the default edgemicro-auth proxy, this URL is: **https://myorg-myenv.apigee.net/edgemicro-auth/publicKey.**
- **edgeconfig:products** - Set this URL to point to the products endpoint on your custom auth service. In the default edgemicro-auth proxy, this URL is: **https://myorg-myenv.apigee.net/edgemicro-auth/products.**

Note: Important: If you override the default auth proxy, the new proxy or service must return a response format that is identical to the default **edgemicro-auth** proxy.

Managing log files

Edge Microgateway logs information about each request and response. Log files provide useful information for debugging and troubleshooting.

Note: This section describes the built-in log utility for Edge Microgateway. If you need to use a third-party logging module, like Bunyan or Winston, you can write a custom plugin. For details, check out this blog: Tutorial: Adding a Logger Plugin to Apigee Edge Microgateway (https://apigee.com/about/blog/developer/tutorial-adding-logger-plugin-apigee-edge-microgateway). See also Develop custom plugins (https://docs.apigee.com/api-platform/microgateway/2.3.x/develop-custom-plugins-v2.3.x.html).

Where log files are stored

By default, log files are stored in `/var/tmp`.

How to change the default log file directory

The directory where log files are stored is specified in the Edge Microgateway configuration file. For details on making configuration changes, see Making configuration changes (#Making%20configuration%20changes).

```
edgemicro:
  home: ../gateway
  port: 8000
  max_connections: -1
  max_connections_hard: -1
  logging:
    level: info
    dir: /var/tmp
    stats_log_interval: 60
    rotate_interval: 24
```

Change the **dir** value to specify a different log file directory.

Send logs to the console

You can configure logging so that log information is sent to standard output instead of to a log file. Set the **to_console** flag to true as follows:

```
edgemicro:
  logging:
    to_console: true
```

With this setting, logs will be sent to standard out. Currently, you cannot send logs to both stdout and to a log file.

How to set the logging level

You can set these log levels: **info**, **warn**, and **error**. The info level is recommended. It logs all API requests and responses, and it is the default.

How to change log intervals

You can configure these intervals in the Edge Microgateway config file. For details on making configuration changes, see Making configuration changes (#Making%20configuration%20changes).

The configurable attributes are:

- **stats_log_interval**: (default: 60) Interval, in seconds, when the stats record is written to the API log file.
- **rotate_interval**: (default: 24) Interval, in hours, when log files are rotated. For example:

```
edgemicro:
  home: ../gateway
  port: 8000
  max_connections: -1
  max_connections_hard: -1
  logging:
    level: info
    dir: /var/tmp
    stats_log_interval: 60
    rotate_interval: 24
```

Note: Archived log files are not compressed. When the interval starts, a new log file is created with a new timestamp.

Good log file maintenance practices

As log file data accumulates over time, Apigee recommends that you adopt the following practices:

- Because log files can become quite large, be sure that the log file directory has sufficient space. See the following sections Where log files are stored (#Where%20log%20files%20are%20stored) and How to change the default log file directory (#How%20to%20change%20the%20default%20log%20file%20director).
- Either delete or move log files to a separate archive directory at least once a week.
- If your policy is to delete logs, you can use the CLI command **`edgemicro log -c`** to remove (clean) older logs.

Log file naming convention

Each Edge Microgateway instance produces three types of log files:

- **api** - Logs all requests and responses that flow through Edge Microgateway. API counters (stats) and errors are also logged to this file.
- **err** - Logs anything sent to stderr.
- **out** - Logs anything sent to stdout.

This is the naming convention:

```
edgemicro-<Host Name>-<Instance ID>-<Log Type>.log
```

For example:

```
edgemicro-mymachine-local-MTQzNTgNDMxODAyMQ-api.log
edgemicro-mymachine-local-MTQzNTg1NDMODAyMQ-err.log
edgemicro-mymachine-local-mtqzntgndmxodaymq-out.log
```

About log file contents

Added in: v2.3.3

By default, the logging service omits the JSON of downloaded proxies, products, and the JSON Web Token (JWT). If you wish to output these objects to the log files, set the **`DEBUG=*`** when you start Edge Microgateway. For example:

```
DEBUG=* edgemicro start -o docs -e test -k abc123 -s xyz456
```

Contents of the "api" log file

The "api" log file contains detailed information about the flow of requests and responses through Edge Microgateway. The "api" log files are named like this:

```
edgemicro-mymachine-local-MTQzNjIxOTk0NzY0Nw-api.log
```

For each request made to Edge Microgateway, four events are captured in the "api" log file:

- Incoming request from the client
- Outgoing request made to the target
- Incoming response from the target
- Outgoing response to the client

Each of these separate entries is represented in a shorthand notation to help make the log files more compact. Here are four sample entries representing each of the four events. In the log file, they look like this (the line numbers are only for reference in the doc, they don't appear in the log file).

```
(1) 1436403888651 info req m=GET, u=/, h=localhost:8000, r=::1:59715, i=0
(2) 1436403888665 info treq m=GET, u=/, h=127.0.0.18080, i=0
(3) 1436403888672 info tres s=200, d=7, i=0
(4) 1436403888676 info res s=200, d=11, i=0
```

Let's look at them one by one:

1. Sample of incoming request from client:

```
1436403888651 info req m=GET, u=/, h=localhost:8000, r=::1:59715, i=0
```

- **1436403888651** - Unix date stamp
- **info** - Depends on the context. Can be info, warn, or error, depending on the log level. Can be stats for a stats record, warn for warnings, or error for errors.
- **req** - Identifies the event. In this case, request from the client.
- **m** - The HTTP verb used in the request.
- **u** - The part of the URL following the basepath.

- **h** - The host and port number where Edge Microgateway is listening.
- **r** - The remote host and port where the client request originated.
- **i** - The request ID. All four event entries will share this ID. Each request is assigned a unique request ID. Correlating log records by request ID can give valuable insight into the target's latency.
- **d** - The duration in milliseconds since the request was received by Edge Microgateway. In the example above, the target's response for request 0 was received after 7 milliseconds (line 3), and the response was sent to the client after an additional 4 milliseconds (line 4). In other words, the total request latency was 11 milliseconds, out of which 7 milliseconds were taken by the target and 4 milliseconds by Edge Microgateway itself.

2. Sample of outgoing request made to the target:

```
1436403888665 info treq m=GET, u=/, h=127.0.0.1:8080, i=0
```

- **1436403888651** - Unix date stamp
- **info** - Depends on the context. Can be info, warn, or error, depending on the log level. Can be stats for a stats record, warn for warnings, or error for errors.
- **treq** - Identifies the event. In this case, target request.
- **m** - The HTTP verb used in the target request.
- **u** - The part of the URL following the basepath.
- **h** - The host and port number of the backend target.
- **i** - The ID of the log entry. All four event entries will share this ID.

3. Sample of incoming response from the target

```
1436403888672 info tres s=200, d=7, i=0
```

1436403888651 - Unix date stamp

- **info** - Depends on the context. Can be info, warn, or error, depending on the log level. Can be stats for a stats record, warn for warnings, or error for errors.
- **tres** - Identifies the event. In this case, target response.
- **s** - The HTTP response status.

- **d** - The duration in milliseconds. The time taken for the API call by the target.
- **i** - The ID of the log entry. All four event entries will share this ID.

4. Sample of outgoing response to the client

```
1436403888676 info res s=200, d=11, i=0
```

1436403888651 - Unix date stamp

- **info** - Depends on the context. Can be info, warn, or error, depending on the log level. Can be stats for a stats record, warn for warnings, or error for errors.
- **res** - Identifies the event. In this case, response to the client.
- **s** - The HTTP response status.
- **d** - The duration in milliseconds. This is the total time taken by the API call, including the time taken by the target API and the time take by Edge Microgateway itself.
- **i** - The ID of the log entry. All four event entries will share this ID.

Log file schedule

Log files are rotated on the interval specified by the **rotate_interval** configuration attribute (#log-intervals). Entries will continue being added to the same log file until the rotation interval expires. However, each time Edge Microgateway is restarted it receives a new UID and creates a new set of log files with this UID. See also Good log file maintenance practices (#Good%20log%20file%20maintenance%20practices).

Edge Microgateway configuration reference

Location of the configuration file

The configurations attributes described in this section are located in the Edge Microgateway configuration file. For details on making configuration changes, see Making configuration changes (#Making%20configuration%20changes).

Note: See the Setting up and configuring Edge Microgateway (https://docs.apigee.com/api-platform/microgateway/2.3.x/setting-and-configuring-edge-microgateway-v2.3.x.html) for more information about how to use this file.

edge_config attributes

These settings are used to configure interaction between the Edge Microgateway instance and Apigee Edge.

- **bootstrap**: (default: none) A URL that points to an Edge Microgateway-specific service running on Apigee Edge. Edge Microgateway uses this service to communicate with Apigee Edge. This URL is returned when you execute the command to generate the public/private key pair: **`edgemicro genkeys`**. See the Setting up and configuring Edge Microgateway (https://docs.apigee.com/api-platform/microgateway/2.3.x/setting-and-configuring-edge-microgateway-v2.3.x.html) for details.
- **jwt_public_key**: (default: none) A URL that points to the Edge Microgateway proxy that is deployed on Apigee Edge. This proxy serves as an authentication endpoint for issuing signed access tokens to clients. This URL is returned when you execute the command to deploy the proxy: **edgemicro configure**. See the Setting up and configuring Edge Microgateway (https://docs.apigee.com/api-platform/microgateway/2.3.x/setting-and-configuring-edge-microgateway-v2.3.x.html) for details.

edgemicro attributes

These settings configure the Edge Microgateway process.

- **port**: (default: 8000) The port number on which the Edge Microgateway process listens.
- **max_connections**: (default: -1) Specifies the maximum number of simultaneous incoming connections that Edge Microgateway can receive. If this number is exceeded, the following status is returned:

```
res.statusCode = 429; // Too many requests
```

- **max_connections_hard**: (default: -1) The maximum number of simultaneous requests that Edge Microgateway can receive before shutting down the connection. This setting is intended to thwart denial of service attacks. Typically, set it to a number larger than max_connections.
- **logging**:
    - **level**: (default: error)
        - **info** - Logs all requests and responses that flow through an Edge Microgateway instance.
        - **warn** - Logs warning messages only.
        - **error** - Logs error messages only.
    - **dir**: (default: /var/tmp) The directory where log files are stored.
    - **stats_log_interval**: (default: 60) Interval, in seconds, when the stats record is written to the api log file.
    - **rotate_interval**: (default: 24) Interval, in hours, when log files are rotated.
- **plugins**: Plugins add functionality to Edge Microgateway. For details about developing plugins, see Develop custom plugins (https://docs.apigee.com/api-platform/microgateway/2.3.x/develop-custom-plugins-v2.3.x.html).
    - **dir**: A relative path from ./gateway directory to the ./plugins directory, or an absolute path.
    - **sequence**: A list of plugin modules to add to your Edge Microgateway instance. The modules will execute in the order they are specified here.
- **debug:** Adds remote debugging to the Edge Microgateway process.
    - **port**: The port number to listen on. For example, set your IDE debugger to listen on this port.
    - **args**: Arguments to the debug process. For example: `args --nolazy`
- **config_change_poll_interval:** (default: 600 seconds) Edge Microgateway loads a new configuration periodically and executes a reload if anything changed. The polling picks up any changes made on Edge (changes to products, microgateway-aware proxies, etc) as well as changes made to the local config file.
- **disable_config_poll_interval:** (default: false) Set to **true** to **turn off** automatic change polling.

headers attributes

These settings configure how certain HTTP headers are treated.

- **x-forwarded-for**: (default: true) Set to false to prevent x-forwarded-for headers to be passed to the target.
- **x-forwarded-host**: (default: true) Set to false to prevent x-forwarded-host headers to be passed to the target.
- **x-request-id**: (default: true) Set to false to prevent x-request-id headers to be passed to the target.
- **x-response-time**: (default: true) Set to false to prevent x-response-time headers to be passed to the target.
- **via**: (default: true) Set to false to prevent via headers to be passed to the target.

oauth attributes

These settings configure how client authentication is enforced by Edge Microgateway.

Note: You can find the steps for obtaining and using access tokens in the Setting up and configuring Edge Microgateway (https://docs.apigee.com/api-platform/microgateway/2.3.x/setting-and-configuring-edge-microgateway-v2.3.x.html)

.

- **allowNoAuthorization**: (default: false) If set to true, API calls are allowed to pass through Edge Microgateway without any Authorization header at all. Set this to false to require an Authorization header (default).
- **allowInvalidAuthorization**: (default: false) If set to true, API calls are allowed to pass if the token passed in the Authorization header is invalid or expired. Set this to false to require valid tokens (default).
- **authorization-header**: (default: Authorization: Bearer) The header used to send the access token to Edge Microgateway. You may wish to change the default in cases where the target needs to use the Authorization header for some other purpose.
- **api-key-header**: (default: x-api-key) The name of the header or query parameter used to pass an API key to Edge Microgateway. See also Using an API key (#usinganapikey).

- **keepAuthHeader**: (default: false) If set to true, the Authorization header sent in the request is passed on to the target (it is preserved).

Plugin-specific attributes

See Using plugins for details on configurable attributes for each plugin.

Filtering proxies

You can filter which microgateway-aware proxies an Edge Microgateway instance will process. When Edge Microgateway starts, it downloads all of the microgateway-aware proxies in the organization it's associated with. Use the following configuration to limit which proxies the microgateway will process. For example, this configuration limits the proxies the microgateway will process to three: **`edgemicro_proxy-1`**, **`edgemicro_proxy-2`**, and **`edgemicro_proxy-3`**:

```
proxies:
  - edgemicro_proxy-1
  - edgemicro_proxy-2
  - edgemicro_proxy-3
```

Masking analytics data

The following configuration prevents request path information from showing up in Edge analytics. Add the following to the microgateway configuration to mask the request URI and/or request path. Note that the URI consists of the hostname and path parts of the request.

```
analytics:
  mask_request_uri: 'string_to_mask'
  mask_request_path: 'string_to_mask'
```

Debugging and Troubleshooting

Connecting to a debugger

You can run Edge Microgateway with a debugger, such as node-inspector (https://www.npmjs.com/package/node-inspector). This is useful for troubleshooting and debugging custom plugins.

1. Restart Edge Microgateway in debug mode. To do this, add `DEBUG=*` to the beginning of the start command. For example:

   ```
   DEBUG=* edgemicro start -o myorg -e test -k
   db4e9e8a95aa7fabfdeacbb1169d0a8cbe42bec19c6b98129e02 -s
   6e56af7c1b26dfe93dae78a735c8afc9796b077d105ae5618ce7ed
   ```

2. Start your debugger and set it to listen on the port number for the debugging process.
3. You can now step through the Edge Microgateway code, set breakpoints, watch expressions, and so on.

You can specify standard Node.js flags related to debug mode. For example, `--nolazy` helps with debugging asynchronous code.

Checking log files

If you're having problems, be sure to examine the log files for execution details and error information. For details, see Managing log files (#Managing%20log%20files).

Using API Key security

API keys provide a simple mechanism for authenticating clients making requests to Edge Microgateway. You can obtain an API key by copying the Consumer Key (also called Client ID) value from an Apigee Edge product that includes the Edge Microgateway authentication proxy.

Caching of keys

API keys are exchanged for bearer tokens, which are cached. You can disable caching by setting the `Cache-Control: no-cache` header on incoming requests to Edge Microgateway.

Required workaround if you are on Apigee Edge Private Cloud version 15.07

To use API Key security on Edge Private Cloud 15.07, you must implement the workaround described here. The workaround requires you to change one line in a Node.js file in the **edgemicro-auth** proxy and restart Edge Microgateway.

Note: This workaround is required for all versions of Edge Microgateway if you are on Private Cloud 15.07. This workaround is not needed if you are on Private Cloud 16.01 or later.

Implement the workaround in the Edge UI

This procedure requires you to re-run the **edgemicro start** command when you are finished.

1. In the Edge UI, open the edgemicro-auth proxy in the proxy editor.
2. Select the Develop tab.
3. In the Navigator, under Scripts, open the JavaScript file called **verify-api-key.js**.
4. Go to line 109:

   ```
   api_product_list: apigeeToken.app && apigeeToken.app.apiproducts ? apigeeToken.app.apiproducts : []
   ```
5. Replace the part after the "api_product_list:" with a "hard-coded" array of the Product names that are associated with any API keys you wish to use. So, for example, if you want to use a key from a developer app that has "Product-1" and "Product-2" in it, then you code the line like this:

   ```
   api_product_list: ["Product-1", "Product-2"]
   ```
6. Click Save.
7. Execute the **edgemicro start** command.

Implement in the local Edge Microgateway code base

This procedure requires you to re-run the edgemicro **configure command**, followed by **edgemicro start**.

1. Open the file `<microgateway-root-dir>/edge/auth/api/controllers/verify-api-key.js`, where `<microgateway-root-dir>` is the directory where Edge Microgateway was installed when you ran the `npm install` command.

2. Go to line 109:

   ```
   api_product_list: apigeeToken.app && apigeeToken.app.apiproducts ?
   apigeeToken.app.apiproducts : []
   ```

3. Replace the part after the "api_product_list:" with a "hard-coded" array of the Product names that are associated with any API keys you wish to use. So, for example, if you want to use a key from a developer app that has "Product-1" and "Product-2" in it, then you code the line like this:

   ```
   api_product_list: ["Product-1", "Product-2"]
   ```

4. Save the file.
5. Execute the **edgemicro configure** command. (Or, **edgemicro private configure** if you are on Edge Private Cloud.) This command re-deploys the edgemicro-auth proxy.
6. Execute the **edgemicro start** command.

Using OAuth2 token security

For details on using an OAuth token with proxy requests, see Secure Edge Microgateway (https://docs.apigee.com/api-platform/microgateway/2.3.x/setting-and-configuring-edge-microgateway-v2.3.x.html#part4secureedgemicrogateway)

.

Using an API key

For details on using API keys with proxy requests, see Secure Edge Microgateway (https://docs.apigee.com/api-platform/microgateway/2.3.x/setting-and-configuring-edge-microgateway-v2.3.x.html#part4secureedgemicrogateway)

.

Note: You can also pass the API key in a query parameter. For example, where `[key]` is the API key, a string of letters and numbers:

```
curl "http://localhost:8000/hello?"x-api-key=[key]"
```

Configuring the API key name

By default, **`x-api-key`** is the name used for the API key header or query parameter. You can change this default in the configuration file, as explained in Making configuration changes (#Making%20configuration%20changes). For example, to change the name to **apiKey**:

```
oauth:
 allowNoAuthorization: false
 allowInvalidAuthorization: false
 api-key-header: apiKey
```
