# xLab 4: Design a new API with RAML

[Lab 4: Design a new API with RAML](#_witrf0b4iuwq)

[Overview](#_lglsi286mbtm)

[Step 1: Design the Order Entry API using RAML](#_ihqkszpczhng)

[Step 2: Enable the Mocking Service](#_tluvk7kck34r)

[Step 3: Test the Mock Service with SOAP-UI](#_8b819g7gitpr)

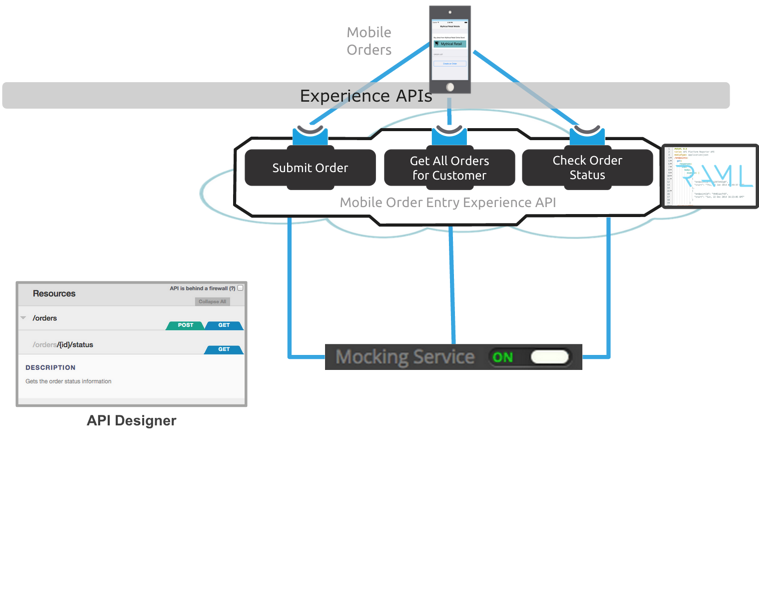
[Summary](#_knuzle5uc8kd)

## Overview

The previous labs focused on hooking existing services into the Anypoint Platform for APIs to allow them to be secured, managed and monitored. This lab will start a series of labs that follow an **API-first design approach**.

The first part of the API Lifecycle is planning and designing. The goal of this phase is to be able to design APIs that are easy to use for their intended target audience. You will use the Anypoint Platform’s API Designer to plan and design the Order Entry API for submitting orders and checking order status. The Order Entry API will follow REST best practices to promote adoption to your consumers. You will use an API-first design approach using the RESTful API Modeling Language (RAML) standard. For more information on RAML, please see:<http://www.raml.org>.

You will also utilize the Mocking Service to unlock resource dependencies of your RAML design. This significantly cuts down the time spent building the mobile application by turning the RAML design over to the mobile developers immediately. Developers can utilize the mocked up API capabilities to produce a "working" mobile app that calls the Mocking Service.



In this lab, we will start designing a new REST API and test it before implementing it.

You have a new requirement for a mobile commerce application that customers can use to order products and check on order status. To do this, the IT team would like to build a mobile app that will use an Order Entry API to access information in the order processing system.

Using the Anypoint Platform for APIs API Designer, you will now design the resources needed for the Order Entry API:

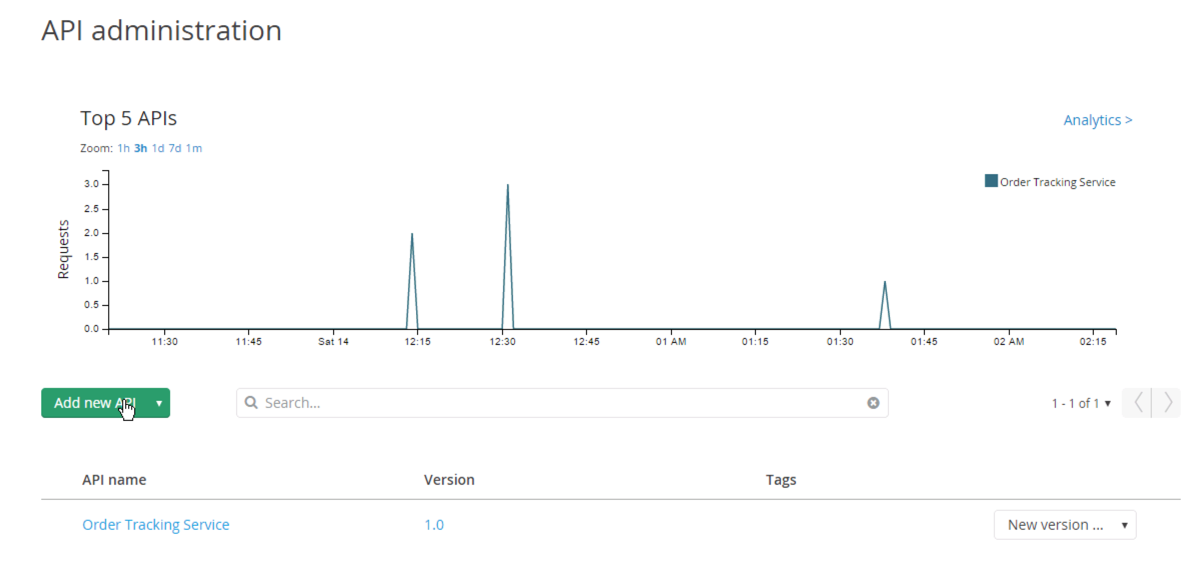
● GET for retrieving the existing orders

● POST for submitting a new order

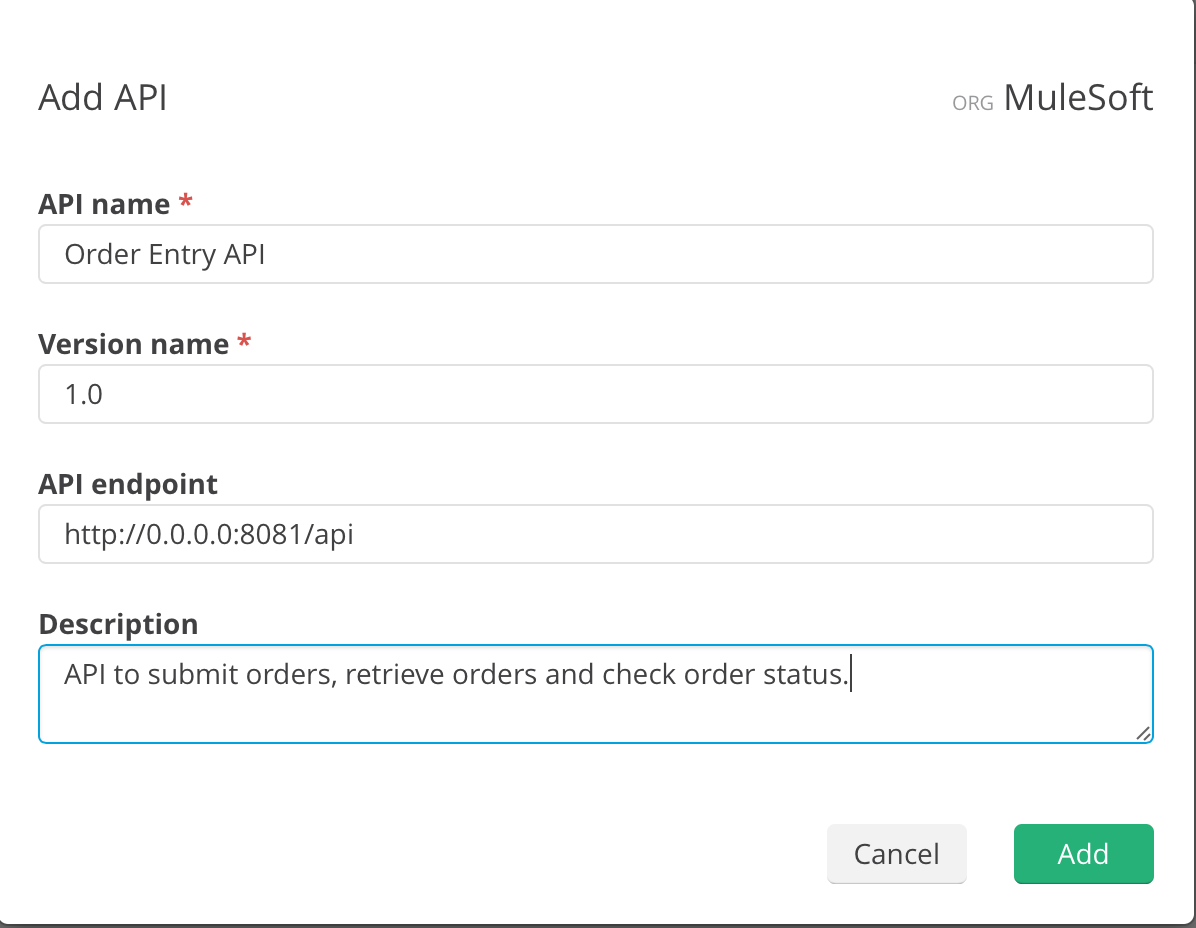
● GET for retrieving the state of an specific order

## Step 1: Design the Order Entry API using RAML

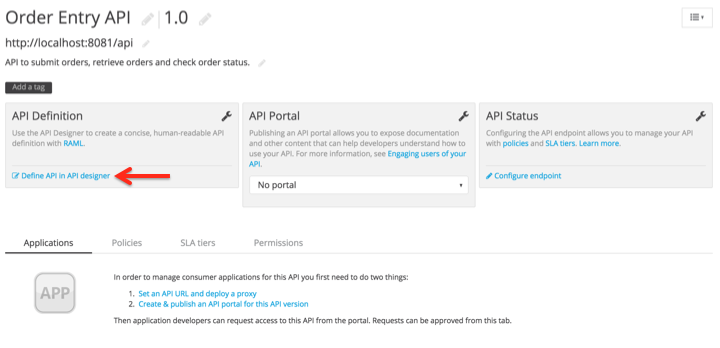
In this step, you will create your first API and design it using the Anypoint Platform for API’s API Designer. The API you design will have GET operations to check order status and list existing orders. A POST operation to submit a new order.



1. Go to the browser
2. In the Anypoint Platform API Administration, click **Add New API**.

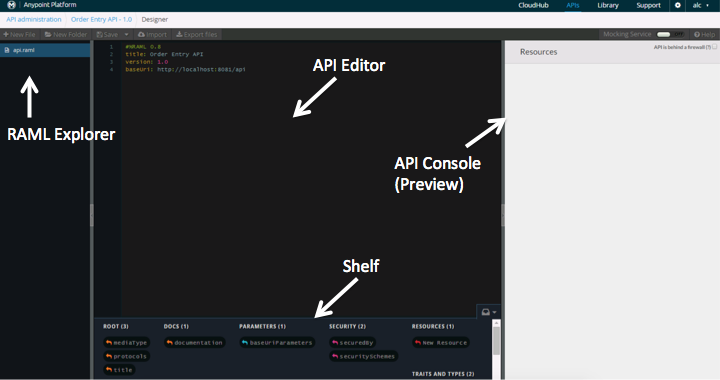


1. Name the API **Order Entry API**.
2. Set version to **1.0**
3. Enter an API Endpoint of [**http://0.0.0.0:8081/api**](http://localhost:8081/api)
4. Add a description as shown above.
5. Click **Add.**



Let us now define the API with RAML.

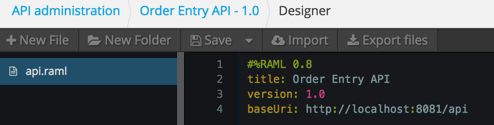
1. Click on **Define API in API designer.** This will bring you to the API Designer.



The **API Designer** will provide an empty API definition to start with. The screenshot and table below gives an overview of the various sections of the API Designer that you will use.

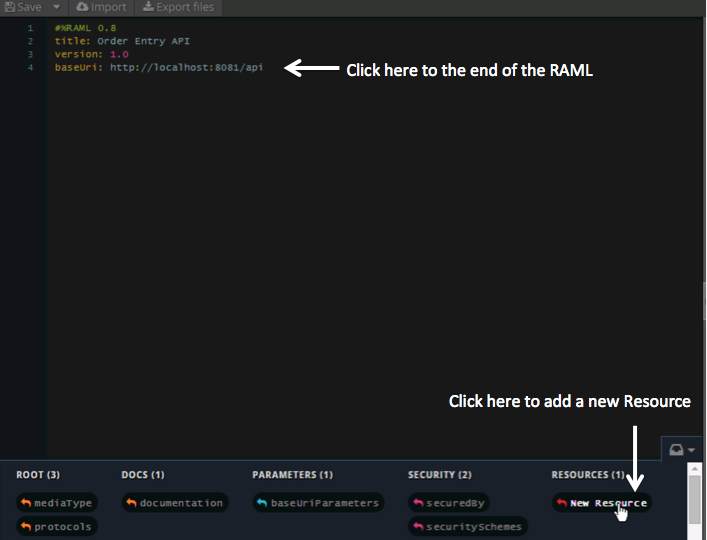
|  |  |
| --- | --- |
| RAML Explorer | You can work with different RAML definitions at the same time. |
| API Editor | Where you define your API. |
| Shelf | Where you have suggested commands available to use depending on where you are. This is contextual. |
| API Console | This is a live preview of your API. As you modify your RAML, the changes will be reflected on this pane. This is what users of your API will see and use to understand how to use the API. |

|  |
| --- |
| **NOTE**: For usability (if you prefer a light theme, instead of the default dark theme), you can toggle the background color from black to white by entering the following in the browser address bar: j**avascript:setTheme('light'); void(0).** You can set it back to black by enteringj**avascript:setTheme('dark'); void(0).** In Chrome for MacOS, you can use the hot-key [**Ctrl+Shift+T**] to switch the theme.  For convenience, there are bookmarks saved in Chrome to allow you to switch the theme as shown: |

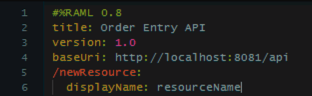


Using the API Editor, you will enter the RAML definition. You should already have the title and version to start with.

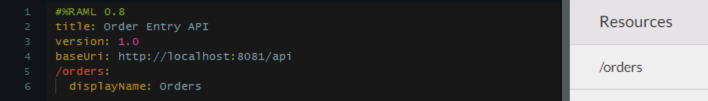
Now you need to add resource. A resource is the definition of an entity that your API will handle.



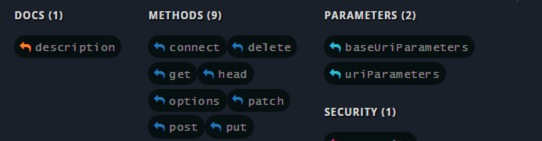
1. Position your cursor after the **baseUri**
2. Click to the end of the RAML
3. At the bottom **Shelf** section, click **New Resource**.



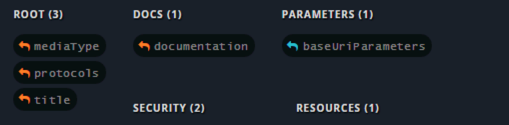
You should see a resource called **/newResource.**



1. Rename the resource to **/orders**
2. Set the display name to **Orders**. As you type the RAML, the API Console on the right gives a preview of your API. That is, a graphical view of what you will get with this RAML definition.



Notice the context sensitive suggestions on the Shelf.

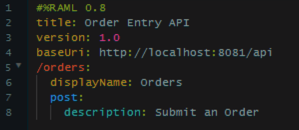


1. Click on /orders on the line above. Notice the suggestions are different. The Shelf is context-aware and will make suggestions depending on where you are.

****

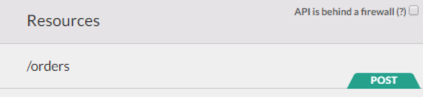
Add a POST method for this resource.

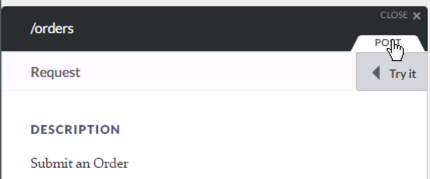
1. Click at the end of the RAML
2. Click **post** from the Shelf.



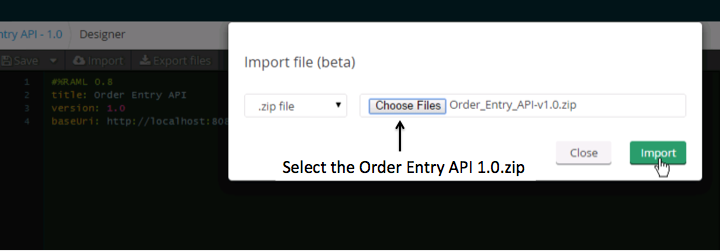
This will add a new line with a **POST** and another line with **description**.

1. Set the value for the description as Submit an Order.





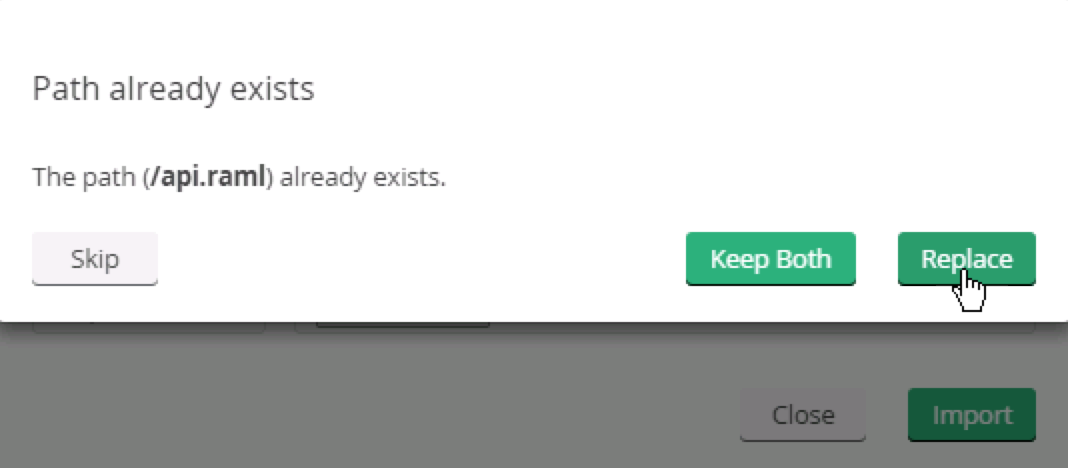
You should see the **orders** resource and the **Submit an Order** post method rendered in the API Console.

****

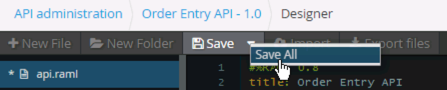
Let's import the whole definition for the API.

1. Click on Import and then select the zip file:  
   **C:\Users\msps\Desktop\Workshop\APIs\Order\_Entry\_API-v1.0.zip**

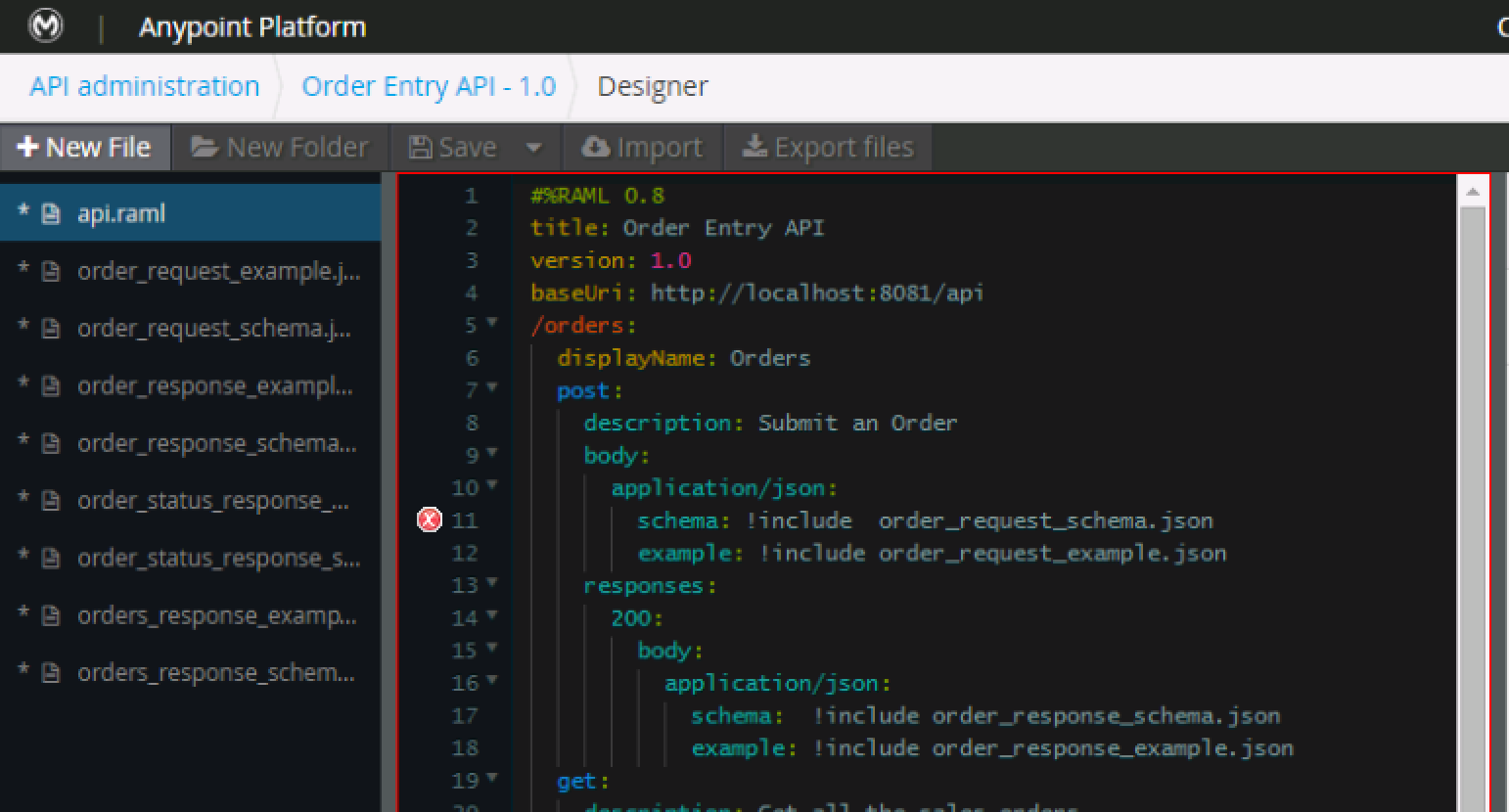
Note: The [Order\_Entry\_API-v1.0.zip](https://drive.google.com/file/d/0B_u3GZs5g0-lOGJGY0QteHY5dHM/view?usp=sharing) API project can be downloaded from here if needed.



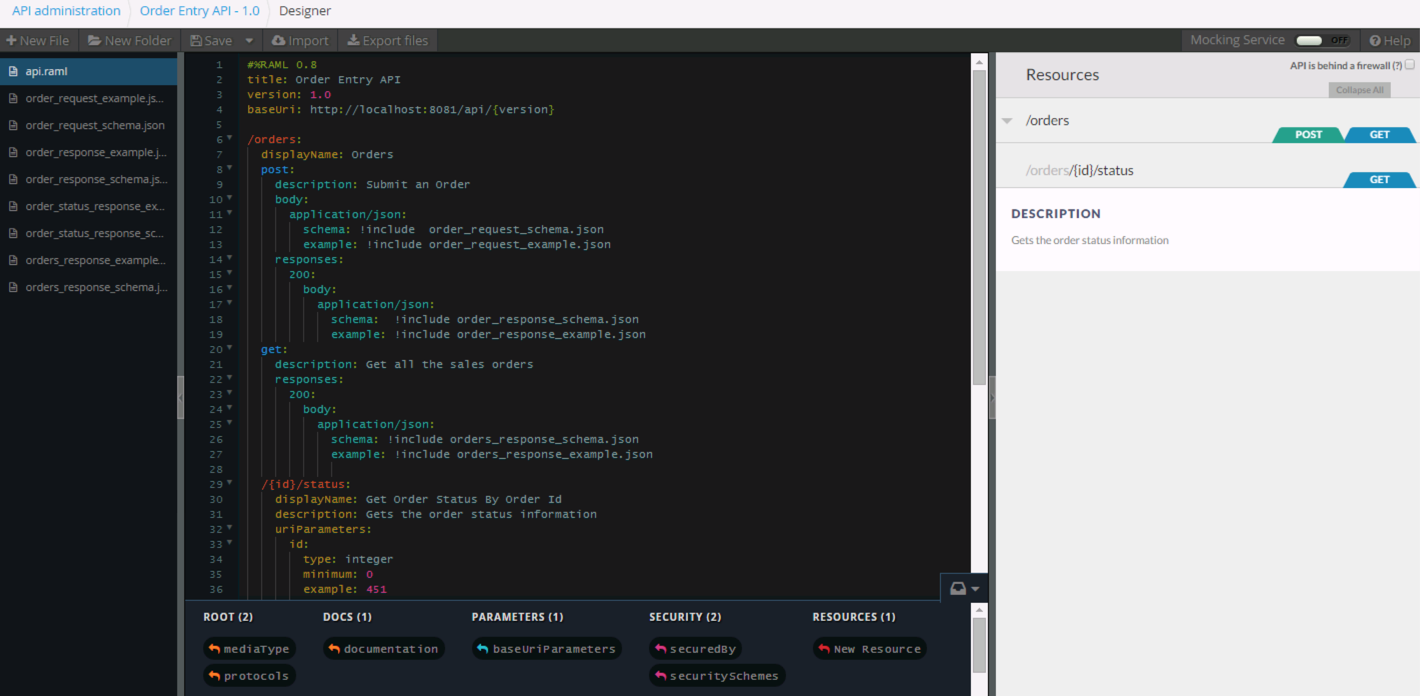
1. If you get a message saying the Path already exists, just click **Replace**. This will replace the RAM you started defining with the completed one you are importing.
2. You will see an error after import because the reference files are not yet saved.



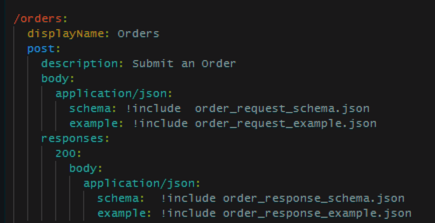
1. Click **Save All**.
2. **Refresh** the page. The error should disappear.



1. If you still see an error just add a space at the end of any line in the RAML
2. Click **Save** and the error should disappear

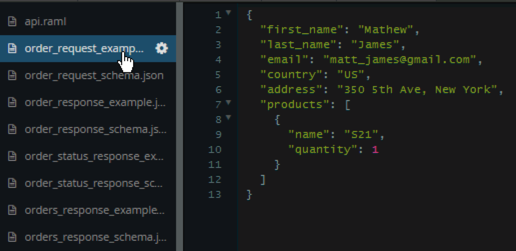


You will see the full definition of the API including the JSON examples and schemas for each method and the preview in the console of the created resources as well.



You will see:

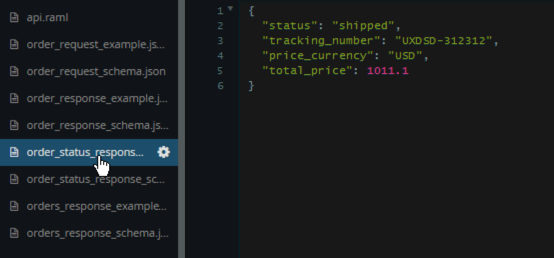
* the **Submit an Order** POST method we created in the **/orders** resource to create a new order.
* a description of the method.
* a body type of application/json complete with the body’s schema and example data which are in referenced files so they can be shared and reused.
* the responses. For this method, there is only a 200 response.
  + the content type is application/json
  + we reused the **schema** and **example** data via the “**!include”** syntax.



1. Explore the included files to see what they look like.



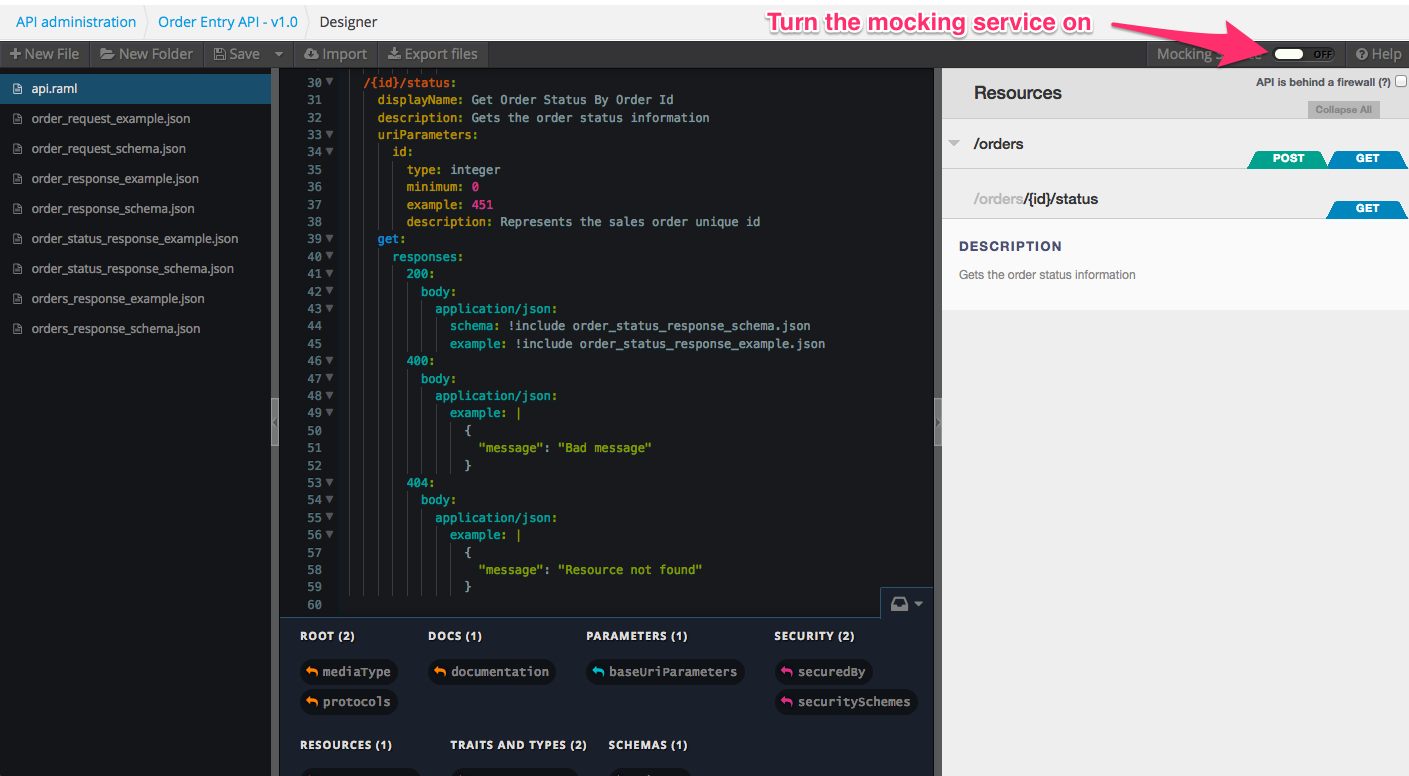
1. Go to the bottom, you’ll see the method for **Get order status by order id.** This resource includes URI parameter called {**id**}. You’ll see the information about it, such as the **type**, **example** and **description**, under the resource’s **uriParameters**. You’ll also see the **response** information.



1. Examine the JSON response example and schema associated to the files being referenced by clicking on the files representing the examples and schemas.

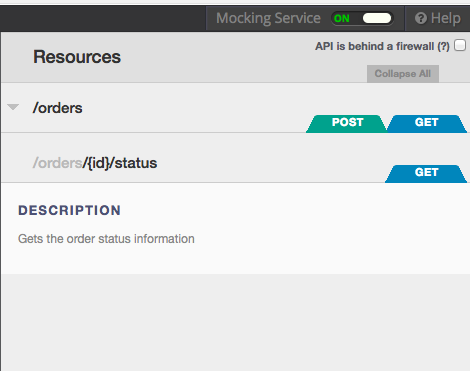
## Step 2: Enable the Mocking Service

A very useful feature during the design phase is having the ability for your consumers to interact with your API without having to code anything. The API editor provides a **mocking service** that will read the RAML file, create the API/service and return example data responses. The mocking service allows users to interact with the API as if it was built and deployed. This feature allows you to rapidly iterate the API design with it’s consumer to finalize the contract.

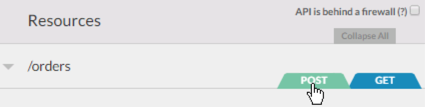


1. Enable the mocking service by switching the **Mocking Service** to “on.”
2. The mocking service will create a URL for the baseUri. The newly created URL will respond to API in the way the RAML file was designed.
3. Click **Save.**

**Make sure you Save the RAML after you enable the mocking service.**

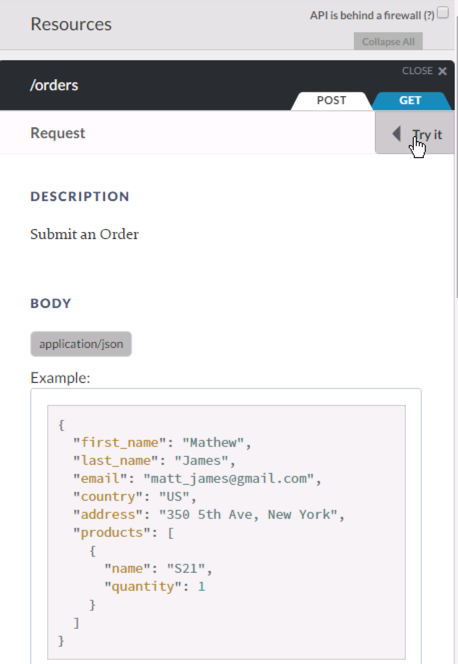


1. Once the Mocking service is enabled, you can use the RAML console to submit an API request and receive the example response defined in the RAML.

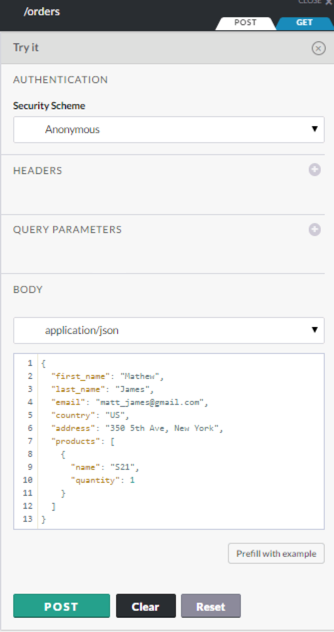


Let's try to create a new order using the Mock service.

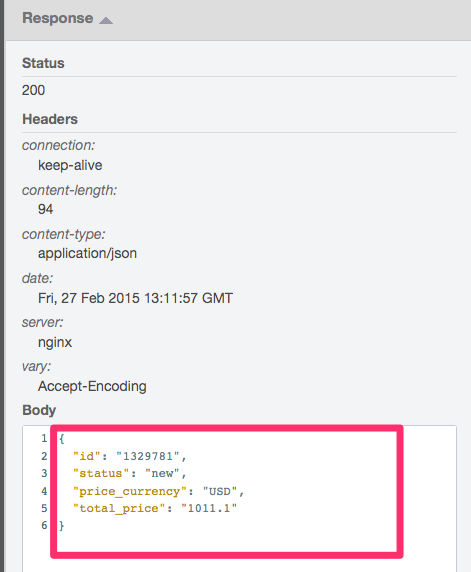
1. Click on the POST operation.



1. Click **Try It**. Here you will see the definition for the Order creation operation which includes:
   * Operation description.
   * Request and response examples and schemas.
   * URI Parameters.



1. Click on the POST button to submit a new order using the JSON in the body.



1. Scroll down and you should see a 200 response with the new Order record in a JSON format from the JSON sample file you provided in the RAML design.

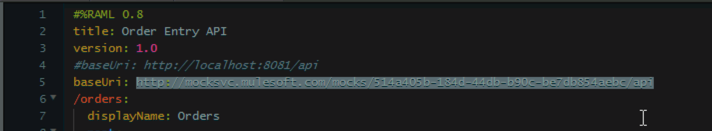
## Step 3: Test the Mock Service with SOAP-UI

You have defined the Order Entry API but you have not implemented the API. In the past, that would mean the a mobile developer, a web designer, a partner, etc. would be on a programmer to build the API before they could start building their app. To accelerate this, many people would code a mock service that would enable them to write their client code independent of the API implementation effort. If you didn’t realize it, you just tested your API using the API Console and the mock service without writing any **code**! Now you can see why this is one of the most popular features in the Anypoint Platform!! Think of how this can accelerate your API/service projects...

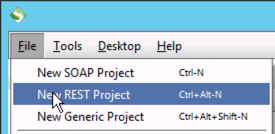
Now let’s test this using SOAP-UI. This will demonstrate accessing the API using a non-MuleSoft product.

|  |
| --- |
| **NOTE**: You can also use other tools like the PostMan extension for Chrome or the RESTClient add-on for Firefox. |

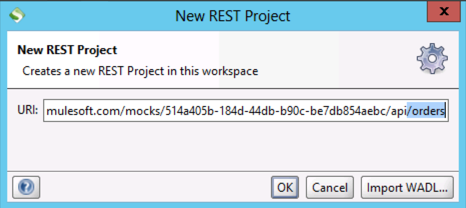
For this test we will POST an order.



1. From the API Designer, copy the baseUri from the RAML. This is the **Mock Service URI** generated for your API.
2. Open SOAP UI from the Desktop: 

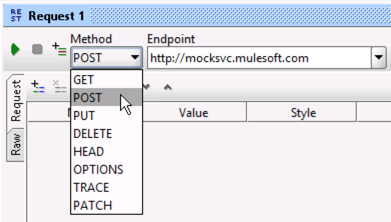


1. Click **File**
2. Click **New REST Project**

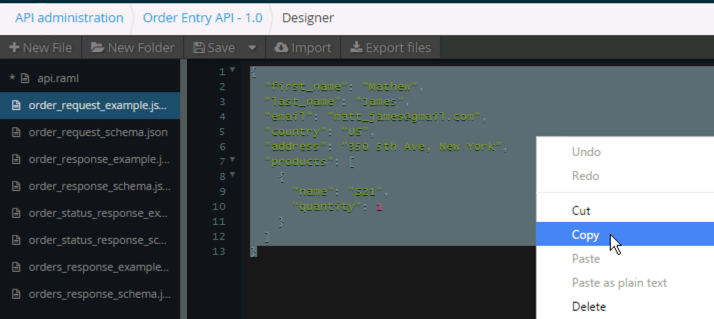


1. Paste the Mock Service URI you copied from the API Console into the URI. This is the root URI for your API.
2. Add **/orders** at the end of the Mock Service URI. This will create your SOAP-UI REST project based on the orders resource and open up the **Request 1** window.

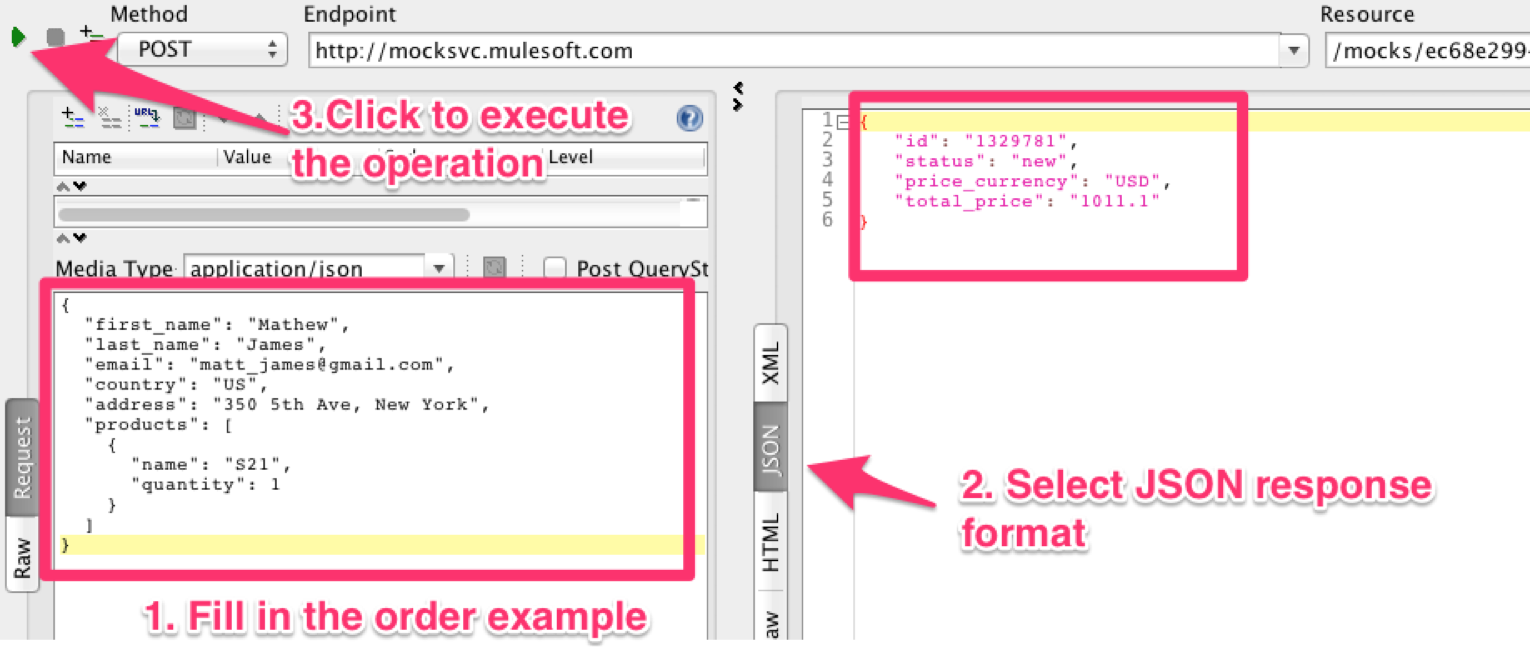
|  |
| --- |
| **NOTE:** Make sure to add **/orders** at the end to test the orders resource. |



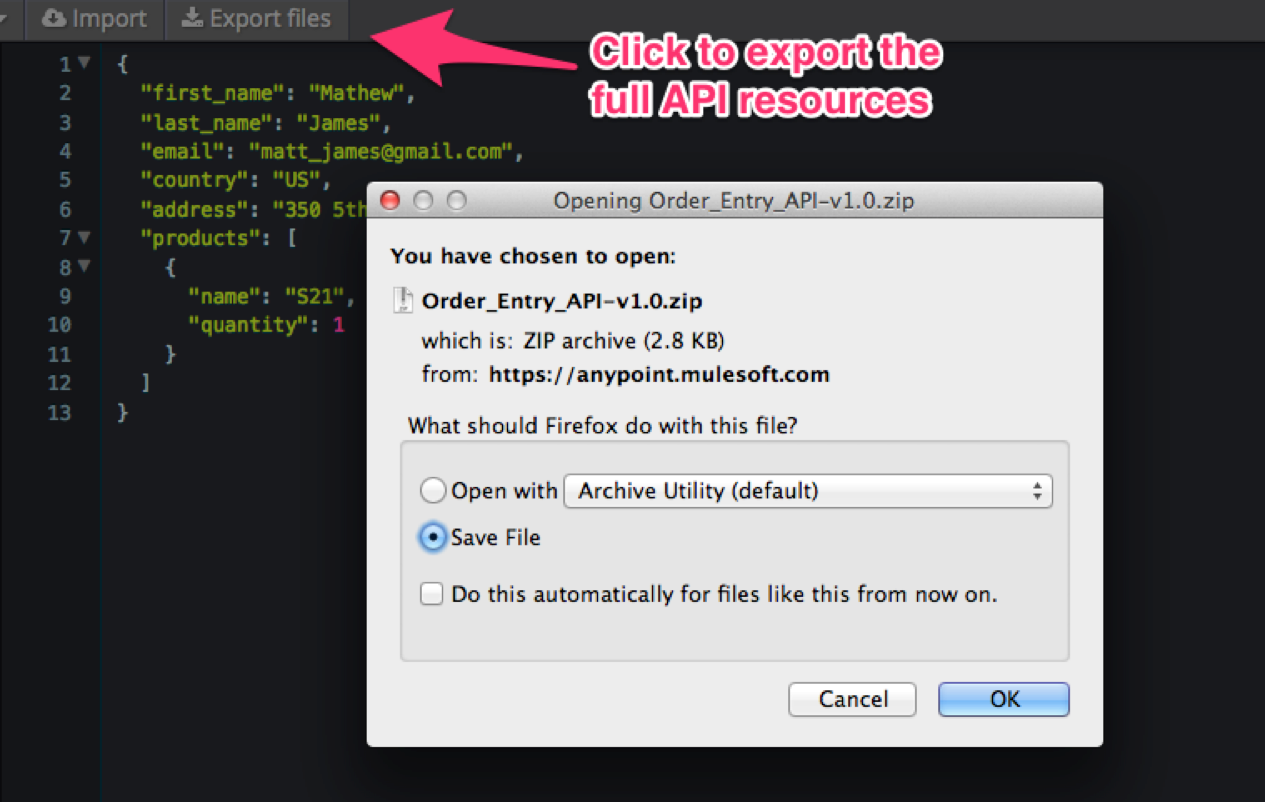
1. Select the POST Method.



1. Go back to the API Designer in the browser.
2. Click on the **order\_request\_example.json** file
3. Click in the RAML
4. Press CTL+A to select the complete contents
5. Press CTL+C to copy to the clipboard.



1. Now go back to SOAP-UI
2. Click in the bottom left box
3. Click CTL+V to paste the sample request into the box
4. Click on the Play button (3) and you should see the JSON response (2) from the mock service.



1. Once you have tested your API go to the API Designer again and export the full API definition. Your next lab will use this zip file to implement the API.  
     
   **NOTE**: Depending on the browser you are using the file will be automatically saved or you will have to confirm to save it.

## 

## Summary

In this lab, you completed the following steps:

[Step 1: Design the Order Entry API using RAML](#_ihqkszpczhng)

[Step 2: Enable the Mocking Service](#_tluvk7kck34r)

[Step 3: Test the Mock Service with SOAP-UI](#_8b819g7gitpr)

We easily created and designed a new experience API for our mobile order entry application, providing the ability to submit orders, get orders and check order status. We leveraged RAML for a **design first approach**.

We saw how the mock service can be utilized to provide application developers an API mock up they can build their applications on. This significantly **speeds up end to end development**. Additionally, we tested the API from an external consumer like SOAP UI simulating a real client of the API.

To learn more about RAML follow this [tutorial](http://raml.org/docs.html).

See [Designing Your API](http://www.mulesoft.org/documentation/display/current/Designing+Your+API) for more information on API design.

Congratulations! You have finished Lab 4.

Please update the spreadsheet indicating you have completed Lab 4.