

API Integration Guide

Use Case 2

Business to Gouvernement (M2M)



Document history

Version	Date	Author	Notes
1.0	23/03/2021	DSI	Initial release
2.0	03/06/2021	DSI	Updated Section 6 on Error Management

Table of Contents

1. Introduction.....	4
1.1. Scope.....	4
1.2. Contents.....	4
1.3. Audience.....	4
2. Overview.....	5
2.1. Definitions.....	5
2.2. Prerequisites.....	5
3. The API Portal.....	6
4. Organizations management.....	7
4.1. Registering an organization with the API Gateway.....	7
4.2. Tying a technical representative with an organization.....	8
5. Initial enrollment of your client application and access management.....	9
5.1. Registering a client application with the API Gateway.....	9
5.2. Receiving your client ID and Client Secret and token request URL.....	10
5.3. Oauth flow : Client credential flow (requesting and receiving the access token).....	10
5.4. Accessing the API.....	14
6. Error Management.....	15
6.1. Error codes.....	15
6.2. API Gateway Error Message Format.....	16
6.3. Error handling policy in the API Gateway.....	16

1. Introduction

1.1. Scope

This document details the security measures regarding **accessing APIs made available by CTIE-hosted applications**, specifically in the context of consuming these APIs from B2G applications running on enterprise infrastructure.

1.2. Contents

A first overview of the scenario and of the architecture is presented in chapter 2.

The necessary steps for registering your organization are outlined in chapter 4.

Chapter 5 details the initial client application enrollment and tokens exchange, then chapter 5.4 delves into the technical details of your application accessing the API.

1.3. Audience

This document is intended for **project managers, architects and developers involved in the development of enterprise applications** running on external enterprise infrastructure and accessing APIs made available by CTIE-hosted applications.

2. Overview

2.1. Definitions

2.1.1. API Consumers

An API Consumer is typically an organization that wants to access one or more APIs made available by an API Developer.

An API Consumer must always request access to each given API it wishes to use.

2.1.2. API Developers

An API Developer is typically an organization that wants to make one or more APIs available to external organizations.

An API Developer must always authorize API Consumers to access its APIs.

2.1.3. Client applications

A client application is an application run by an API Consumer and accessing one or more APIs made available by one or more API Developers.

2.2. Prerequisites

2.2.1. Registering an organization as an API Consumer

An organization wishing to become an API Consumer must register itself with the CTIE API Gateway, and delegate one or more agents allowed to make further requests on behalf of the organization.

This is detailed in 4 Organizations management, page 7.

2.2.2. Requesting and obtaining access to an API

Once an organization is registered as an API Consumer, it must define one or more client applications. Then for each application, it must request access to one or more APIs that it wishes the application to use.

For each request, the API Developer for the corresponding API must explicitly grant access.

Once the access is granted, the API Consumer will receive a Client ID and a Client Secret, that can then be used to obtain an Access token eventually allowing the Client application to access the APIs it needs.

This is detailed in 5 Initial enrollment of your client application and access management, page 9.

3. The API Portal

As of now, there is no API portal, and most processes are carried on using PDF forms as outlined in subsequent chapters. This document will be updated regularly as the API portal is put together and the processes are adapted to be based on online forms.

4. Organizations management

4.1. Registering an organization with the API Gateway

4.1.1. Manual procedure

Registering an organization with the API Gateway currently involves a manual procedure.

The organization representative must provide a properly filled-in form (ref. IAM-CTIE-042) that will contain the following required information :

- Name of the organization ;
- Type of the organization (SA, Sàrl, ASBL, etc) ;
- Luxembourg Business Registers (RCS) number ;
- Luxembourg National Register (matricule) number ;
- Address ;
- General phone number of the organization (for IT issues) ;
- General email of the organization (for IT issues).

This form must be electronically signed (using e.g. a LuxTrust certificate) by an official representative as described in the organization status and documented in the Luxembourg Business Registers.

When filled in and signed, this form must be provided to CTIE, along with a copy of the ID documents of the official representative.

Accuracy of the provided information will be checked with the Luxembourg Business Registers and the Luxembourg National Register.

An entry in the API Gateway will then be created for the organization.

4.1.2. About organization types

Numerous organization types are considered (Luxembourg-based companies, institutions publiques, centres des recherche, communes, GIE européens, etc...).

4.1.3. Organization unique ID

Each organization accessing a CTIE-hosted API will be identified by a unique ID currently the Matricule number for physical or moral persons is used.

4.2. Tying a technical representative with an organization

At least one technical representative must be delegated for each organization. The technical representative will introduce further API access requests on behalf of the organization, including the required technical details.

Registering a technical representative with an organization linked to the API Gateway currently involves a manual procedure.

The organization representative must provide a properly filled-in form that will contain the following required information. Practically, this is done using the same form used under 4.1 Registering an organization with the API Gateway, page 7 (form ref. IAM-CTIE-042).

Required details when registering a technical representative :

- First name and last name ;
- Luxembourg National Register number (matricule) ;
- Job title of the user ;
- Mobile Phone number of the user (for password) ;
- Email of the user.

5. Initial enrollment of your client application and access management

5.1. Registering a client application with the API Gateway

A registered technical representative (as per 4.2 Tying a technical representative with an organization) with a registered organization (as per 4.1 Registering an organization with the API Gateway) must provide a properly filled-in form (ref. IAM-CTIE-043).

The following information must be provided in the form :

- Requesting organization name and Luxembourg Business Registers number (allowing to tie the request with a previously-defined organization) ;
- Application or project name as used in your organization ;
- Public IP address(es) that will be used for the API consumption ;
- List of requested APIs and their respective owners withing the Government administrations (entity making the API available).

This form must be electronically signed (using e.g. a LuxTrust certificate) by the technical representative.

When filled in and signed, this form must be provided to CTIE.

The request as defined in the form will be approved by a business representative on the CTIE / Government administrations side.

5.2. Receiving your client ID and Client Secret and token request URL

After the form submitted as per 5.1 Registering a client application with the API Gateway has been processed, you will receive the following information through a secure channel (OTX link) :

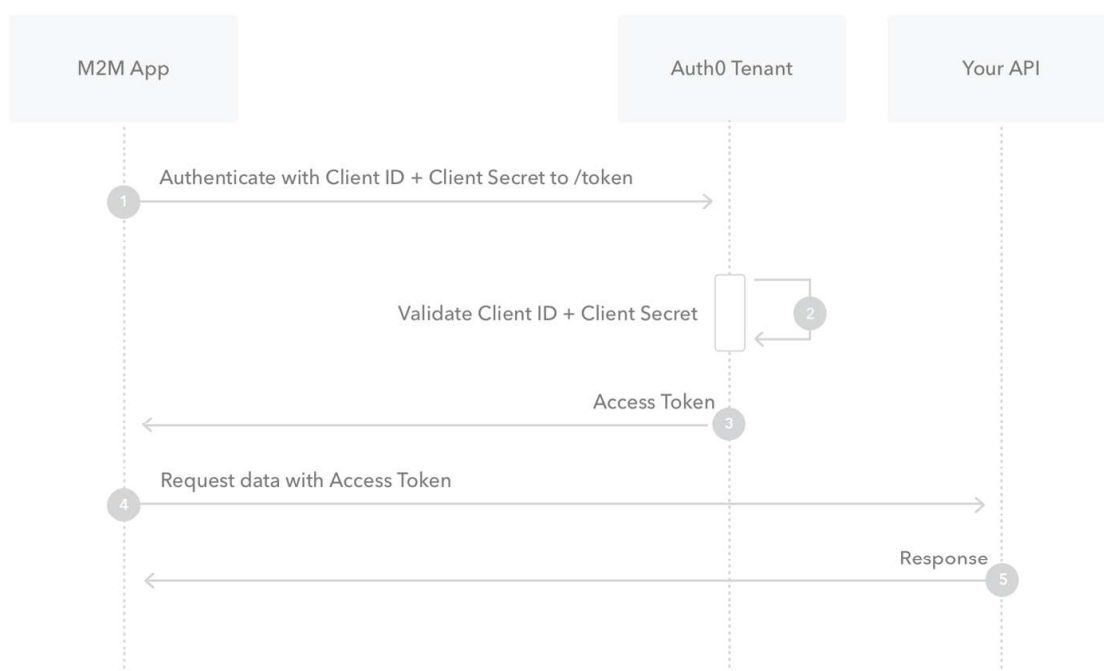
- Client ID
- Client Secret

It is the responsibility of the client application owner / organization to maintain the confidentiality of the client ID / client secret. Should these be lost, the tech rep should contact CTIE to initiate a new client ID / client secret.

These are unique to each organization's client application.

5.3. Oauth flow : Client credential flow (requesting and receiving the access token)

The following schema illustrates the process :



5.3.1. Requesting your access token

You need to use this information received as per 5.2 Receiving your client ID and Client Secret and token request URL, in order to receive a new access token.

The token service to receive the Access Token is available under the following URL:

- Url in Stage : <https://oauth.api-stg.etat.lu/api/oauth/token>
- Url in Prod : <https://oauth.api.etat.lu/api/oauth/token>

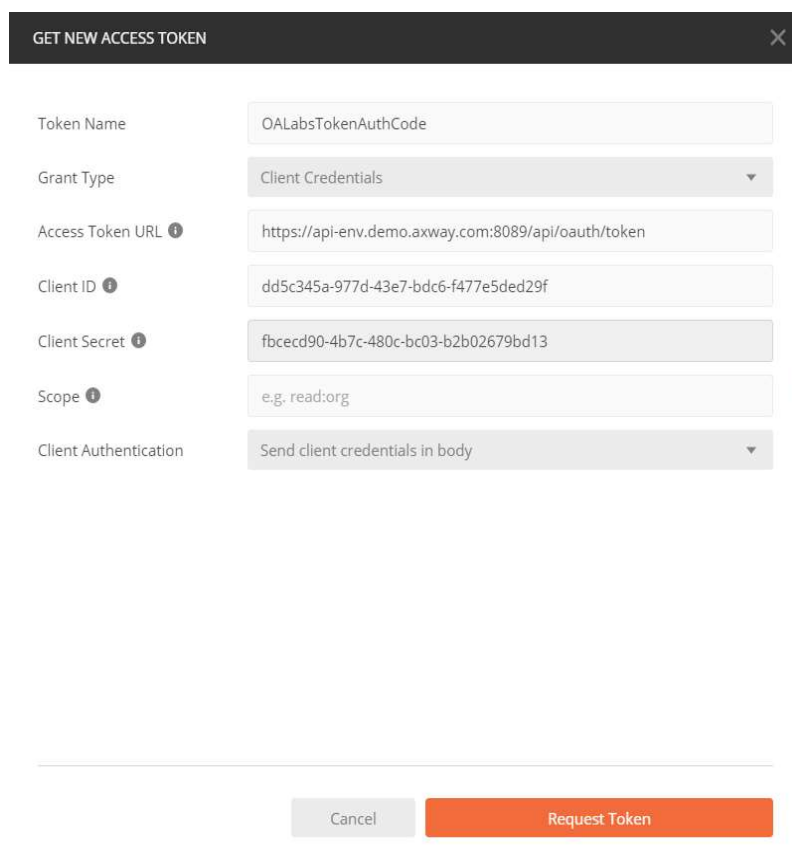
5.3.1.1. Request details

A **post** with a request body containing the following information:

Attribute	Example	Details
grant_type	"client_credentials"	The grant type must be client credentials
client_id	"d1dd7055-1a73-4c19-8c27-6a6ef0b07dda"	The client id is provided by the CTIE
client_secret	"321d5c6e-1717-467a-9a36-4c2678464b97"	The client secret is provided by the CTIE

1. grant_type: "client_credentials"
2. client_id: "d1dd7055-1a73-4c19-8c27-6a6ef0b07dda"
3. client_secret: "321d5c6e-1717-467a-9a36-4c2678464b97"

Example of a request in postman:



The screenshot shows the 'GET NEW ACCESS TOKEN' dialog box in Postman. It contains the following fields and values:

- Token Name: OALabsTokenAuthCode
- Grant Type: Client Credentials (dropdown)
- Access Token URL: https://api-env.demo.axway.com:8089/api/oauth/token
- Client ID: dd5c345a-977d-43e7-bdc6-f477e5ded29f
- Client Secret: fbcecd90-4b7c-480c-bc03-b2b02679bd13
- Scope: e.g. read:org
- Client Authentication: Send client credentials in body (dropdown)

At the bottom, there are two buttons: 'Cancel' and 'Request Token'.

5.3.2. Receiving the access token

After requesting the access token as per 5.3.1 Requesting your access token, you will receive your token.

The token is valid for 3600 seconds and must be renewed when expired.

5.3.2.1. Typical token details

Attribute	Example	Details
token_type	"Bearer"	Type of the Token Bearer
expires_in	3600	Expiration time in seconds, tokens are valid for one hour
scope	"resource.WRITE resource.READ"	
iss	"CTIE"	Issuer of the Token

```
{
  "access_token": "CxGJPWpGt8QLaELDwybw5E10Oc4xPAthn4brYXYEar4J7WziaFtbVK",
  "token_type": "Bearer",
  "expires_in": 3600,
  "scope": "resource.WRITE resource.READ",
  "iss": "CTIE"
}
```

5.3.2.2. In case of invalid client id or client secret

Note : In case of an invalid client or secret, one receives a “400 bad request” reply, with the following message :

```
{
  "error": "invalid_client",
  "error_description": "Client authentication failed (e.g. unknown client, no client authentication included, or unsupported authentication method). The authorization server MAY return an HTTP 401 (Unauthorized) status code to indicate which HTTP authentication schemes are supported. "
}
```

5.4. Accessing the API

The previously-issued token can now be used to access the API's of the application.

The call needs to contain the Token in the Authorization Header.

Header: Authorization

Authorization Bearer [AT]

```
Authorization Bearer CxGJPWpGt8QLaELDwybw5E10Oc4xPAthn4brYXYEar4J7WziaFtbVK
```

If the Access Token is not valid anymore, the called api will answer with an **HTTP 401** unauthorized.

The client application / developer is responsible for managing the access token and it's validity. The client application needs to be able to dynamically request a new access token once the old one is expired or a 401 is received. This is not the responsibility of the APIGW.

6. Error Management

6.1. Error codes

The following table explains which Error codes which the client application may receive from the API Gateway.

HTTP Status	Error Code	Description
401 Unauthenticated / Unauthorized	7004	Your token is invalid or expired.
	7011	Please check your sending the correct Bearer Token.
	7012	
	7005	Unable to authenticate API key. Please check your sending the correct Bearer Token.
	7006	Your token is invalid. Please check the “issued at” (iat) claim.
	7007	Your token is invalid. Please check the “audience” (aud) claim.
	7008	Your token is invalid. Please check the “issuer” (iss) claim.
	7010	Your token is invalid. Unable to retrieve the x-id-token.
405	6002	Bad Method The method is not allowed.
404	6003	Requested path in URI was not found or the responding server provided the error. Please contact CTIE for further troubleshooting.
400	Various Error Codes	Bad Request. The API definition is not respected. Please verify the request parameters. Please contact CTIE for further troubleshooting.
503	6008	The Application is currently in maintenance mode Please retry later and if the problem persists please contact CTIE for further troubleshooting.
500	Various Error Codes	Internal Server Error Please retry later and if the problem persists please contact CTIE for further troubleshooting.

If the problem persist please contact our Helpdesk (Helpdesk@ctie.etat.lu) with the detailed error message.

6.2. API Gateway Error Message Format

In the event of an error, as described previously in Section 6.1, the API Gateway will return a response to the client application with a JSON error string with the following schema:

```
{
  "id": "Id-1806a5600817ac40a51c4769",
  "code": "ERR:7017",
  "message": "Internal Server Error"
}
```

6.3. Error handling policy in the API Gateway

Depending on the HTTP error status code that the API Gateway received from the backend web service, certain interception and transformation of data may take place. The following table explains the standard policy defined in the CTIE API Gateway.

HTTP Status	Description
400	The APIGW will allow backend error messages with these status codes to transit completely to the client without transformation.
403	
404	
405	
406	
408	
409	
415	
401	The APIGW will intercept this and provide a generic message in the JSON string of the body of the response: <pre>{ "id": "Id-1806a5600817ac40a51c4769", "code": "ERR:7017", "message": "Internal Server Error" }</pre>
500	The APIGW will intercept this and remove all the data in the body of the server response, EXCEPT for the field "id", which should be an internal error reference for the backend application, which will be sent to the client. This will allow the client to share this with the application owner in order to facilitate troubleshooting, and will prevent any sensitive technical information from being leaked.