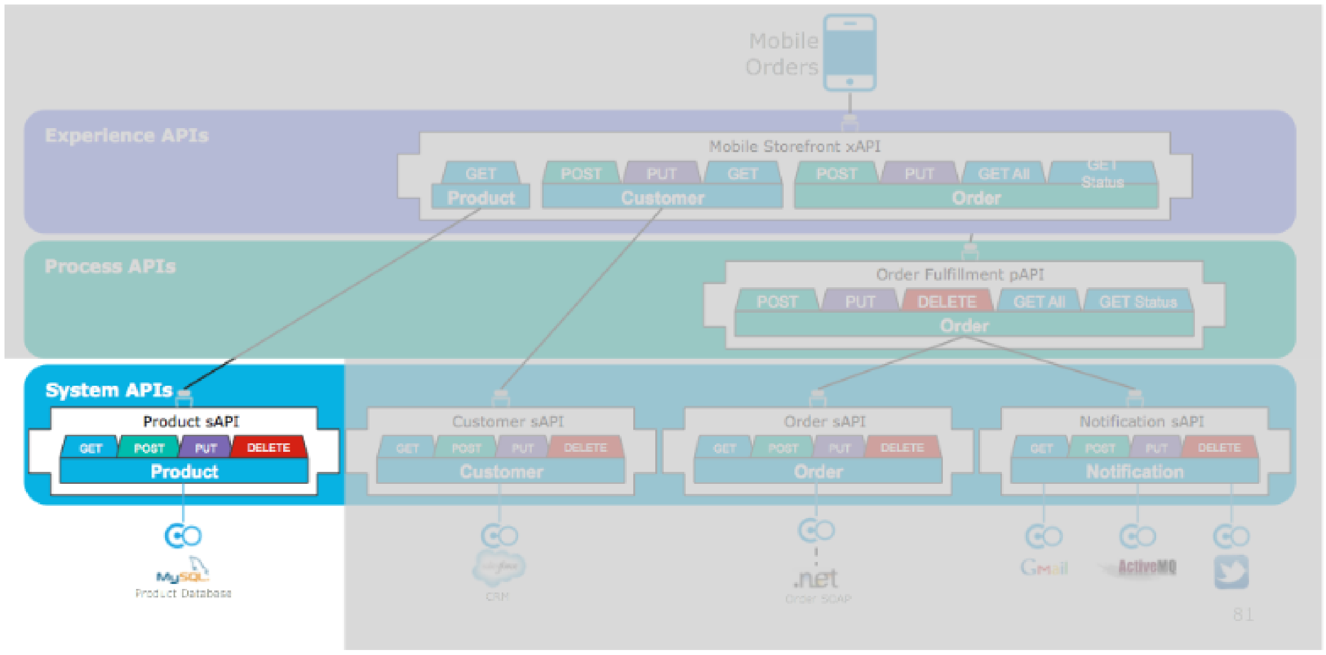
Lab 1
Implement the Product API in Studio

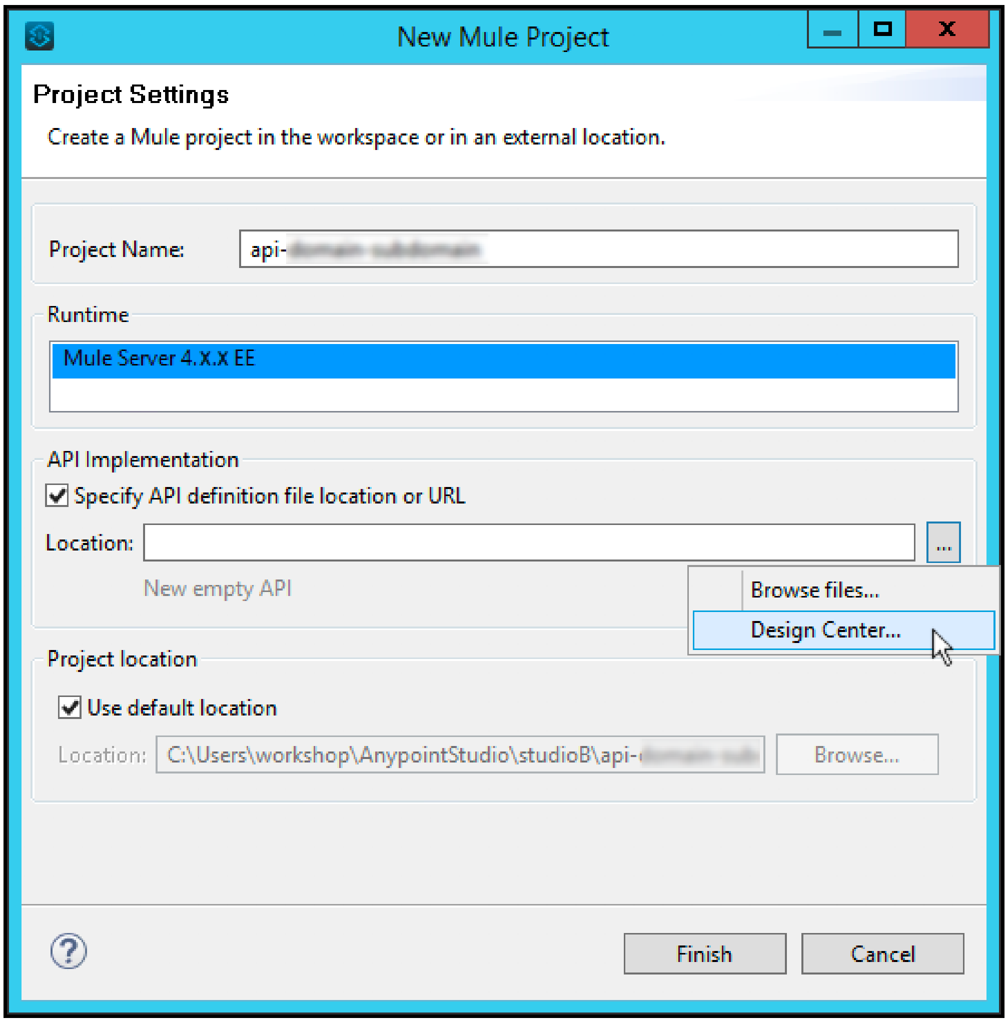
Overview

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



Steps

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-products**
4. Select the **Mule Server 4.1.1 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.


New Mule Project

Project Settings

Create a Mule project in the workspace or in an external location.

Project Name: api-

Runtime

Mule Server 4.X.X EE

API Implementation

[✓] Specify API definition file location or URL

Location:

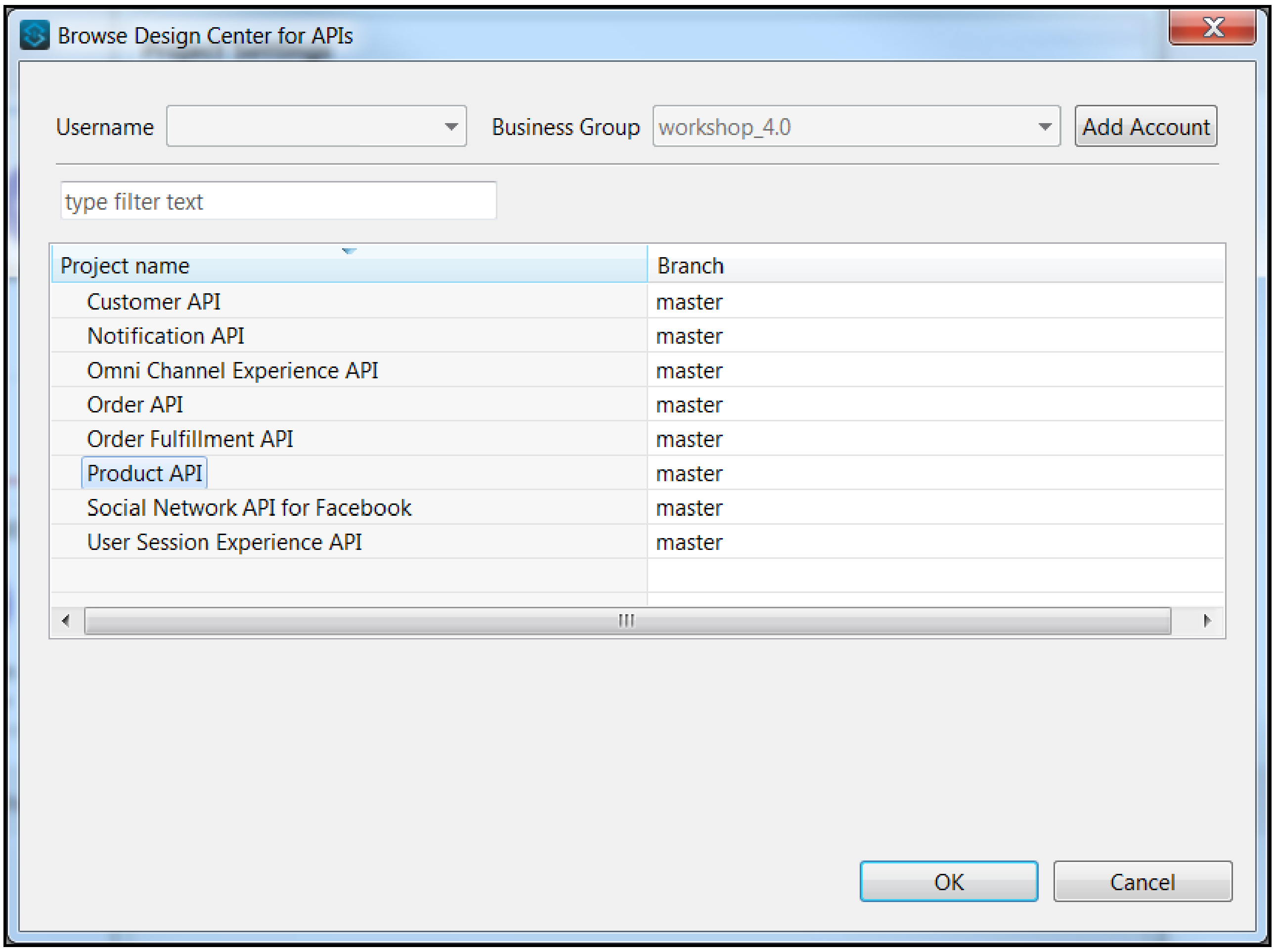
New empty API

Project location

[✓] Use default location

Location: C:\Users\workshop\AnypointStudio\studioB\api-

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 corresponding business group.
8. Select the Business Group from the dropdown, and then select the **Product API** and click **OK**.


Browse Design Center for APIs

Username [ ] Business Group [workshop_4.0] Add Account

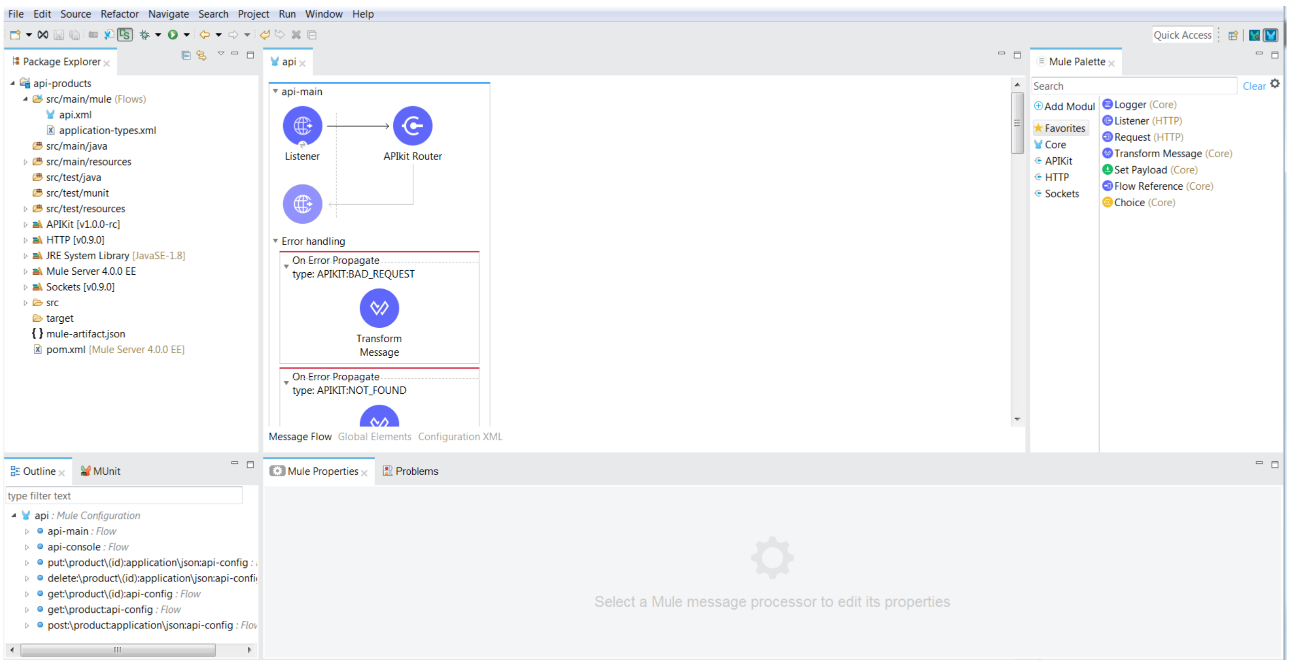
type filter text

Project name	Branch
Customer API	master
Notification API	master
Omni Channel Experience API	master
Order API	master
Order Fulfillment API	master
Product API	master
Social Network API for Facebook	master
User Session Experience API	master

OK Cancel

WARNING If you don't see the **Product API**, press **Load more projects**. This link only appears when there are too many projects on the API Designer.

9. Now click **Finish**. This will create a skeleton project that implements your API.



- api-products
  - src/main/mule (Flows)
    - api.xml
    - application-types.xml
  - src/main/java
  - src/main/resources
  - src/test/java
  - src/test/munit
  - src/test/resources
  - APIKit [v1.0.0-rc]
  - HTTP [v0.9.0]
  - JRE System Library [JavaSE-1.8]
  - Mule Server 4.0.0 EE
  - Sockets [v0.9.0]
  - src
  - target
  - mule-artifact.json
  - pom.xml [Mule Server 4.0.0 EE]

type filter text

- api : *Mule Configuration*
  - api-main : *Flow*
  - api-console : *Flow*
  - put:\product\(id):application\json:api-config :
  - delete:\product\(id):application\json:api-confi
  - get:\product\(id):api-config : *Flow*
  - get:\product:api-config : *Flow*
  - post:\product:application\json:api-config : *Flov*

CAUTION	If you do not see the src/main/resources/api folder with the api.raml files and corresponding JSON files (as shown in the screenshot), the project generation was not successful. Try deleting/re-generating your project or alert your instructor to troubleshoot.

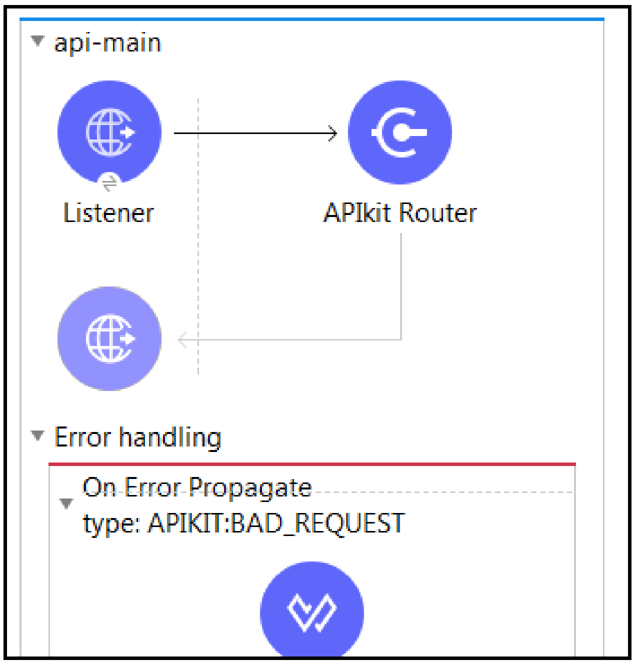
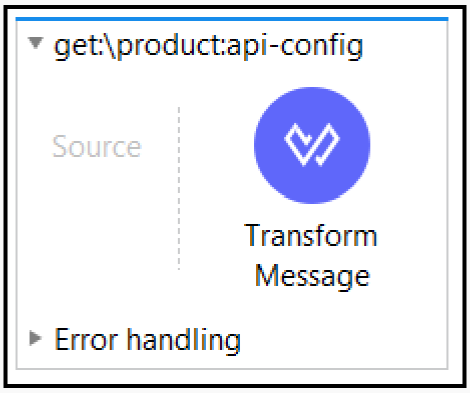
10. **Let’s explore what was automatically generated.**

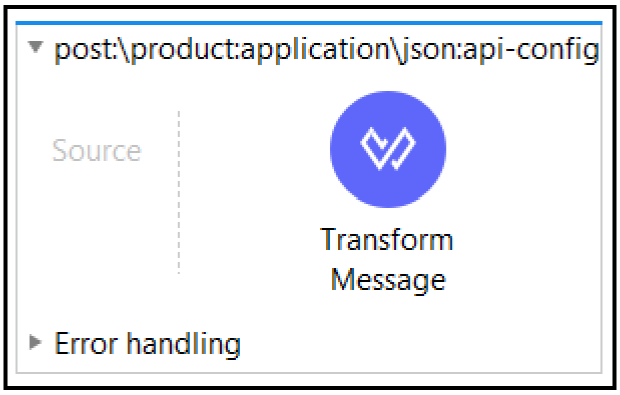
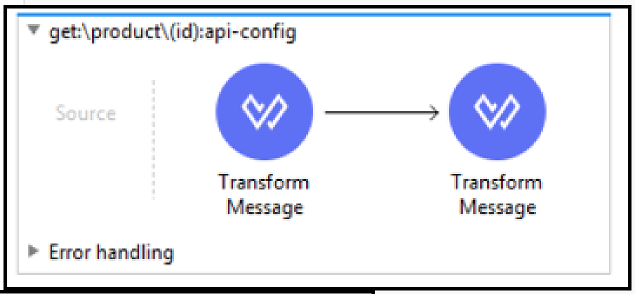
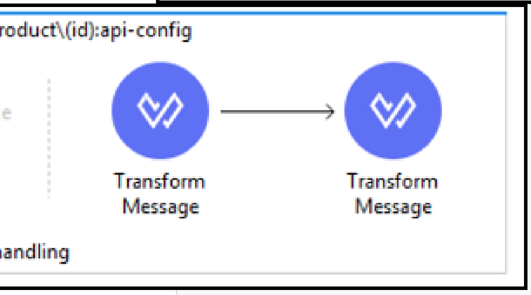
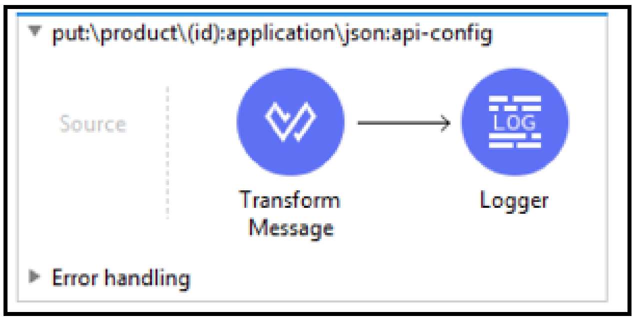
    At the top of the diagram you will see a flow called **api-main** with five sub-flows, one for each method defined in the RAML. (*There will also be an api-console flow, as well as an exception handling block in these flows.*)

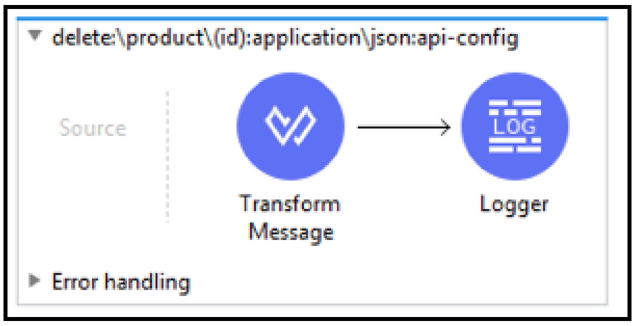
    The flows below are defined by your API in the RAML file. Typically, there will be flows that look like this get:\resource, post:\resource, put:\resource, etc. Note that the name of the flow is very important for the APIkit router to be able to route the request to the appropriate flow - you should not change these.

    When APIkit detects example data in the response of a method in the RAML it inserts a **Transform Message** component into the flow which returns the static response specified by the example data reference.

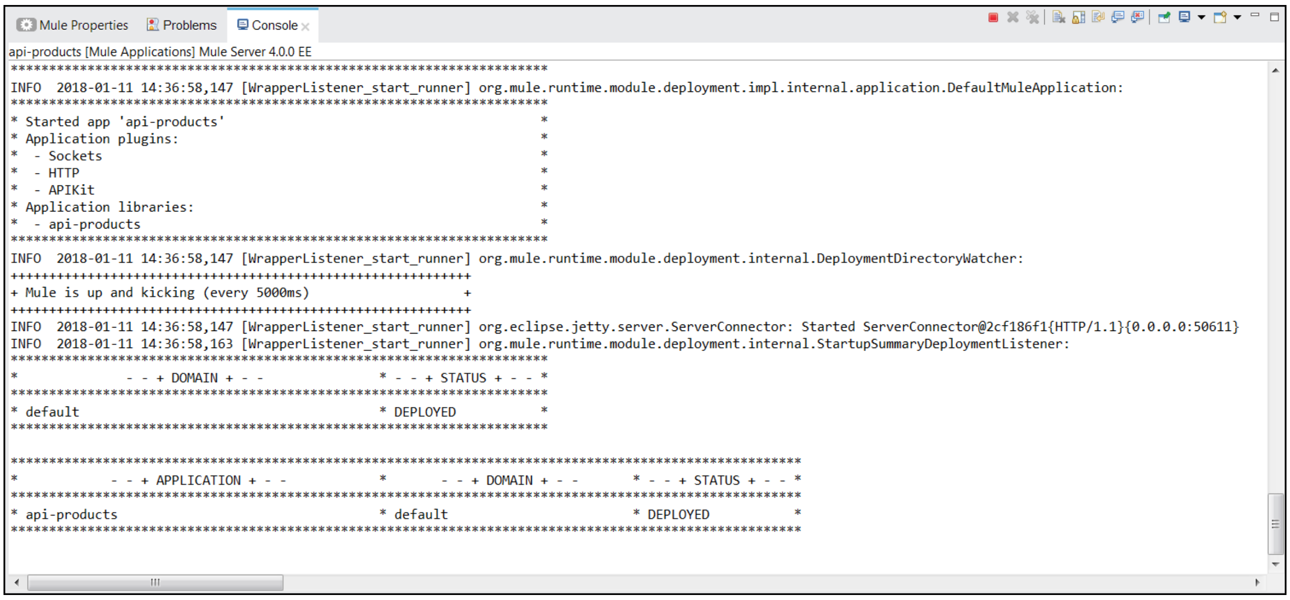
    The static response returned by the auto-generated **Transform Message** allows you to to test the stub immediately after generation. These flows can be enhanced to provide more advanced mock services as well as evolve into the full API implementation.

 api-main Listener APIkit Router Error handling On Error Propagate type: APIKIT:BAD_REQUEST	**api-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.
 get:\product:api-config Source Transform Message Error handling	**get:\product** This flow returns all the products from the database.

 post:\product:application\json:api-config Source Transform Message Error handling	**post:\product** This flow creates a new product in the database.
 get:\product\(id):api-config Source Transform Message Transform Message Error handling  roduct\(id):api-config e Transform Message Transform Message andling	**get:\product\(id)** This flow is used for getting the detail of a product based on the product id.
 put:\product\(id):application\json:api-config Source Transform Message LOG Logger Error handling	**put:\product\(id)** This flow is used for updating a product based on the product id.

 delete:\product\(id):application\json:api-config Source Transform Message Logger Error handling	**<u>delete:\product\(id)</u>** This flow is used for deleting a product based on the product id.

11. To check the API, let's run it within Studio first. Right click **api-products** in the Package Explorer view.
12. Select **Run As** > **Mule Application**.
13. The application will start running, and the console will show the Mule Runtime logs. Wait until the console shows the application has completed deployment.



```
api-products [Mule Applications] Mule Server 4.0.0 EE
**********************************************************************
INFO  2018-01-11 14:36:58,147 [WrapperListener_start_runner] org.mule.runtime.module.deployment.impl.internal.application.DefaultMuleApplication:
**********************************************************************
* Started app 'api-products'                                         *
* Application plugins:                                               *
*  - Sockets                                                         *
*  - HTTP                                                            *
*  - APIKit                                                          *
* Application libraries:                                             *
*  - api-products                                                    *
**********************************************************************
INFO  2018-01-11 14:36:58,147 [WrapperListener_start_runner] org.mule.runtime.module.deployment.internal.DeploymentDirectoryWatcher:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ Mule is up and kicking (every 5000ms)                    +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
INFO  2018-01-11 14:36:58,147 [WrapperListener_start_runner] org.eclipse.jetty.server.ServerConnector: Started ServerConnector@2cf186f1{HTTP/1.1}{0.0.0.0:50611}
INFO  2018-01-11 14:36:58,163 [WrapperListener_start_runner] org.mule.runtime.module.deployment.internal.StartupSummaryDeploymentListener:
**********************************************************************
*              - - + DOMAIN + - -               * - - + STATUS + - - *
**********************************************************************
* default                                       * DEPLOYED           *
**********************************************************************

*****************************************************************************************************
*            - - + APPLICATION + - -            *       - - + DOMAIN + - -       * - - + STATUS + - - *
*****************************************************************************************************
* api-products                                  * default                        * DEPLOYED           *
*****************************************************************************************************
```

14. Test the application using the console. Press the link and a new window in the browser will appear.
15. Click the **POST** tab to test the **Create a new Product** method.
16. In the newly expanded **Try it** section, scroll down and inspect the request body.
17. Click the POST button in the **Try it** section. An example response from the flow is provided.

    You can change the Transform Message response to prove to yourself that the console is invoking the actual flows, as well as testing some of the other actions against the resources.

18. Go to the console tab and press the red squared button to stop the Mule runtime server.

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

Summary

In this module lab we created a skeleton API from a RAML stored in Exchange using Anypoint Studio project creation with APIKit. We ran and tested the API on the local Mule runtime which comes embedded in Anypoint Studio.

- Overview

We saw how easy it is to generate a skeleton implementation project from the API design (RAML) stored in Exchange using APIKit with Anypoint Studio. Any other RAML can also be referenced as part of the project creation process, and it is also possible to import a RAML later and generate the interface on demand.

See the APIKIt documentation for more information.

Please proceed to Lab 2: Connect the Product API to a MySQL Database

Take me to the TOP