

External AES Tutorial

Table of Content

[Download Prerequisites](#)

[1 - Backend and Proxy API](#)

[2 - External OAuth Provider](#)

[3 - Apply the External OAuth2 Policy](#)

[4 - Request API Access](#)

[5 - Test protected API](#)

Download Prerequisites

- [API Gateway 2.0.0 RC](#) (temporary until 2.0 GA is available [here](#))
- [External OAuth2 Provider Template](#) (temporary until the template is available [here](#))
- [External OAuth2 Custom Policy](#) (temporary until it is available as an out of the box policy)
- [Sample Proxy](#)
- [keystore.jks](#)
- [Anypoint Studio](#) with API Gateway 2.0.0 runtime (use [this repository](#))

1 - Backend and Proxy API

1. Download the latest API Gateway 2.0 version
2. Configure it to use the client_id and client_secret of your organization or of one of its Business Groups
 - a. Login to [Anypoint platform](#) and get the client_id and client_secret from your organization
 - b. Edit `./config/wrapper.conf` to add the following two additional parameters as shown below :

```
wrapper.java.additional.7=-Danypoint.platform.client_id=<your org client ID>
wrapper.java.additional.8=-Danypoint.platform.client_secret=<your org client-secret>
```

NOTE: *the numbers in these parameters (wrapper.java.additional.<N>) must run sequentially in order starting with 1 on the top parameter in the file*

```
#wrapper.java.additional.<n>=-Dmule.clusterNodeId=1
#wrapper.java.additional.<n>=-Dmule.clusterSize=2
#####

#####
# Anypoint Platform 2.0 Settings
#####
# The following option is mandatory and identifies your Mule instance against
# the Anypoint Platform.
#
#wrapper.java.additional.7=-Danypoint.platform.client_id=
#wrapper.java.additional.8=-Danypoint.platform.client_secret=
#
# For the client to use a proxy when communicating back to the Anypoint Platform, you
# need to configure the following properties
#
# wrapper.java.additional.<n>=-Danypoint.platform.proxy_host=XXXXXXXXX
# wrapper.java.additional.<n>=-Danypoint.platform.proxy_port=XXXXXXXXX
# wrapper.java.additional.<n>=-Danypoint.platform.proxy_username=XXXXXXXXX
# wrapper.java.additional.<n>=-Danypoint.platform.proxy_password=XXXXXXXXX
#
# On-Prem Configuration
#
#wrapper.java.additional.<n>=-Danypoint.platform.on_prem=false
```

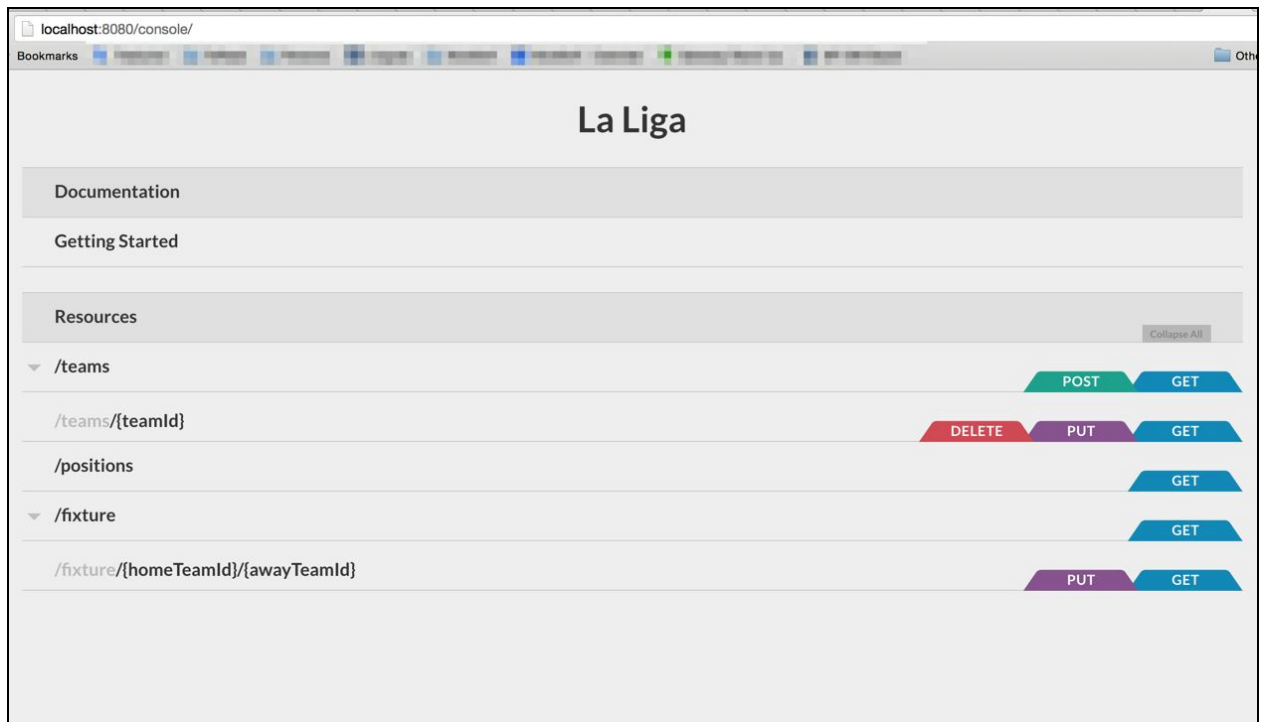
3. Deploy an app with an API - This is the API that should be protected by the OAuth policy
 - a. Start the API Gateway
 - b. Copy `./examples/apps/leagues-rest` (from to the gateway home) to the `/apps` folder within your gateway installation.

NOTE: *You will need to copy the entire leagues-rest directory*

- c. In an internet browser, open the Leagues App by requesting the <http://localhost:8080/api/teams> resource.

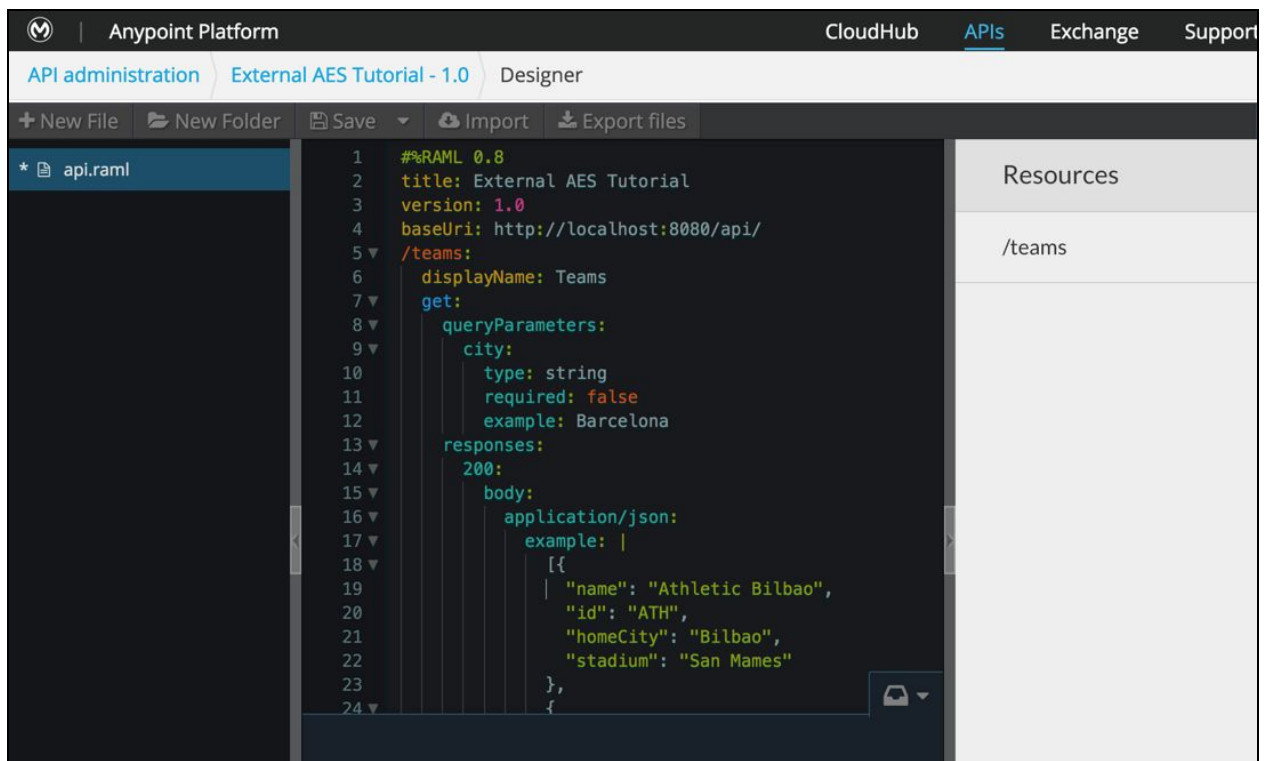


- d. Again in the browser, open the RAML console at <http://localhost:8080/console/>. From here you can make calls to the leagues API via a simple UI



4. Login to [Anypoint platform](#)
5. Register a new API in your Anypoint platform account, through this platform you will add a proxy in front of the backend API. For this tutorial, make sure to use the name "External AES Tutorial" and version "1.0".

You can use this [RAML file](#) as a reference.



- a. Save it and return to the API Version Details Page
6. Create an HTTP proxy by clicking on “Configure endpoint” within the version details page for that API and filling in the required information as follows - Make sure you use HTTPS

Configure endpoint

Use a **basic endpoint** if you are implementing your API in the MuleSoft API Gateway. Otherwise, choose an **endpoint with a proxy** for an existing API.

☐ Basic endpoint
☒ Endpoint with a proxy

Endpoint settings

Type

RAML

Implementation URI

http://localhost:8080/api/ Get from RAML

Proxy settings

☐ Configure proxy for CloudHub?

Scheme	Port ?	Path
HTTPS	8081	/leagues

⚠ Any change to the port setting will only be effective for proxies running in an API Gateway v1.3.x or lower. Proxies generated for an API Gateway v2.0.0 or higher will run on port 8081 by default. To customize the proxy port in an API Gateway v2.0.0 or higher, please refer to the [API Gateway admin guide](#).

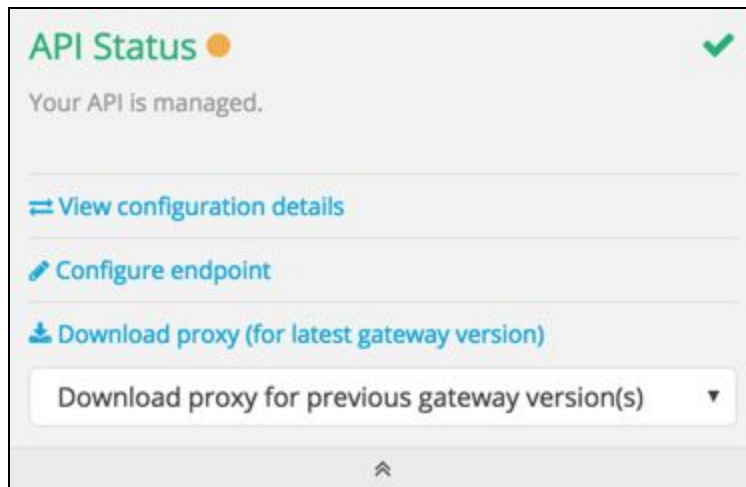
Cancel
Save & deploy
Save

a. Save

7. Use the proxy and keystore provided by this tutorial.

- Copy the file `keystore.jks` from the prerequisites section to the `/conf` directory within the gateway home.
- Copy the zip file from the prerequisites section to the `/apps` directory within the gateway home.

NOTE: *In the step above, a pre-built proxy is used (as opposed to the one you can download from the platform) since the production version of the API platform does not yet have support for generating proxies that run with the upcoming API Gateway 2.0 version. Future updates of the Anypoint Platform will allow you to download proxies built with this runtime as shown as follows:*



8. The proxy application should be working at <https://localhost:8081/leagues/teams>

2 - External OAuth Provider

1. Import the template file (downloadable in the pre-requisite section of this tutorial) into Studio and make sure it is using API Gateway 2.0.0 Server Runtime
2. Copy the file `keystore.jks` to `src/main/resources`
3. Set the following properties in `src/main/resources/mule.dev.properties`
 - a. For single authentication

```
1 # Properties to be used on the development environment
2 key.store.password=mule123
3 key.store.key.password=mule123
4 key.store.path=keystore.jks
5 admin.name=name
6 admin.password=password
7 validate.endpoint.path=validate
8 authorization.endpoint.path=authorize
9 access.token.endpoint.path=access_token
10
```

4. For LDAP authentication



```
1 # Properties to be used on the development environment
2 key.store.password=mule123
3 key.store.key.password=mule123
4 key.store.path=keystore.jks
5
6 ldap.userDn=cn=Manager,dc=my-domain,dc=com
7 ldap.password=root
8 ldap.url=ldap://localhost:389/dc=my-domain,dc=com
9 ldap.user.search.filter.1=ou=people,dc=my-domain,dc=com
10 ldap.user.search.filter.2=(uid={0})
11
12 validate.endpoint.path=validate
13 authorization.endpoint.path=authorize
14 access.token.endpoint.path=access_token
15 scopes=
16 supported.grant.types=AUTHORIZATION_CODE RESOURCE_OWNER_PASSWORD_CREDENTIALS CLIENT_CREDENTIALS IMPLICIT
17 |
```

Write down the three endpoint paths. They will be used in future steps.

5. Open the project's `config.xml` file in Studio
6. Go to the Global Elements tab, under the canvas
7. Edit the OAuth Provider module

Global Element Properties

OAuth provider module

Global OAuth provider module configuration information.

General **Notes**

Generic

Name: external-oauth2-provider

General

Access Token Endpoint Path: \${access.token.endpoint.path}

Host:

Provider Name: Ping API

Authorization Ttl Seconds: 600

Port: 9999

Client Store Reference: single-user-client-store

Authorization Code Store Reference:

Token Store Reference:

Authorization Endpoint Path: \${authorization.endpoint.path}

Login Page:

Scopes: READ WRITE

Token Ttl Seconds: 86400

Connector Configuration:

Resource Owner Security Provider Reference: single-user-security-provider

Client Security Provider Reference: single-user-client-security-provider

Supported Grant Types: AUTHORIZATION_CODE RESOURCE_OWNER_PASSWORD

Rate Limiter Reference:

☒ Enable Refresh Token

clients reference:

Cancel OK

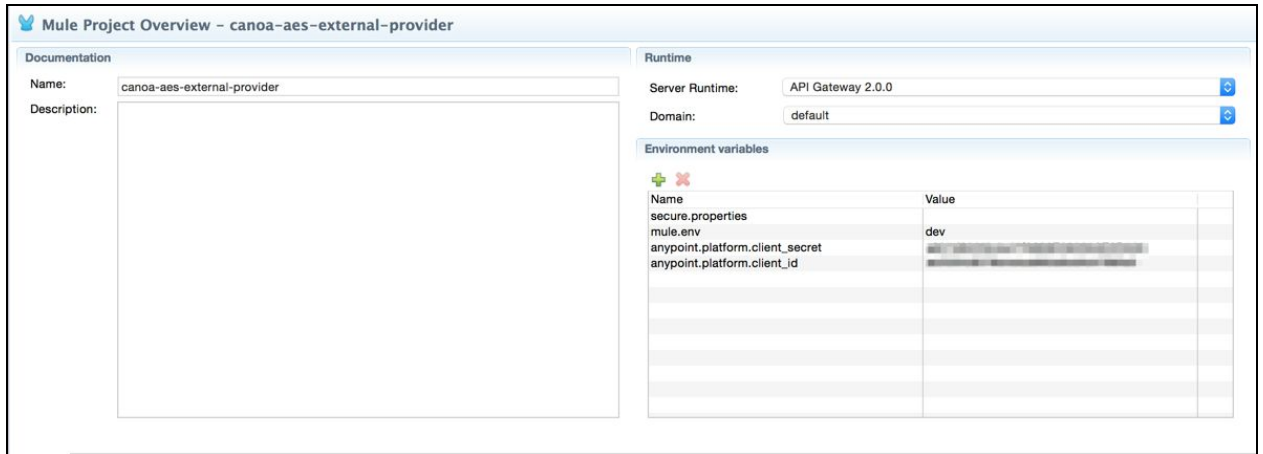
If you want to test the API through the console, Scopes must be empty (defaults are "READ WRITE").

- a. "Configuration XML" leaving `defaultScopes=""` and `scopes=""`
 - b. `userValidation.xml`: within `validateTokenFlow`, `scopes=""` in `oauth2-provider:validate` element.
8. Configure the parameters in Studio's Gateway 2.0 runtime
- a. From the project directory, open the file `mule-project.xml`

- b. Add the `client_id` and `client_secret` from your organization to the runtime Environment variables

`anypoint.platform.client_id=<your org client secret>`

`anypoint.platform.client_secret=<your org client ID>`



9. Run External OAuth2 Provider as Mule Application. A "DEPLOYED" status message for the service provider application should be shown in the console.

3 - Apply the External OAuth2 Policy

1. [Add](#) the External OAuth2 Provider custom policy to the policies in your Anypoint Platform account (you can download the necessary files for this in the prerequisites section)
2. Open the API version page of the API created in the section 1 of this document.
3. Open the policies tab.
4. You'll see the new custom policy "External AES OAuth2 Access Token Enforcement".
5. Add the RAML snippet to the API's RAML in Designer. The updated RAML should look like [this one](#).
6. Replace `./apps/External_AES_Tutorial-v1.0/classes/assets/api.raml` (within the gateway home directory) with the one with the changes made in step 5.
7. Restart the Gateway.
8. If everything went correctly, you'll be able to select "OAuth 2.0" from a dropdown menu in the [application console](#).
9. Apply AES external custom policy providing the validation URL (in this case <https://localhost:8082/validate>).

If you are going to use the console, no scopes must be provided and CORS policy must be applied as well.

Apply "External OAuth2 Access Token Enforcement" policy

Enforces use of an OAuth 2.0 access token issued through an OAuth 2.0 external provider.

This policy will require updates to the RAML definition in order to function. You can obtain the RAML snippet and learn more [here](#).

Scopes

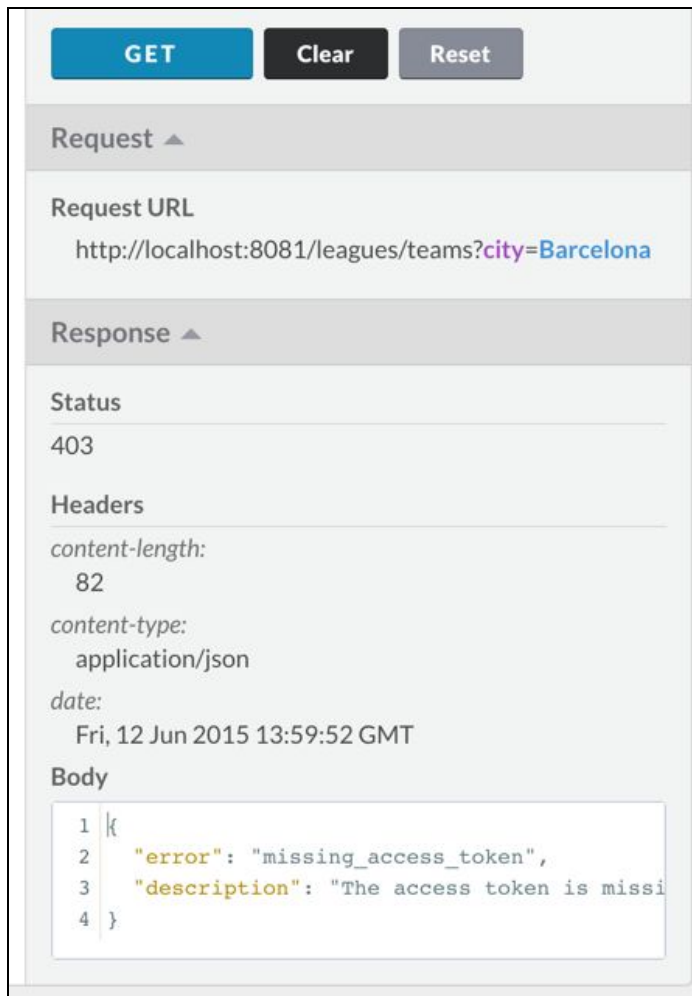
A space-separated list of supported scopes

Access Token validation endpoint url *

The url of the Access Token validation endpoint of the External OAuth2 Provider.

CancelApply

10. Open the [API console](#) and try the teams resource. This time a 403 status code will be returned:



4 - Request API Access

1. Through the API version page, create a New Portal for your API
2. Click on Live Portal to view it
3. Click on the Request API Access button
4. Register a new application to the API (for this tutorial, leave Redirect URI empty) and click on Request API Access
5. Go to Developer Portal → My applications and get the client id and secret of the created app.

5 - Test protected API

1. Open the [API console](#)

2. Try the `teams` resource
 - a. Security Scheme → OAuth2
 - b. Authorization Grant → Implicit
 - c. Client ID → the one from the granted app in section 4

The screenshot shows a REST client interface with a dark header bar containing a "CLOSE" button and a "GET" tab. Below the header is a "Try it" section with a "Try it" button and a close icon. The main configuration area is divided into sections: "AUTHENTICATION" with "Security Scheme" set to "OAuth 2.0" and "Authorization Grant" set to "Implicit"; "Client ID" with a red asterisk and a blurred text field; "Scopes" with a plus icon; "HEADERS" with a plus icon; "QUERY PARAMETERS" with a plus icon; and "city" with a text field containing "Barcelona". At the bottom are three buttons: "GET" (blue), "Clear" (dark grey), and "Reset" (light grey).

- d. Click on GET and put the username and password used in the service provider configuration in Section 2

\$CLIENT_NAME | Login

https://localhost:8082/authorize?scope=&client_id=484fef6573194070bc6ea1f3a7e56e3a&redirect_uri=https%3A%2F%2Flocalh...

Ping API

You can use your Ping API account to sign-in to \$CLIENT_NAME.

Authorize \$CLIENT_NAME to use your account?

Username

name

Password

Login and Authorize

Cancel

- e. Click Login and Authorize. You should see a 200 status code with the response

Request ▲

Request URL
https://localhost:8081/leagues/teams?city=Barcelona

Response ▲

Status
200

Headers
content-type:
application/json
date:
Fri, 12 Jun 2015 22:18:55 +0000
transfer-encoding:
chunked

Body

```
1 |{
2 |  "teams": [
3 |    {
4 |      "id": "BAR",
5 |      "name": "Barcelona",
6 |      "homeCity": "Barcelona",
7 |      "stadium": "Camp Nou"
8 |    },
9 |    {
10 |      "id": "ESP",
11 |      "name": "Espanyol",
12 |      "homeCity": "Barcelona",
13 |      "stadium": "Cornella-El Prat"
14 |    }
15 |  ]
16 | }
```

That's all folks.

