Lab 3

Run and Debug the API
Overview

In this step we are going to test the implementation that we just finished in the previous lab. To test the API, we are going to use Anypoint Studio. We will also learn some basic debugging steps to gain insight into message payloads and perhaps troubleshooting.

Step 1: Run the Mule API(Application) from Anypoint Studio

In Lab 2 we have formatted our inbound and outbound messages, we can proceed to run the application once again, but now actually working with a real already deployed .NET SOAP service, that inserts data into our Order Database.

1. Open up Anypoint Studio to the workspace containing your project.

2. Before you run your application in Debug mode, it is a good idea to use a breakpoint so you can stop the flow and interrogate the message as it goes through the flow. This way, you can see how each processor, like a transform component, changes the message content.

3. Let's turn on a breakpoint for the first Transform Message processor by right-clicking inside the component and selecting the Toggle breakpoint option.

4. With the breakpoint turned on (see red dot), save the project by clicking the button.

5. Then right-click the application (or on the top project folder in Package Explorer).

6. Select Debug As > Mule Application
Alternatively you can right-click on the api project canvas and select **Debug project** like below.
NOTE

Anypoint Studio deployed this application to an embedded Mule run-time. There is no need to deploy to a separate Mule server environment. The developer will be able to develop and test the application locally until it’s ready to be deployed to a shared development or upper environments.

7. You will also be prompted to switch to a Debug perspective. Hit the yes button here.

![Confirm Perspective Switch](image)

Now you are in the **Mule Debug** perspective. This perspective gives you many views.

![Mule Debug Perspective](image)

8. Test the application using the **APIKit Console** link on the lower right. Click on the Basepoint URL: Link. This should open up a browser window to the API Console. The console will look like this in the browser.
9. Click the POST link on the left then the TRY IT link on the top right. This will allow you to test the Orders resource (/orders) using the Create an Order (POST) method. Make sure you explore the Parameters, Headers, and Body tabs on this page to see its contents.

10. You will notice the request body is a pre-populated payload shown in the Body tab. Within this payload is a customerId that is used to map to the SOAP request as defined earlier. Now Click on the SEND link.
11. **Click** the **POST** button. Soon you will see back on the **flow window** the first **breakpoint** highlighted with a dashed outline.

Now you can also see the message details, like properties in the **Mule Debugger** tab.
If you *hit* the **next processor** button (see red arrow above) you will move the message to the next **processor** in the flow. As mentioned before, this is a good way to see how the flow treats the message and it's contents step by step. For more details on using the debugger, check the link provided below.

12. If the request is successful, you should see a **201** for *resource created* response similar to below. Remember this is no longer a mocking service responding. It is a live application deployed to the Mule run-time server, where the **POST** is calling out the a **SOAP** service. You just tested your end-to-end API implementation of the **POST flow** running in your environment.
13. Go to the console tab and press the red button to stop the Mule run-time server.

You can download a complete project from exchange.

**Summary**

In this lab, we were able to:

- Run the Mule app from **Anypoint Studio**
- Test the **Order Create Method** using a SOAP Web Service endpoint to create an order
- Use the **debugger** to see the message traverse the application flow for message insight and troubleshooting purposes
- Deploy a complete **Order API** for DotNet directly to **CloudHub**

Go Further:

- See the link **Studio Visual Debugger** for more information

**Congratulations! You have completed Lab 3.**