Lab 1

*Discover and Reuse Order Fulfillment API from Exchange 2.0*
Overview

In this lab we will create a new Mule application in Anypoint Studio from the Order Fulfillment API RAML Definition stored in the Anypoint Exchange Design Center. This will be the implementation of our REST API.

Step 1: Create a new Mule Project and import RAML Definition

We are going to use our API definition to create an implementation of our Order Fulfillment API. To do this we need to import the RAML Specification from the Design Center

1. If it’s not already opened, start Anypoint Studio from the desktop icon and select the same workspace you’ve been working (example: C:\workspaces\myworkspace).
2. From Anypoint Studio’s menu, select File > New > Mule Project to create a New Mule project. A Window will pop-up to define the details of this new application.
3. Give the project the name api-order-fulfillment
4. Select the Mule Server 4.1.4 EE.
5. Under API Implementation settings, check Specify API definition file location or URL and choose Design Center… by clicking in the (...) button on the right of the Location textbox.

6. Once you clicked there, it should take you to the Browse Design Center for APIs window, if it asks you to login, use your Anypoint Platform account.
7. Select the correct Business Group.
8. This is the same you did in the last module. You will see a list with all the APIs definitions that were created. Select Order Fulfillment API and press OK
If you don’t see the **Order Fulfillment API**, press **Load more projects**. This link only appears when there are too many projects on the API Designer.

9. Press Finish. This will create a skeleton project that implements your API.

10. The import will create a new Anypoint Studio application with a preconfigured implementation based on your API metadata like the following

Let’s explore what was automatically generated.

At the top of the diagram you will see a flow called **order_fulfillment-api-main** along with a placeholder flow of the Process API namely: **post:/order_fulfillment:application/json:order-fulfillment-api-config**. This Process API is a reflection of the POST method defined in the RAML sourced from Exchange 2.0. *(There will also be an api-product-console flow, as well as a global exception handling block.)*

Typically, there will be flows that look like this get:/resource, post:/resource, put:/resource, etc. Note that the name of the flow is very important for the APIkit router to be able to route the request to the appropriate flow - you don’t want to change these.
When APIkit detects example data in the response of a method in the RAML it inserts a Set Payload Transformer into the flow which returns the static response specified by the example data reference.

The static response returned by the auto-generated Set Payload Transformer allows you to test the stub immediately after generation. These flows can be enhanced to provide more advanced mock services as well as evolve into the full API implementation.

api-product-main

This is the main flow. It exposes an HTTP service and processes the requests using the APIKit Router.

The HTTP request will be converted to a mule message, and redirected to the requested flow by the APIKit router.

If you take a look at the HTTP configuration, you will see that it's listening for requests on http://localhost:8081/api
This is the skeleton flow that we’ll implement in the next step. By quickly laying down this skeleton flow or stub, developers and consumers alike can quickly test this flow running as a Mule application. The static response returned by the auto-generated allows one to test the stub immediately after generation. Obviously these flows can be enhanced to provide more advanced mock services as well as evolve into the full API implementation. The evolution of API asset is a natural byproduct of typical Agile development practices adopted by many Mulesoft customers.

Step 2: Run the API

1. To test the API, let’s run it within Studio first. Right click api-order-fulfillment in the Package Explorer view.

2. Select Run As > Mule Application (Feel free to skip this if you’ve done this in the previous lab).

3. The application will start running, and the console will show the Mule Runtime logs. Wait until the console shows the application has completed deployment.

4. Once the application is deployed, Anypoint Studio will open a tab with a link to the API Console
where we can test the application (to maximize the console just double-click the APIKit Console tab)

5. Click on it and a new page will open on your custom browser.

6. Click the **POST** tab on the left to test the **order_fulfillment** method.
7. Press **Try it** button. You will see a Console to test the API.

8. Go to the **Body** tab and check the request body.

9. Click the SEND button.

10. You should see the sample the response from the customer creation.
11. Go to the console tab and press the red button to stop the Mule runtime server.

In the next lab we will update the flows to connect to the database.

**Summary**

In this lab, we

- Import the RAML Definition directly into Anypoint Studio
- Auto-generated a skeleton project to implement the API
- Run the skeleton to check that is correct.

Go Further:

- See the link APIkit doc for more information.

Congratulations! You have completed Lab 1.

Please proceed to Lab 2

Take me to the TOP