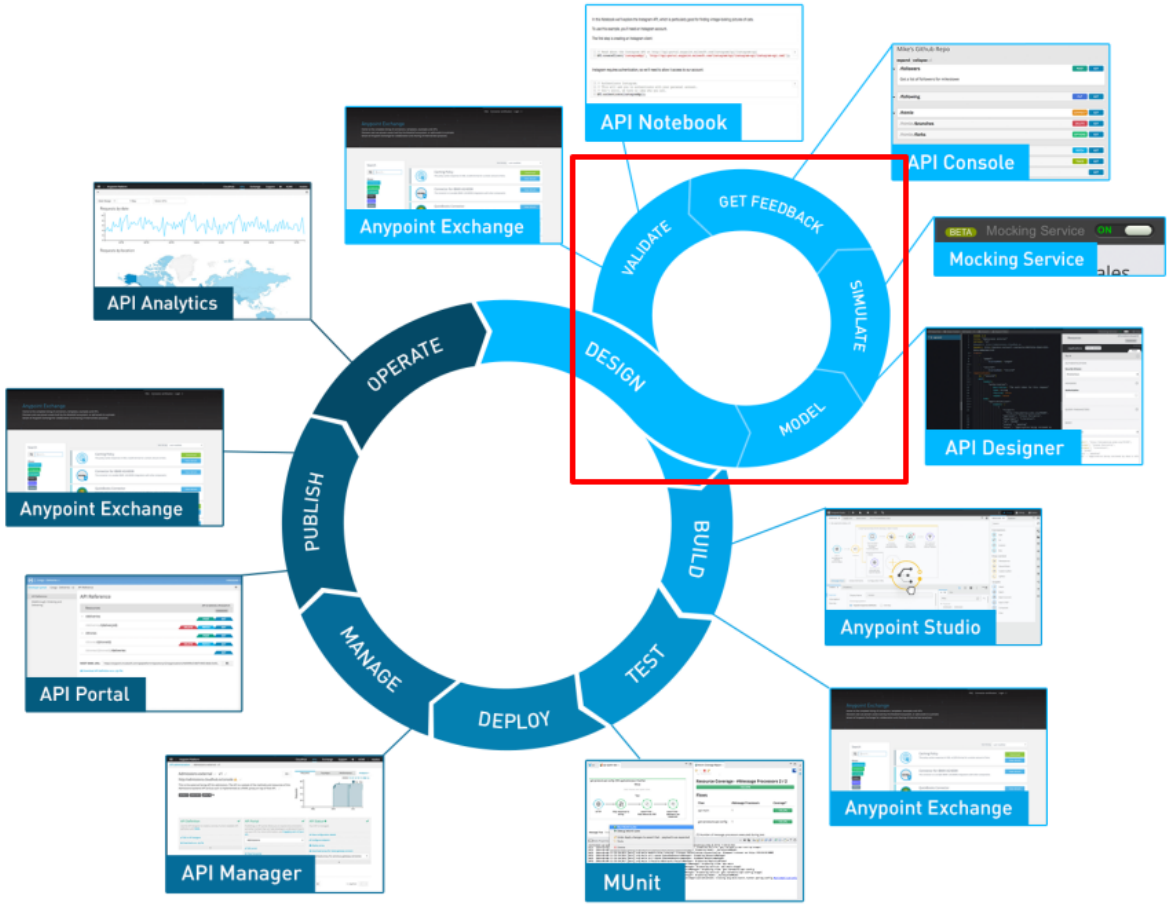
Module 1: The Modern API - Design

Overview

To get started with the workshop, let's take a look at how to discover, design, and publish an API specification using the **Anypoint Platform**. MuleSoft advocates for users to treat APIs as products and therefore adopt a "design first" approach to creating APIs. A "design first" approach is used to enable API consumers the ability to understand, interact, and solicit feedback on the proposed API contract prior to the development effort.



In this first module we will explore the capabilities of MuleSoft's **Anypoint Platform** to design, publish, and discover API specifications.

Anypoint Design Center provides a set of interactive tools that enable API designers to create or modify the specifications for an API. These specifications form the underlying API contract between the provider and the consumer of the API.

API discovery and collaboration is provided by **Anypoint Exchange** and simplifies the way that reusable assets are discovered and consumed across the enterprise.

The first module of the workshop will focus on how to use **Anypoint Design Center** and **Anypoint Exchange** to design, document, publish, and discover the features of a modern API.

The 4 labs in this unit are:

- Lab 1: Search for assets in Exchange
- Lab 2: Design your API
- Lab 3: Publish your API to Exchange

Please proceed to Lab 1

Lab 1: Search for assets in Exchange

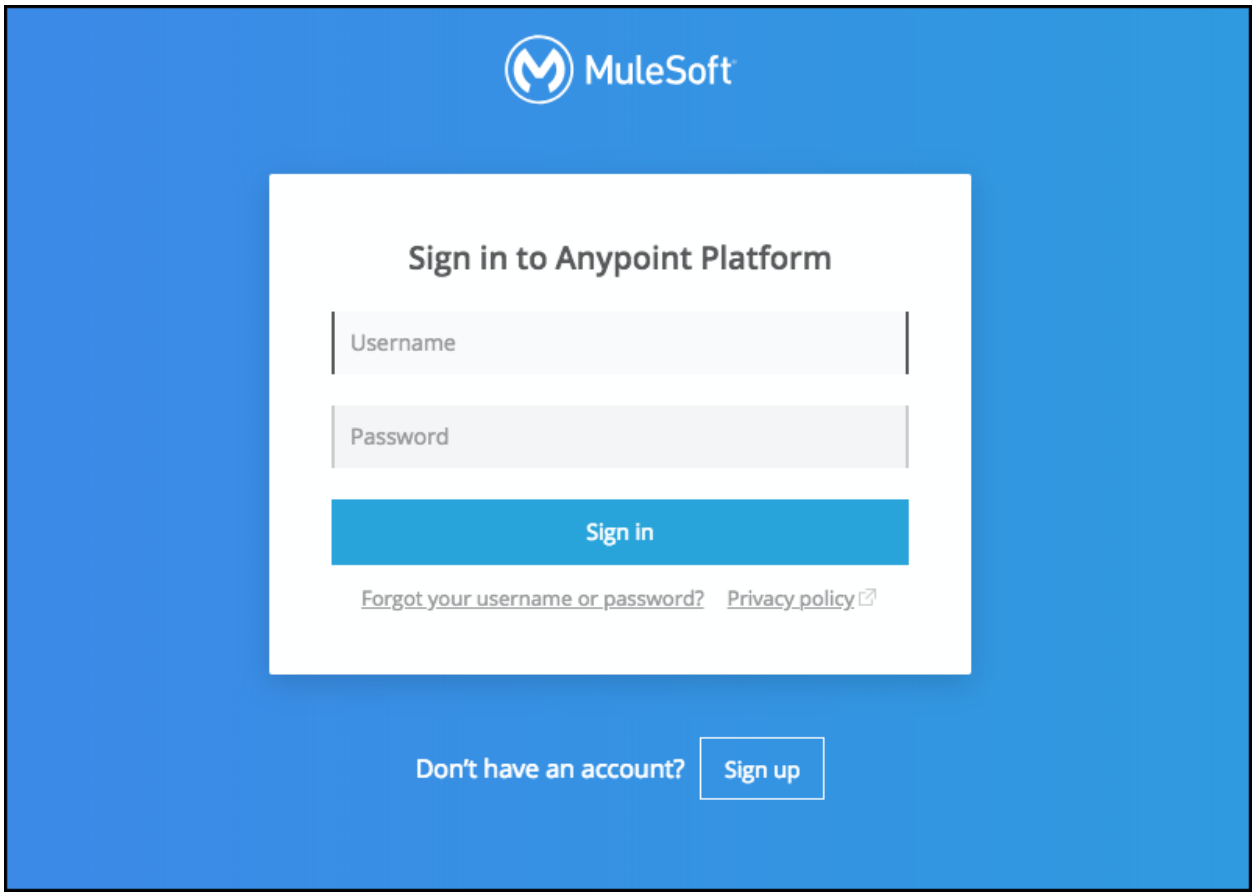
Overview

APIs are the reusable assets that simplify and accelerate the creation of modern software applications. As a MuleSoft developer you will need to consume APIs created by other members of the organization and publish new APIs for others to consume. This new consumption model is the foundation of a new approach for delivering software solutions where APIs form the building blocks of the modern enterprise.

This first lab will focus on using **Anypoint Exchange** to search for APIs and other assets published in Mythical's private exchange.

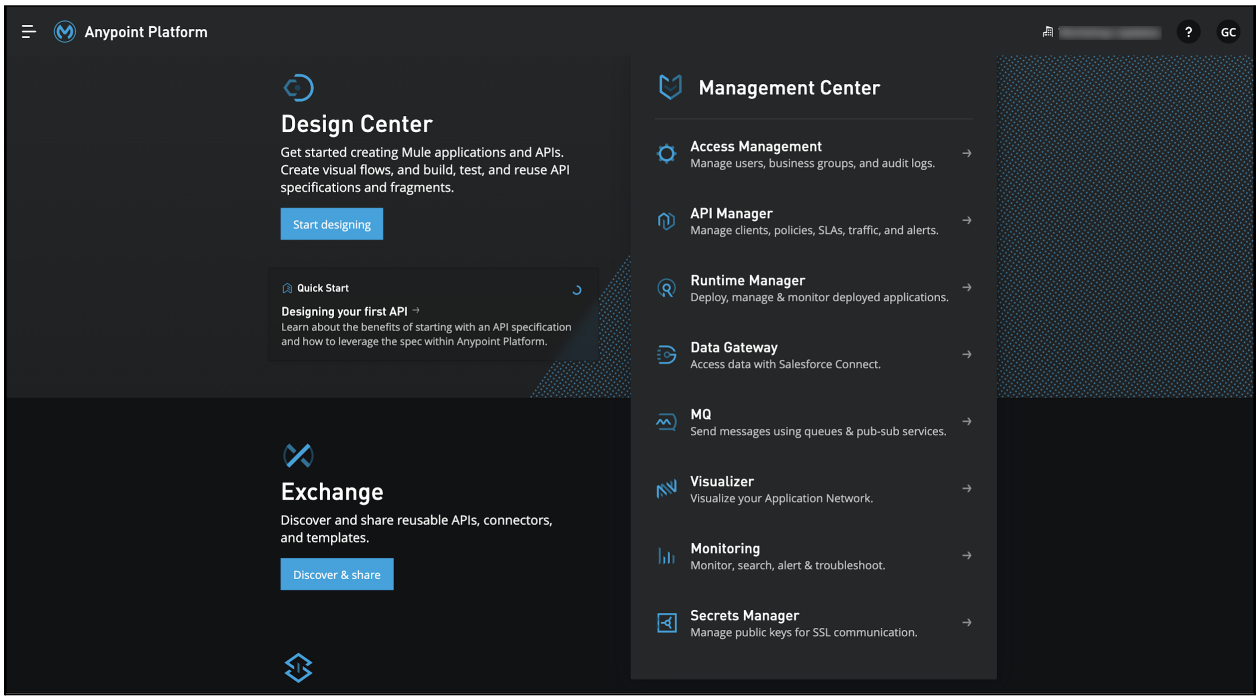
Step 1: Login to Anypoint Platform

1. Go to http://anypoint.mulesoft.com

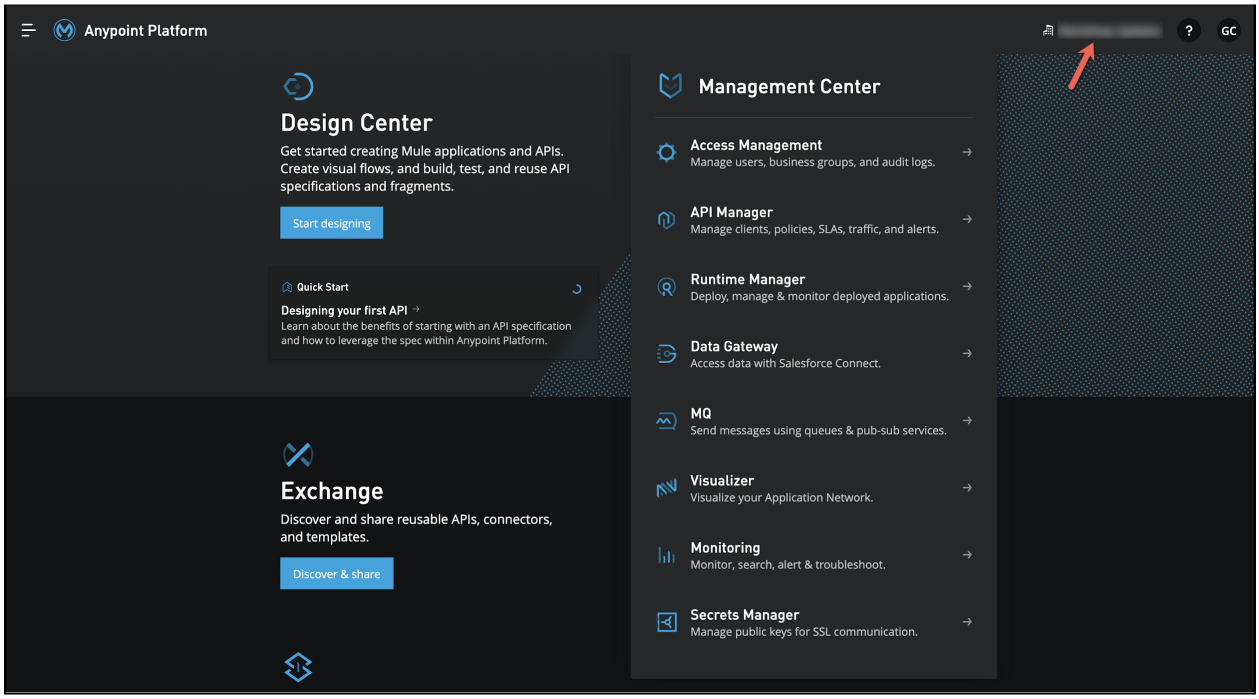


2. Enter your credentials into the Username and Password fields.

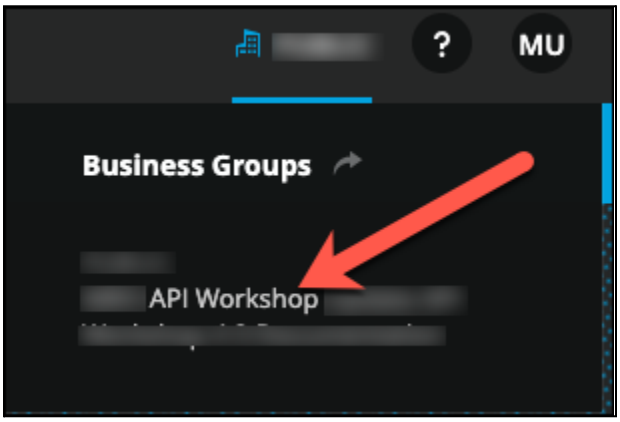
3. You should see the following landing page once you are logged in to Anypoint Platform.



4. Click on the Business Group that is on the right top of the screen. The name will vary with the class.

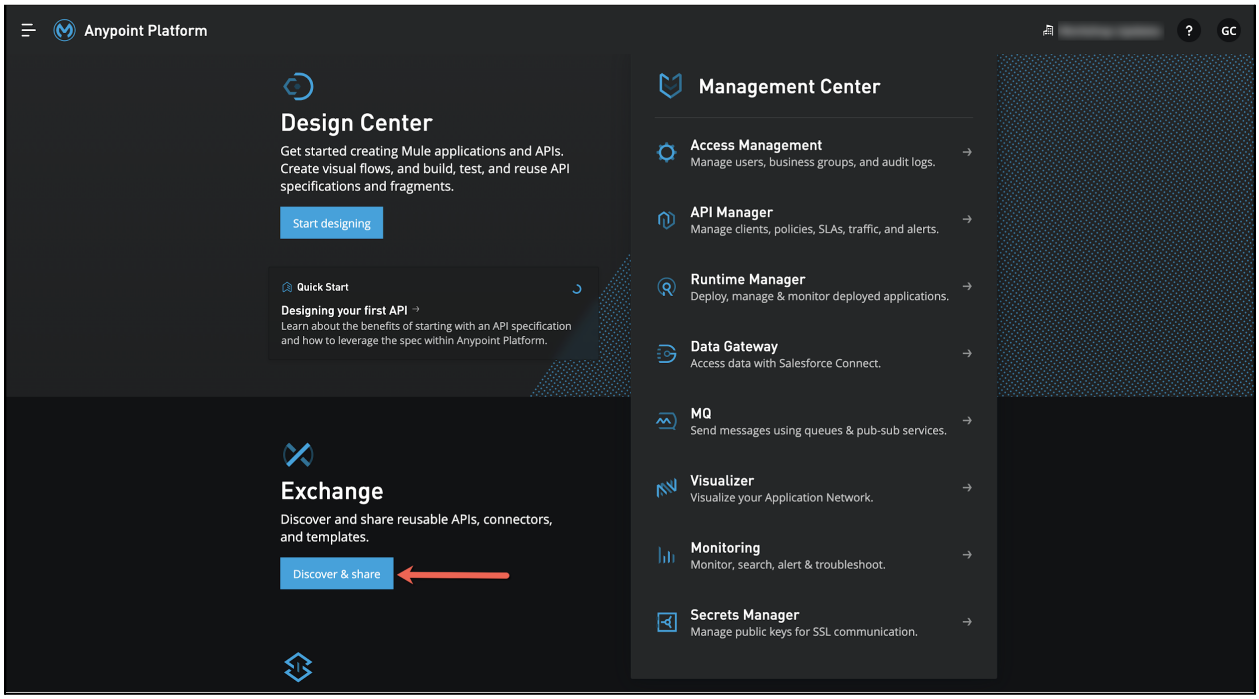


5. There should only be one sub business group. Select the sub business group.

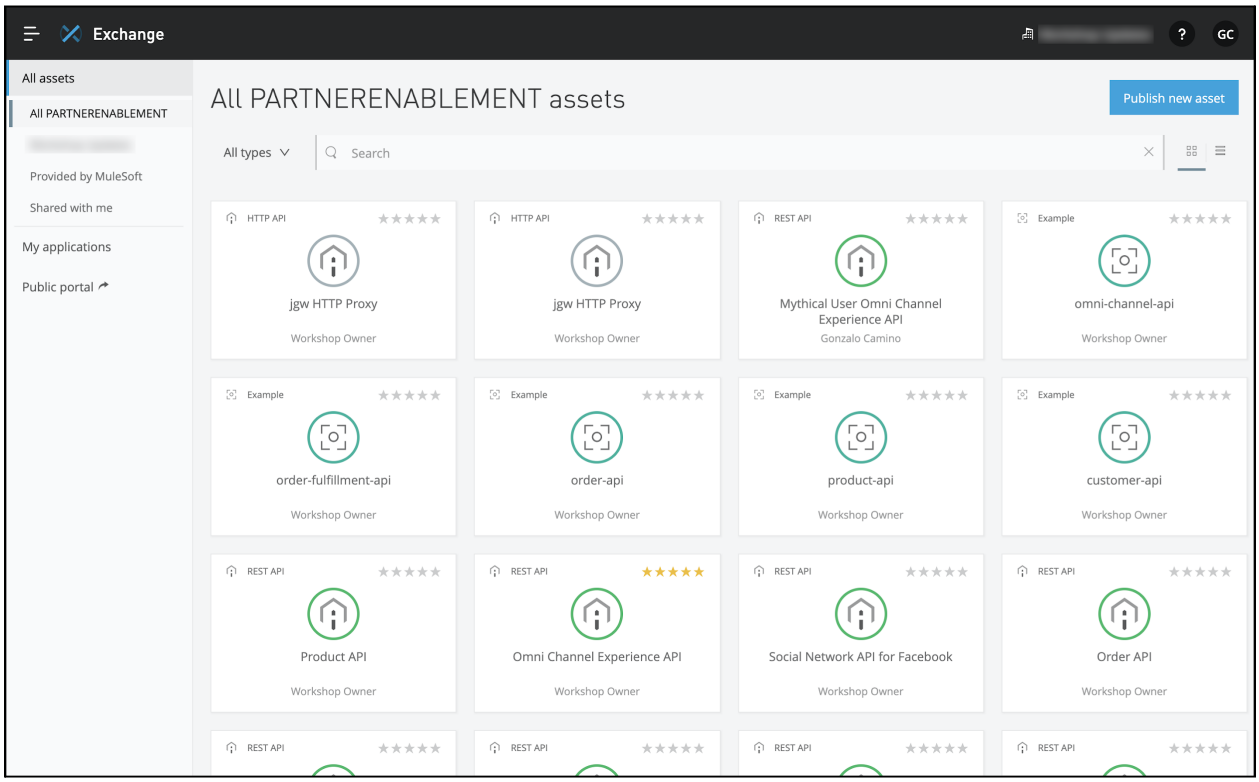


Step 2: Access Anypoint Exchange

1. Click on the icon labeled "Discover & share"



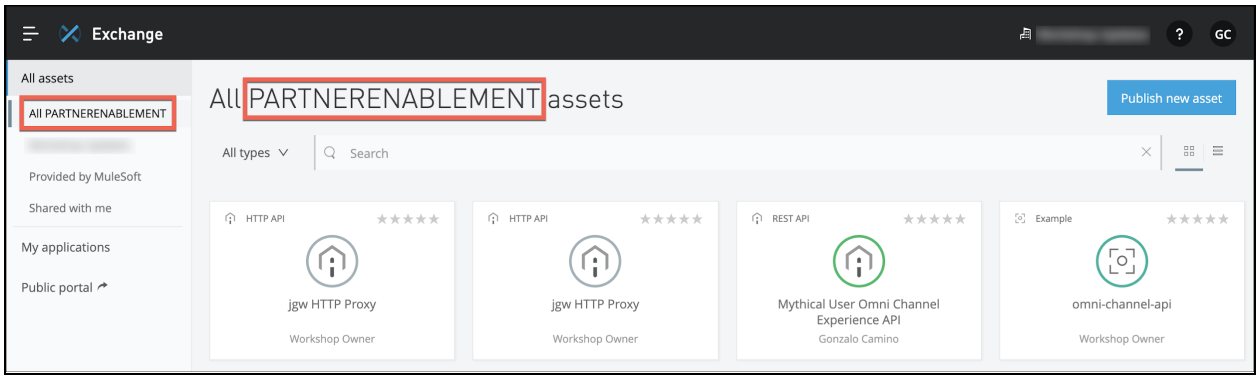
2. You will now be presented the landing page for the **Anypoint Exchange** portal:



Anypoint Exchange should be your initial starting point for just about any project. If you are looking to reuse an API then **Anypoint Exchange** is obviously where you should start. But even if you are creating a new API to be reused by others, it makes sense to first look in Exchange to see if the API already exists. Sometimes you will find that someone else has already undertaken the task of creating the API, or has created a subset of what you need.

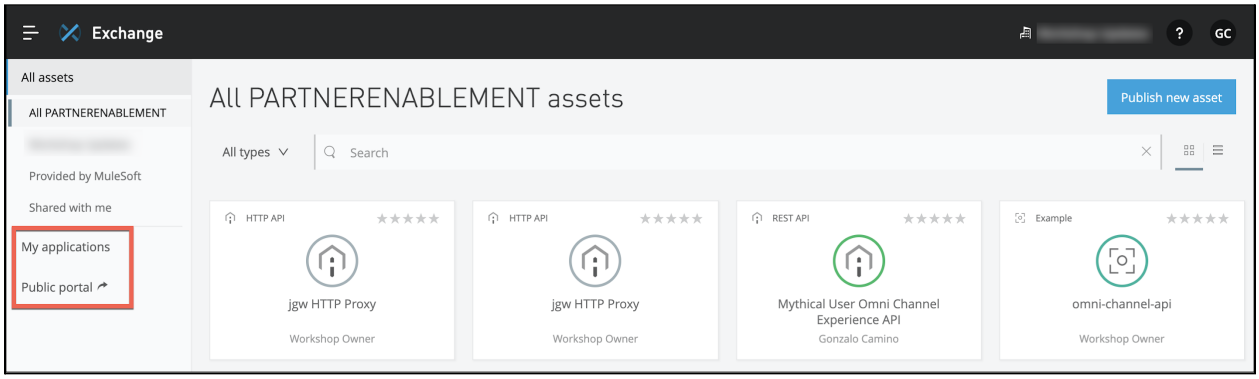
3. **Anypoint Exchange** allows for discovery on the *public* or *private* exchange. Every organization will automatically have a private exchange created for them.

   Software assets stored in **Anypoint Exchange** can only be discovered by users that have permissions for the private exchange for that organization. Furthermore, organization administrators can also create independent *business groups* to provide even more restrictive discovery. **Anypoint Exchange** users can only search within their currently selected business group. The currently selected business group, "Public" in this example, is shown.



As you can see from the screen above, there are organizational search criteria on the left hand side of the page. By default searches will search the private exchange only, but you can restrict or expand your searches by selecting a different organizational level. Select **All assets** to select both MuleSoft public and your private repositories.

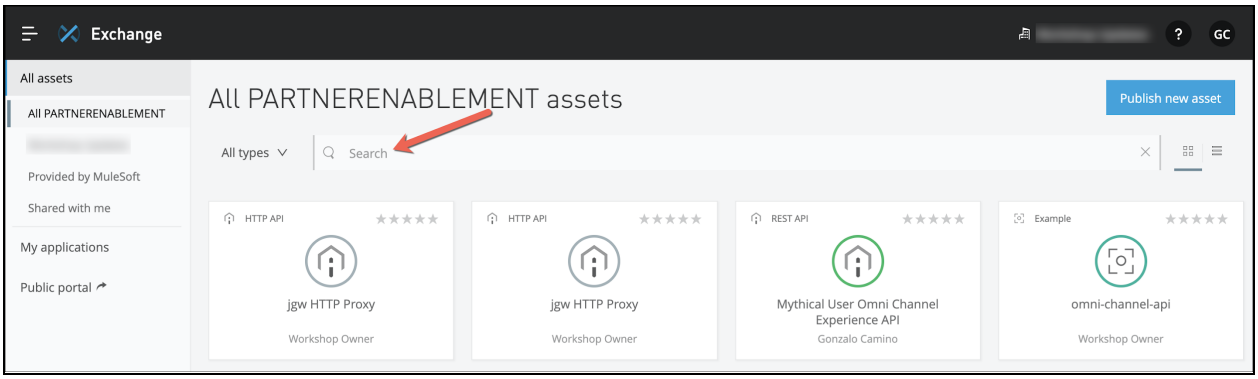
There are two more items behind the last organization **My applications** and **Public Portal**. We will describe them but they are covered in Module 5 and Lab 4.



a. **My applications**: These are the clients registered in the organization that are consuming the APIs that are deployed in the organization.

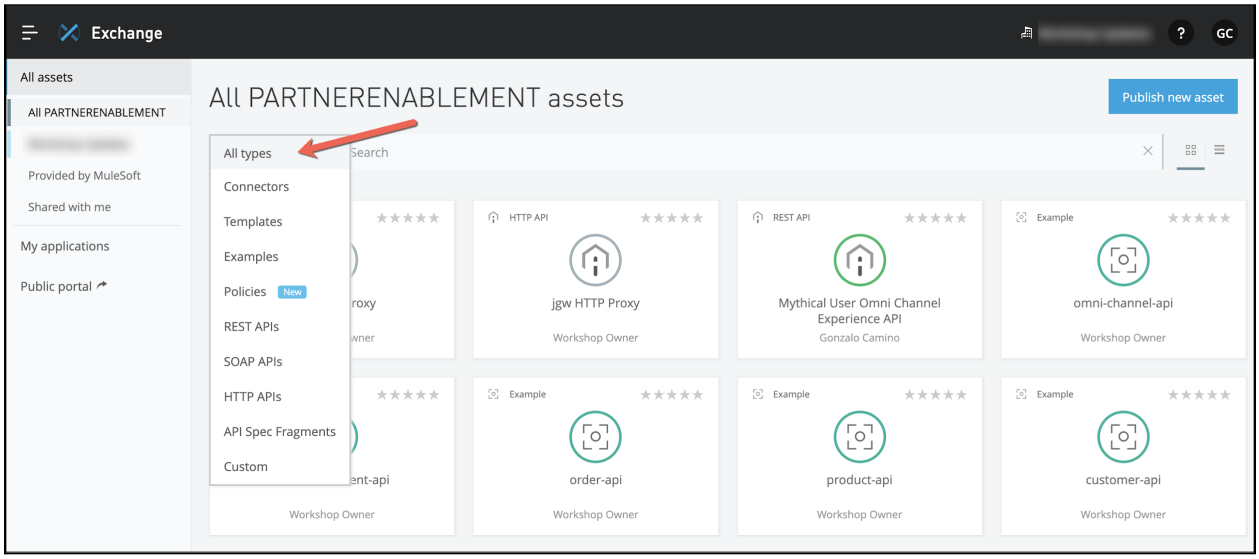
b. **Public Portal**: You can define a public portal to let any user to consume any public org API that the organization may publish.

To search for an API, use the search bar at the top of the Anypoint Exchange portal:



Anypoint Exchange supports keyword searching to find the API you are looking for. However, **Anypoint Exchange** can be used as a discovery tool for more than just APIs. **Anypoint Exchange** is the enterprise repository for the following types of reusable software assets:

4. Click on the **All types** drop down next to the search field to view the different search types.



a. **Connectors** - Packaged connectivity to an endpoint developed and deployed on MuleSoft's Anypoint Platform with third-party APIs and standard integration protocols.

b. **Templates** - Packaged integration patterns that address common use cases and are built on best practices. Applications to which you add your scenario-specific information to complete a use case or solution.

c. **Examples** - Implementation projects that explain development elements within Anypoint Studio and how these can be leveraged to achieve specific API and integration objectives.

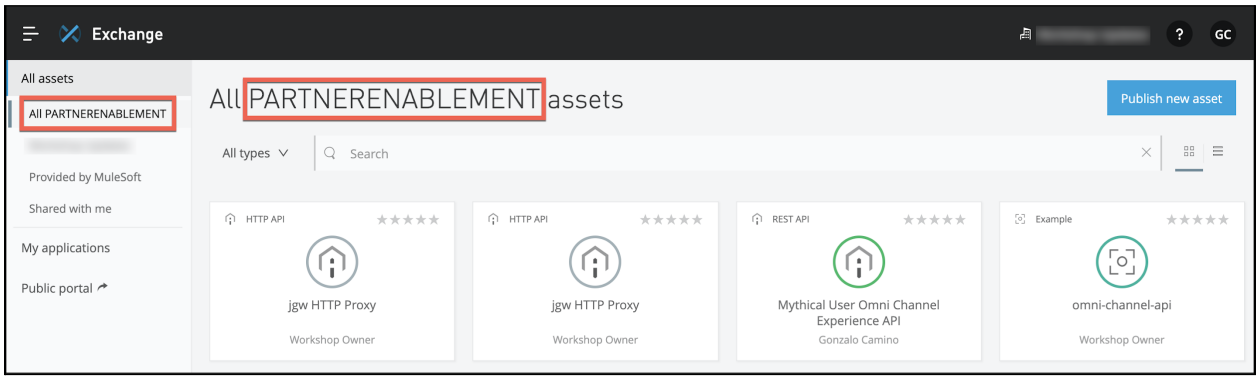
d. **REST APIs** - API descriptions in RAML format that make the consumption of REST APIs faster and easier.

e. **SOAP APIs** - API descriptions in WSDL format that make the consumption of a SOAP APIs faster and easier.

f. **API Spec Fragments** - Shared fragments of RAML files that can be used to assemble new APIs with common traits.

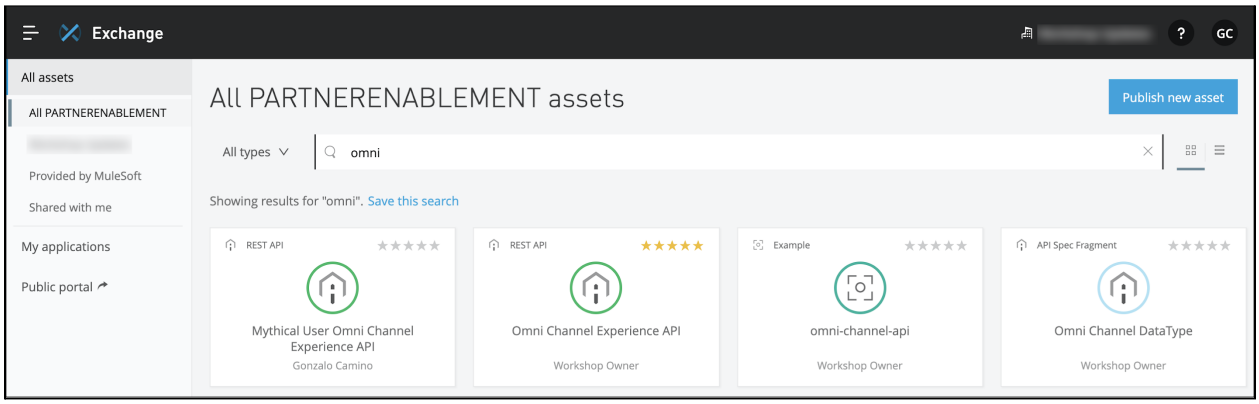
g. **Custom** - A general category for sharing resources such as links, blogs, articles, videos and more.

5. Click on the **All types** drop down again to collapse it.
6. Click on the **All Public** business group. The name can vary by class.

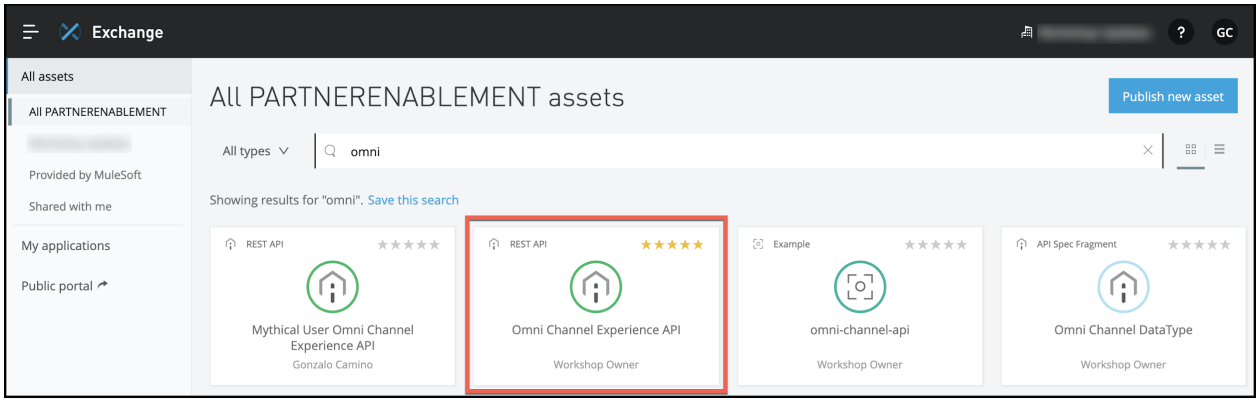


7. Click on the search field and type **"omni"**, then hit the **Enter Key**.

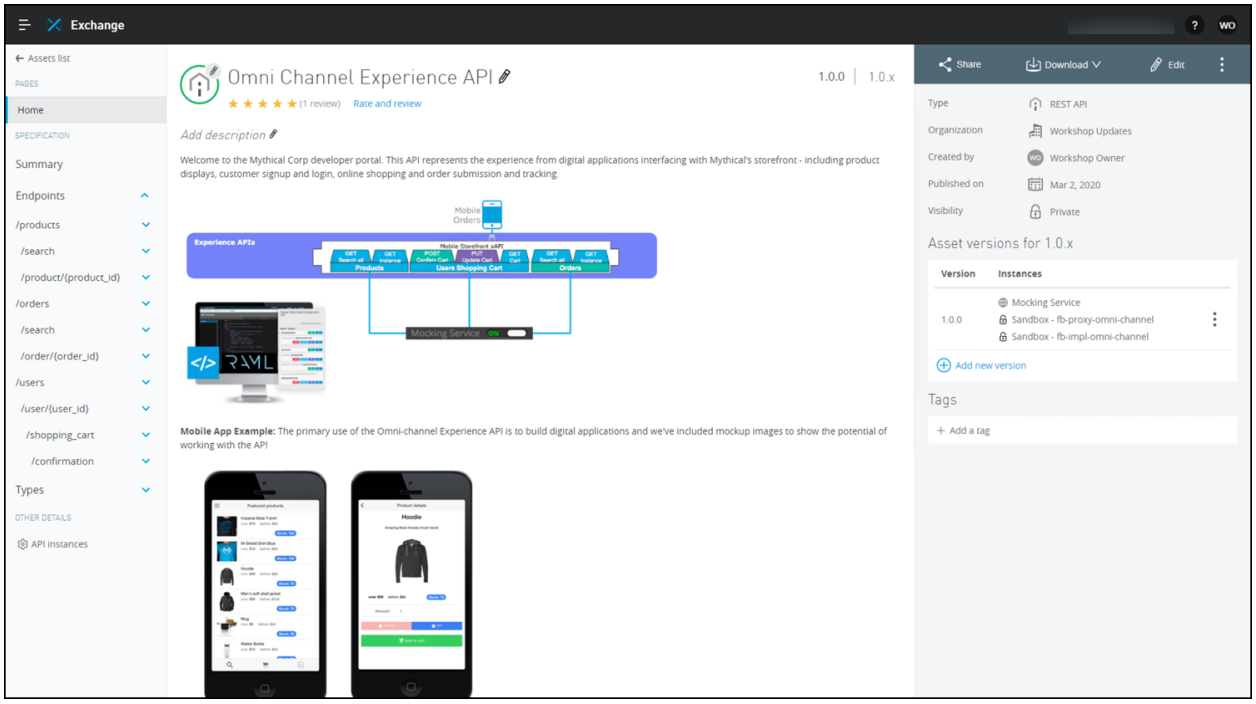
The list of Exchange assets should now be filtered to look something like this:



8. Click on the tile named **Omni Channel Experience API**.

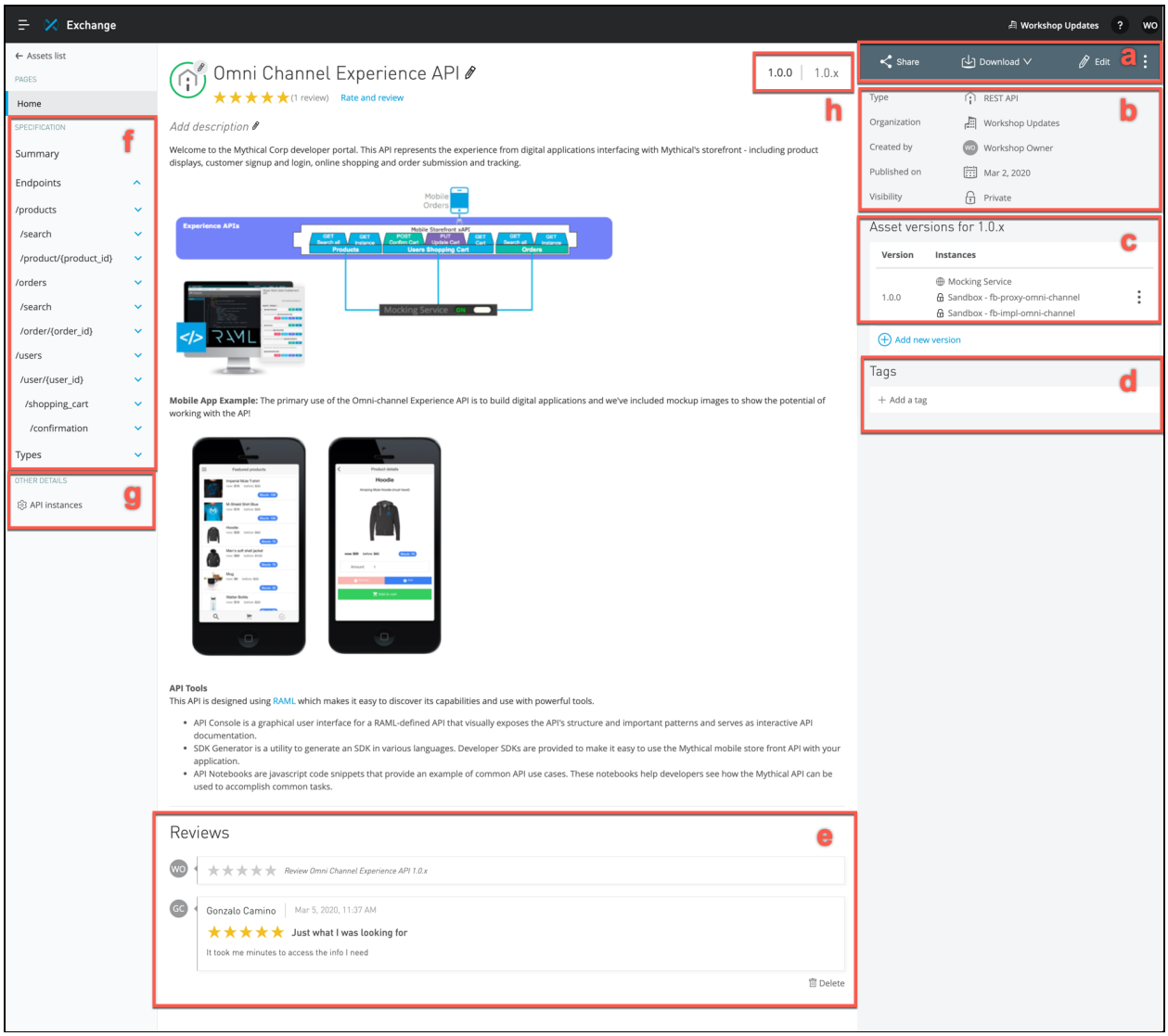


9. You should now see the page dedicated to the API definition.



Exchange provides several new features to support and encourage discovery and collaboration. First note the content in the portal to help you learn how to use the API. Let's do a quick survey of the

key features of the Exchange portal:



a. **Options Panel** - The options panel has three options.

- **Share**: This option allows you to share the API with users, users that have a role, or publish to the Public portal.
- **Download**: This option is to download the RAML Spec or Mule Plugin to use it as a Connector.
- **Edit**: This option allows editing the API portal.

b. **Information Panel** - This provides general information about the API.

c. **Assets Versions** It shows the different versions of the API. It also shows the instances implemented.

By clicking on the "ellipsis" button a new menu is displayed where you can:

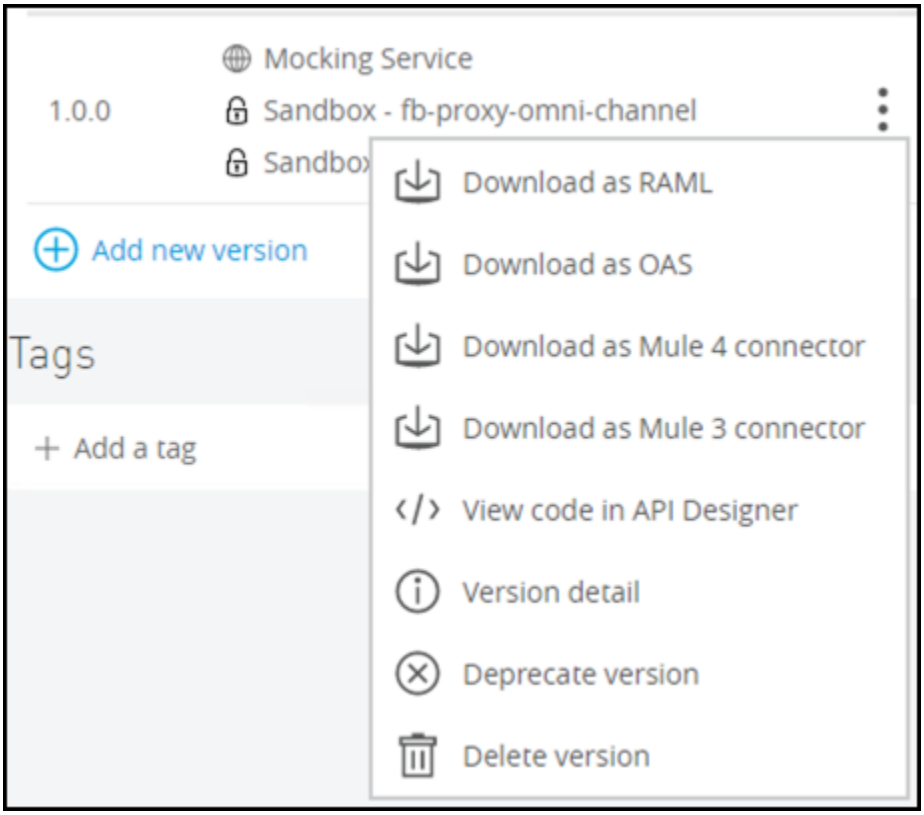
- Download as RAML: You can download the API Specification in RAML format.
- Download as OAS: You can download the API Specification in OAS format.

- Download as Mule 4 connector: You can download the Mule 4 Connector.
- Download as Mule 3 connector: You can donwload the Mule 3 Connector.
- View code in API Designer: It shows the API Spec in flow Designer.
- Version Detail: You can see the dependencies of that API.
- Deprecate version: Put a deprecated mark to this version.

- Delete version: Remove the version from exchange.

Much of this information is automatically generated during the design and development of the API.

Ellipsis Menu



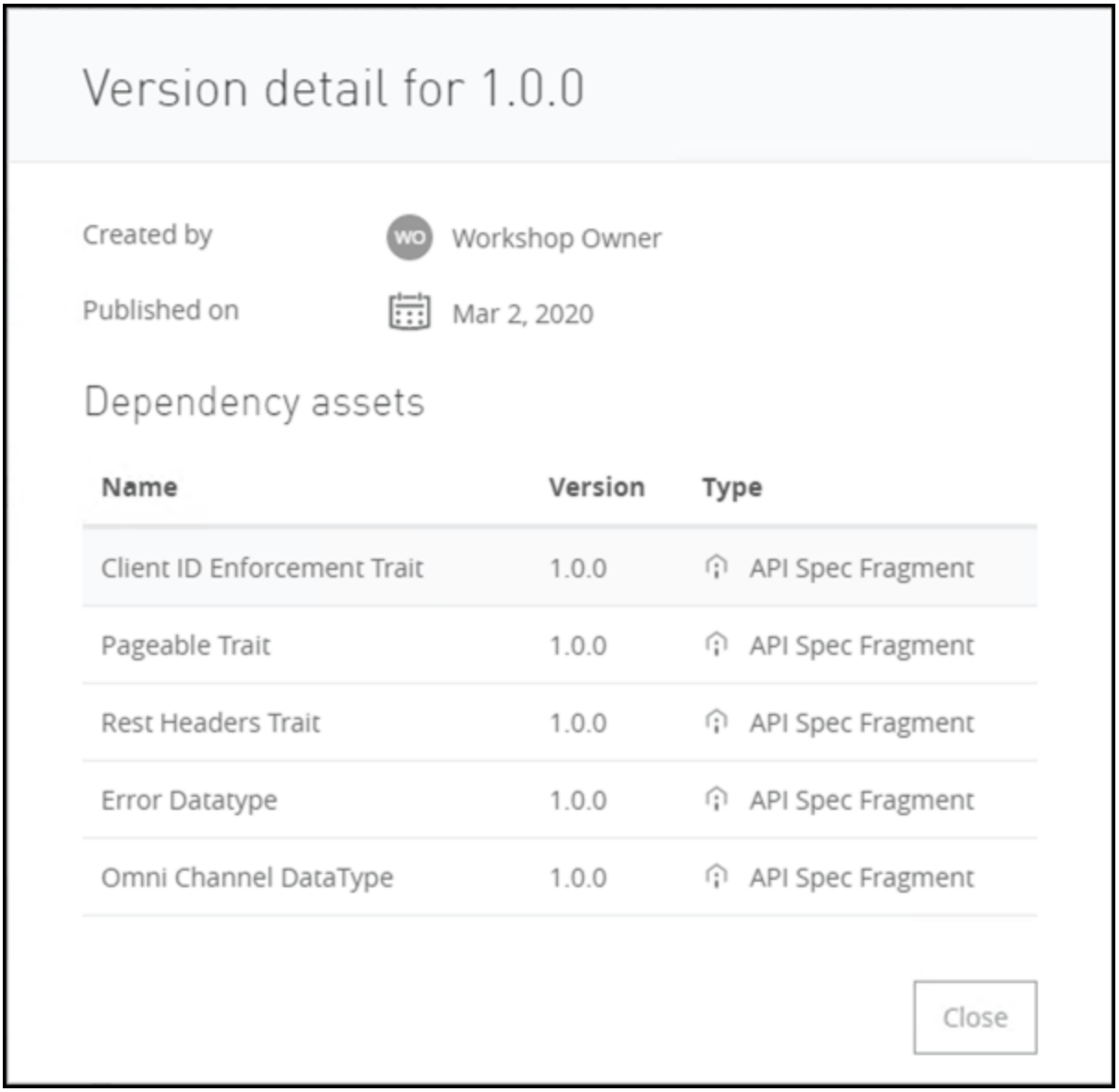
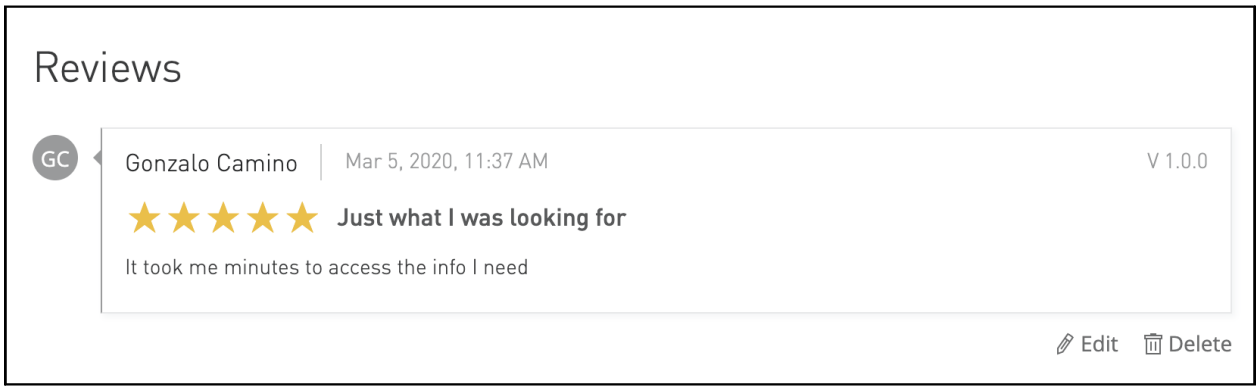


Figure 1. Version Detail

d. **Tags** - *Tags* make it easier to search and discover APIs. Rather than relying on the title or description of the API, tags allow for you to create categories of APIs related to each other by *tags*.

e. **Ratings and Reviews** - New to Exchange, API users can review and comment on an API. This enables collaboration with the API owner and the ability to share experiences using the API that might

help other API consumers.



f. **API Summary** - This is the API specification. API resources can be seen here. You can click on each resource and view the documentation.

g. **API Instances** - This gives you the opportunity to edit the API Implementation endpoints. You can also add other implementations.

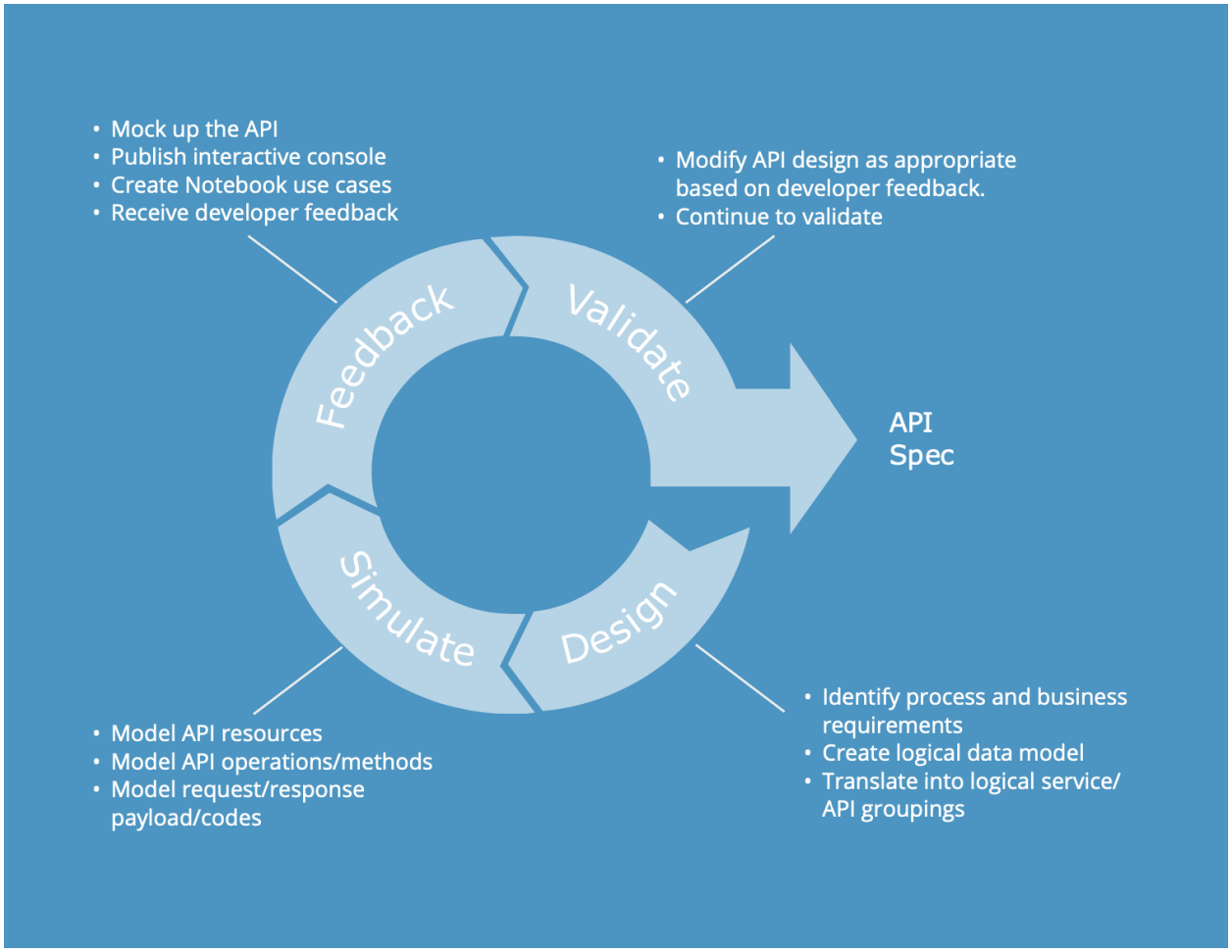
h. **Version** - You can select and see the portal of each API Version.

Let's explore how to design an API in the next lab.

Please proceed to Lab 2

Take me to the TOP of Lab 1

Lab 2: Design your API

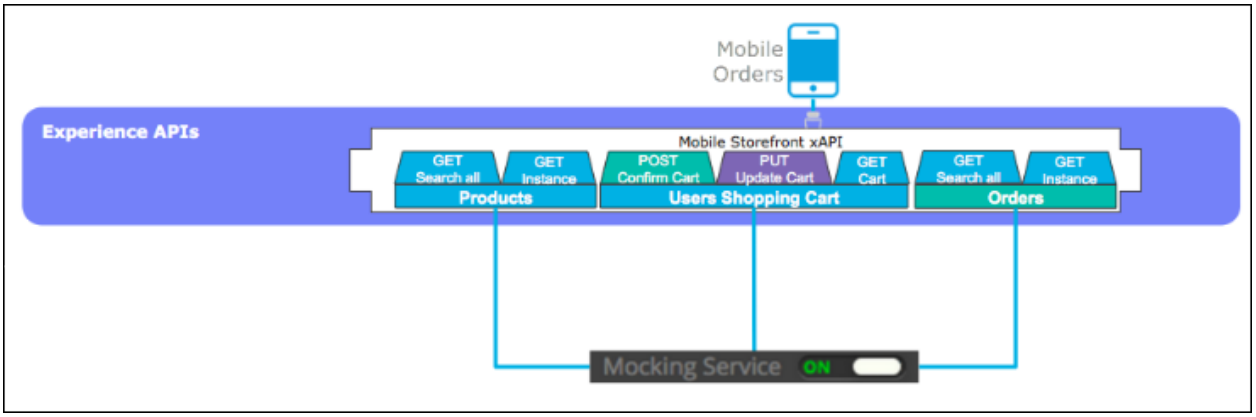


Overview

In the first lab of this module we learned how to discover the Omni Channel Experience API published in Anypoint Exchange. But of course, somebody had to design this API in the first place. For the purposes of this exercise, let's ignore the fact that the Omni Channel Experience API already exists. Instead, let's take on the role of the API designer and explore how to design the experience API using a **Design First** approach with MuleSoft's API-led Connectivity methodology. The goal of our *Design First* approach is to be able to design APIs that are easy to use for their intended target audience.

Anypoint Platform provides first-class tooling to support the needs of the API Designer. Anypoint Design Center provides a robust RAMl editor for designing the specifications of the API and standing up a "mock" service to accelerate development efforts. It is also tightly integrated with Anypoint Exchange to simplify the process of discovering and referencing other API design-oriented assets like common data types, traits, schemas, and data examples.

For the purposes of this workshop we will design new APIs using REST. RESTful APIs are very popular right now and support a variety of patterns that help solve many reliability, scalability and integration challenges you might face. We will use an API-first design approach using the RESTful API Modeling Language (RAML) standard. We will follow REST best practices to promote adoption to your consumers. This lab will also introduce the main concepts of resource oriented design and will show how RAML is used to bring APIs to life. For more information on RAML, please see: www.raml.org.



In this lab, we will start designing a new REST API and test it before implementing it. We will use Anypoint Platform's **Design Center** to design the API. We will then use **Anypoint Exchange** to search and find data definitions and examples published across the organization that enable us to standardize on common data elements. Lastly, we will utilize the **Mocking Service** to unlock resource dependencies of your API design. This significantly cuts down the time spent building the omni-channel application by turning the RAML design over to the omni-channel developers immediately and adopting a rapid prototyping style. Developers can utilize the mocked up API capabilities to produce a "working" application that calls the Mocking Service.

Understanding Resource Oriented Design (ROD)

Designing REST interfaces is a bit of a paradigm change compared to SOAP services. SOAP services design was thinking in actions and methods whereas ROD wants you to think along the lines of your business entities that should be modeled as resources. And resources might be everything from a document, a video, your favorite device or an order process.

Not long ago we modeled interfaces by making Java methods externally available, having action oriented or (insert your programming language) method naming, e.g. getProducts(String name) or the equivalent getter method setProduct(Product product).

Now there is a maturity model that measures the alignment of your Web APIs with the REST architecture style. Martin Fowler provides an excellent explanation of Richardson's model.

Reaching level 2 and maybe 3 is desirable and a ROD targets to achieve that. ROD will help you create stable, mature and scalable APIs that are equipped to evolve nicely, even within Use Cases you can't foresee currently.

The quintessence on ROD is:

1. Use standard methods (e.g. HTTP GET, PUT, POST, DELETE, HEAD) and notice their specifications - e.g. GET, HEAD, DELETE and PUT are idempotent methods
2. Use standard response codes for success and error; use existing patterns e.g. not every successful call should return a 200 OK - often a 201 Created is appropriate after a POST /products or a 202 Accepted when a long running task was started by the server
3. Use standard header parameters
4. Design and craft resources in **your** business language - here is where RAML comes into play
5. **Interlink** as many of your resources as you can; e.g. when you return a collection of products, have every product in the result carry its own link so the client can easily access it

Collection Resources

Typically resources are designed as either collection or instance resources. Collection resources hold a collection of resources that can be queried, ordered, extended and most importantly linked.

Let's take a typical example of a product resource. Most likely there are several of them and you want to be able to create new products also.

By designing a **/products** resource a list of products is available. Typically the methods

- POST (create new list instance)
- GET (get the list)

are available on the list.

A further abstraction is to move the list query onto a **/products/search** resource that one can then nicely scale independently (e.g. by caching) and offer all the tailored search functionality needed like specific query parameters.

Instance Resources

Instance resources contain the details of a specific resource, i.e. the instantiation of a resource. For our product example that might include a product id and the name of the product. Highly valuable are then links to other products that might be part of that product.

Instance resources must be uniquely identify-able and typically look like this **/products/{instance_id}**. Typically the methods

- GET (retrieve product information)
- PUT (update product information)
- DELETE (delete product)

are available on the product resource.

Now we understand resources let's explore how we can work with the resources.

Creating an Instance Resource

An instance resource is created by POST'ing against a collection resource. There are multiple ways to implement that. One could argue that a POST request against the list resource creating an instance should already carry all instance details in the body and then return the created resource.

This is one way, sadly it is not the most reliable when something goes wrong and the response fails. Then a new POST needs to be issued which creates a new instance on the server side.

A more reliable and recommended pattern is the following. First a client issues a POST on a list resource. But instead of the server returning a complete instance already, it returns the link to the instance in the Location header carried with a 201 Created response. Now the client can poll the instance resource using an idempotent GET or PUT until it is successful. In case something went wrong the client can simply continue polling without creating a new instance every time as a get request is idempotent by specification.

Step 1: Review the E-Commerce Application Requirements

You have a new requirement for a new e-commerce application that customers can use for shopping and ordering products through a web portal or mobile device. The e-commerce application has the following requirements we are asked to fulfill. It is written in the language of the e-commerce developer and we will derive the RAML specification from that.

Authentication/Registration:

- Login
  - Mail
  - Password
  - Returns all user information
- Signup
  - Name
  - Mail
  - Password
  - Returns all user information

Functionalities

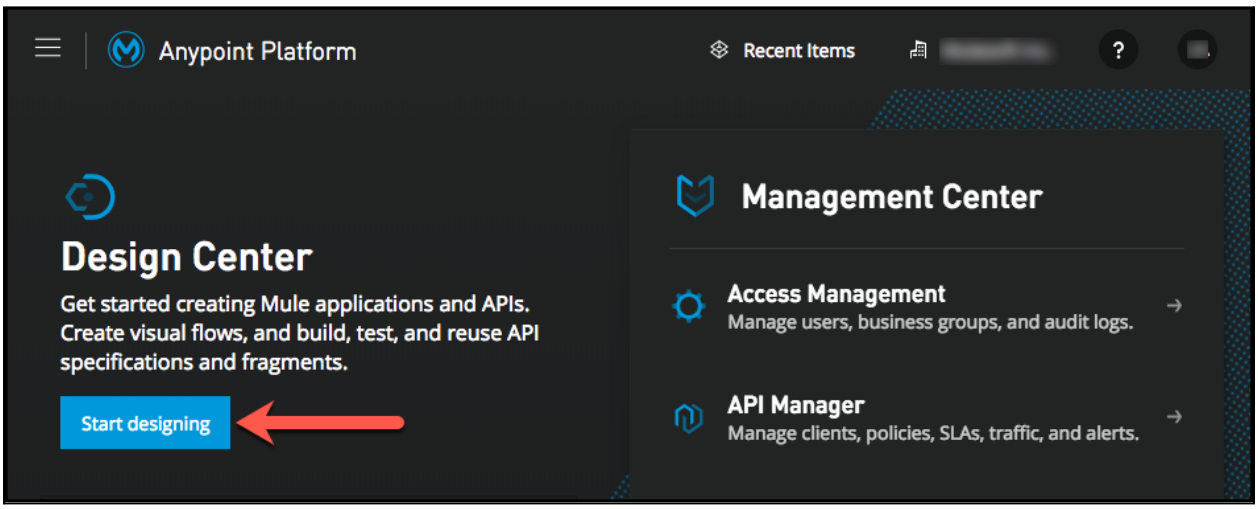
- Products:
  - List all the products available
    - Name
    - Price
    - Stock
    - Id
- Orders
  - Create a new order
    - Returns order id
  - Add product to order:
    - Order id
    - Product id
    - Quantity
  - Remove item from order
  - Confirm order
    - Shipping address data
    - Credit card data
  - List orders of the user
    - Full data

Let's design these now in RAML.

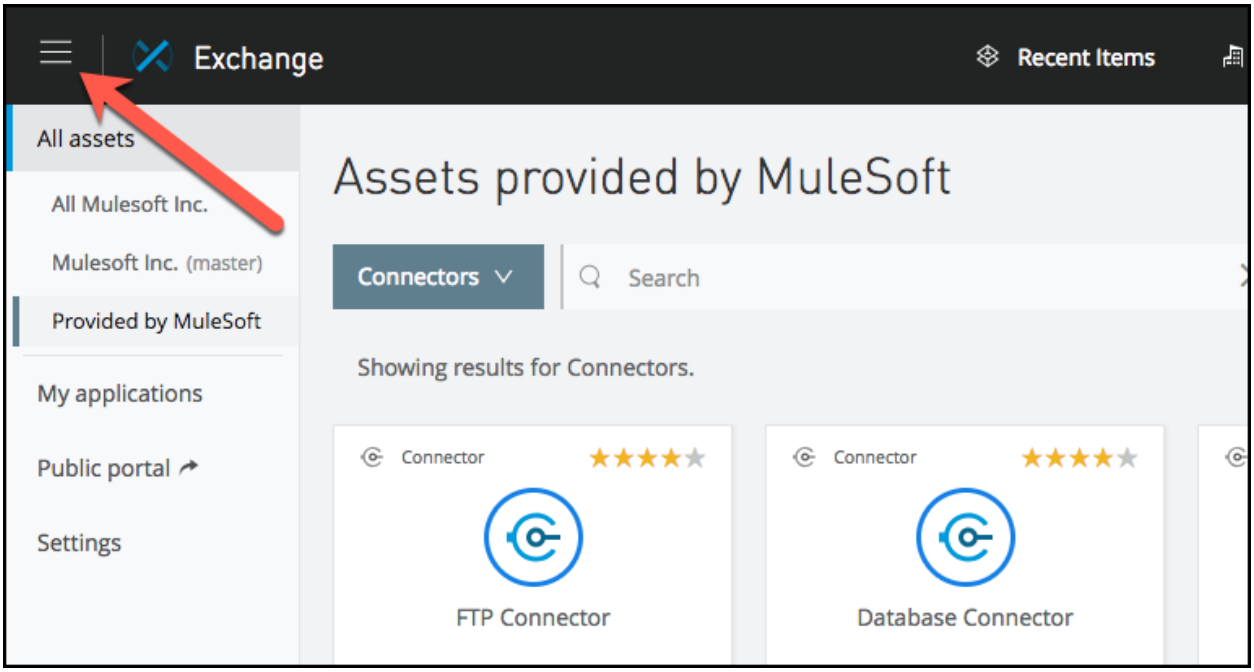
Step 2: Create the Omni Channel Experience API

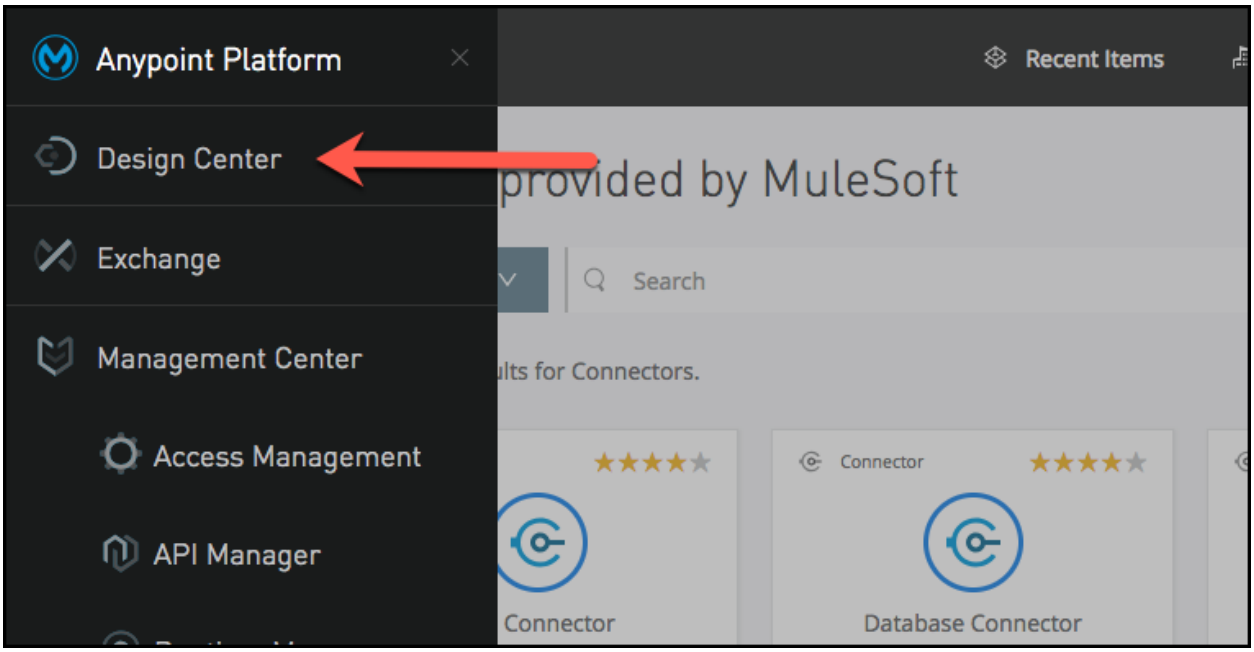
In this step, you will create an API and design it using the Anypoint Design Center.

1. You can access Anypoint Design Center from the home page of Anypoint Platform:

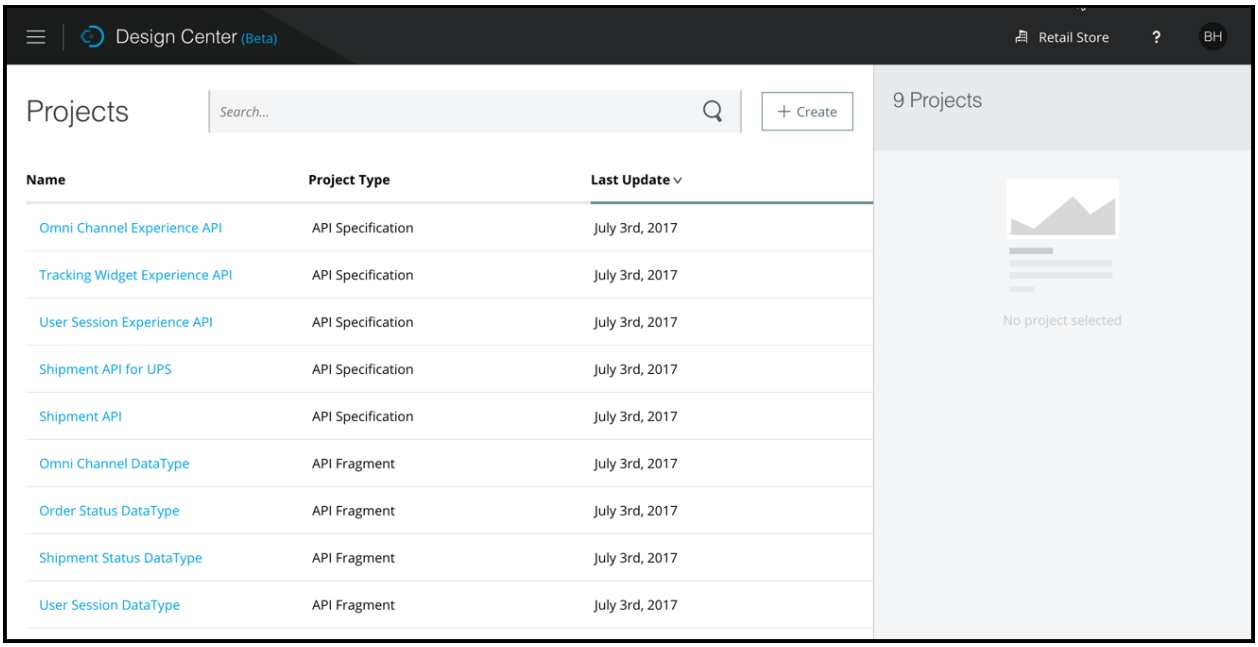


Alternatively you can use the "hamburger menu" from any page within Anypoint Platform portal to navigate to Design Center:

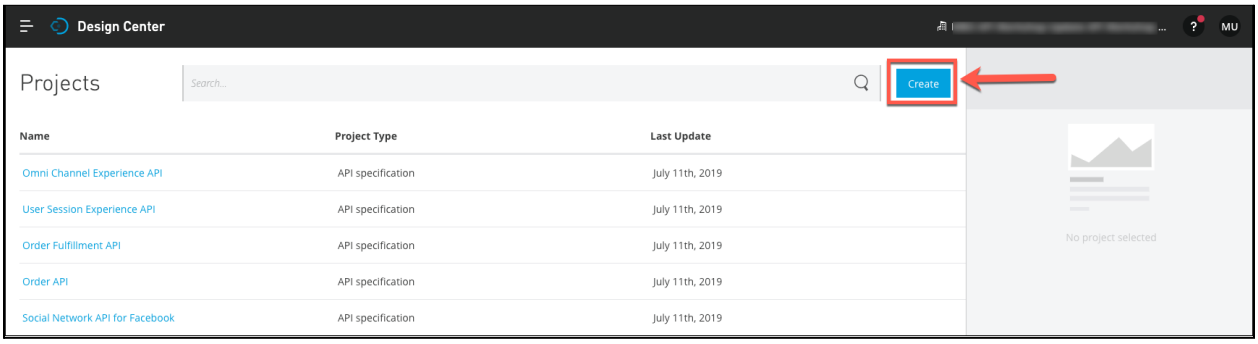

Exchange
Recent Items
All assets
All Mulesoft Inc.
Mulesoft Inc. (master)
Provided by MuleSoft
My applications
Public portal
Settings
Assets provided by MuleSoft
Connectors
Search
Showing results for Connectors.
Connector
FTP Connector
Connector
Database Connector


Anypoint Platform
Recent Items
Design Center
Exchange
Management Center
Access Management
API Manager
provided by MuleSoft
Search
ults for Connectors.
Connector
Connector
Database Connector

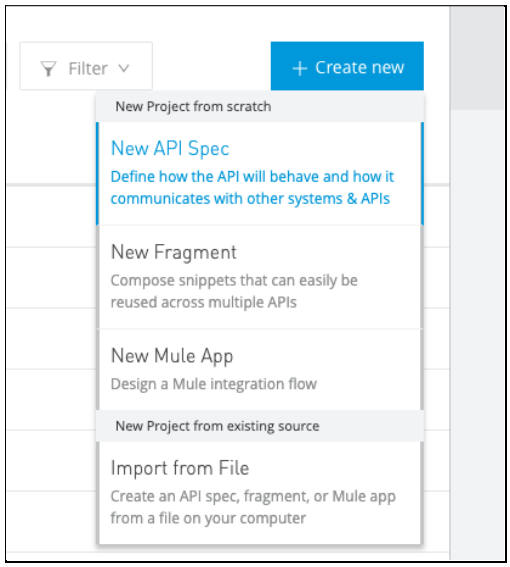
2. **Anypoint Design Center** is used to design your APIs. When you arrive at the Design Center landing page you will see a list of APIs that have already been designed by your organization.



3. To create a new API, Click on the **Create** button. Select "API Specification".

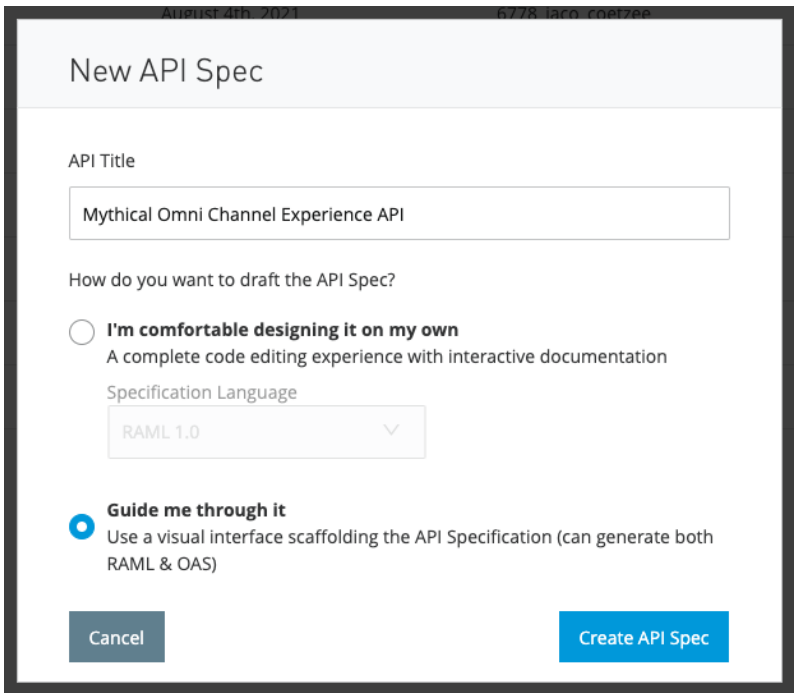


4. Select "New API Spec".

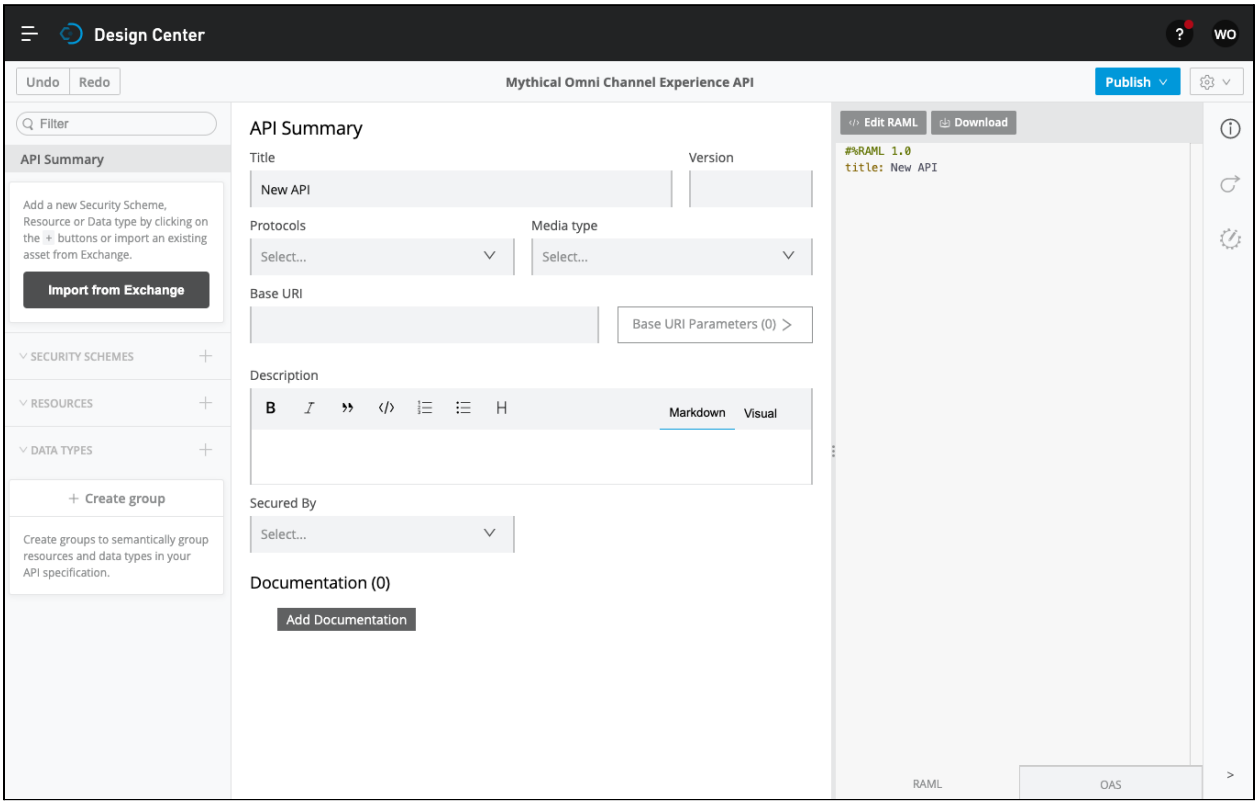


5. Set the following values in the **Add API** form:
   - API Title: **<username> - Omni Channel Experience API**.
   - **Select the *Guide me through it* radio button**.

     Note: Since we are working in a shared organization, please give your API a unique name. For the purposes of this exercise please prepend your API with your Anypoint account username.



6. Click **Create Specification**. This will create the basic structure of your RAML-based API specification. Note that for full RAML support, you should start with the first option (API Designer), which is the standard RAML-based API Designer.



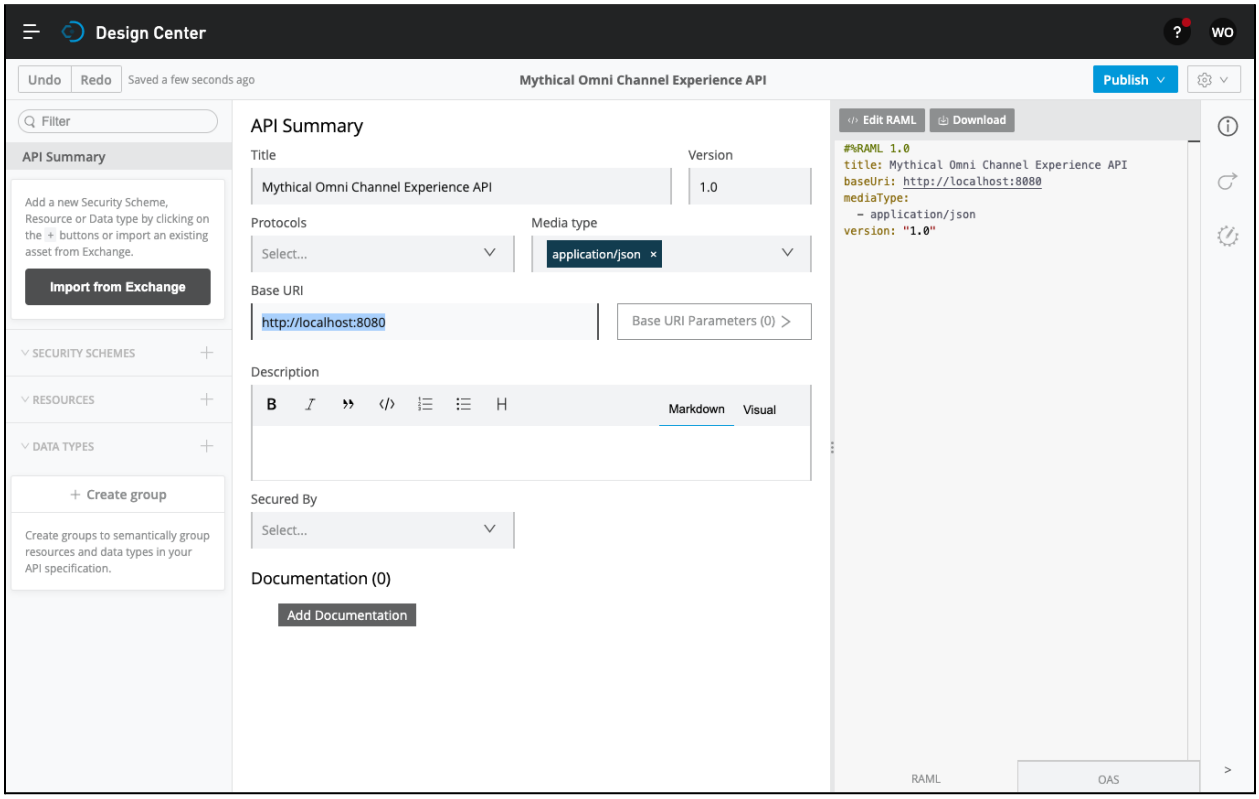
In order to provide order handling functionality for the e-commerce application, we will design a list resource in our API using RAML.

Anypoint Design Center will provide a visual API editor to start with a step-by- step tutorial to guide you through designing your API visually.

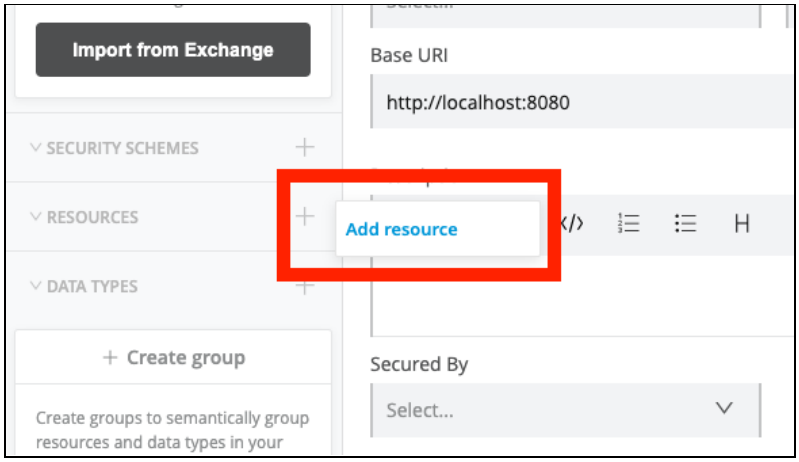
1. Let's create our API and skip the tutorial for now. Click Hide these tips. (Feel free to go through the tutorial later on).
2. Let's add some general details about the API.
   - Name the API by setting the title to **<username> Omni Channel Experience API**.

- Since this is our first design, let's designate this API as version 1.0. Set version to 1.0
- Set the media type to application/json.
- Most API specifications will designate an API endpoint Base URI. This tells the API consumer where they will access the API. Set the URI to http://localhost:8081/api

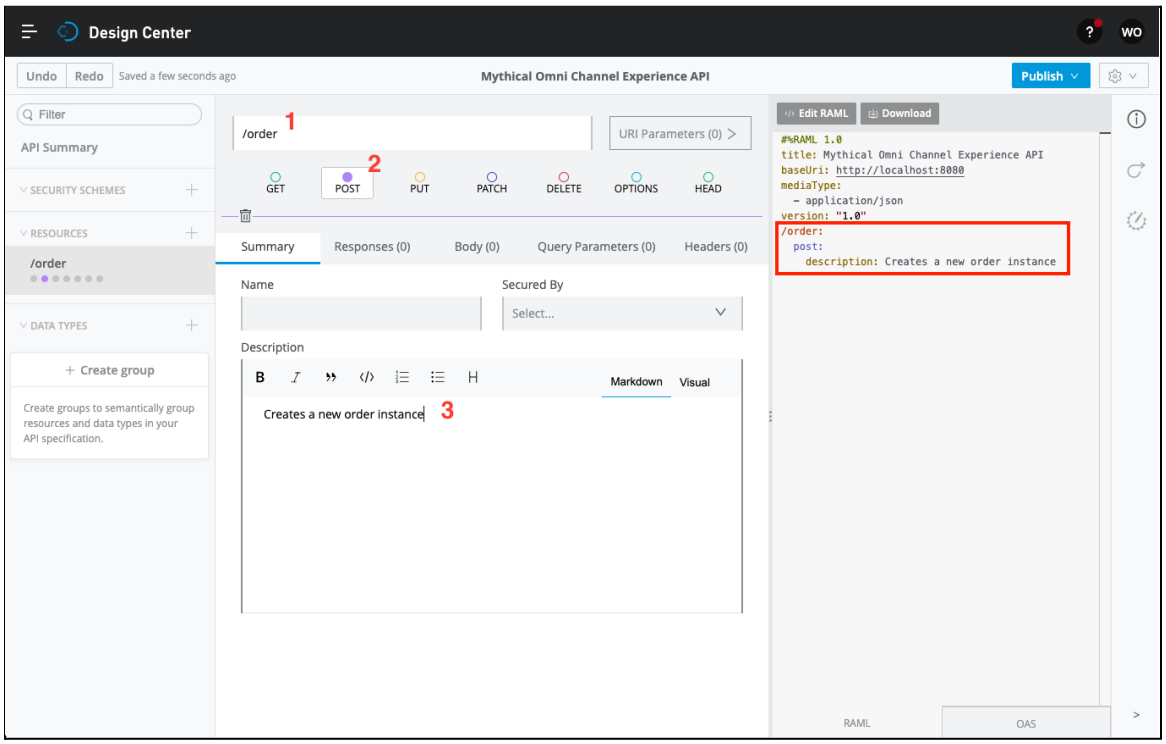
You should now see the following in your API design:



Now you need to add a resource. A resource is the definition of an entity that your API will manage. Click the **+** to the right of **Resources**, then choose **Add resource**.



3. Rename the resource from `/new-resource` to `/order`. Select the `POST` option and under documentation, give it a description: `Creates a new Order instance` as shown. Notice the generated RAML on the right.



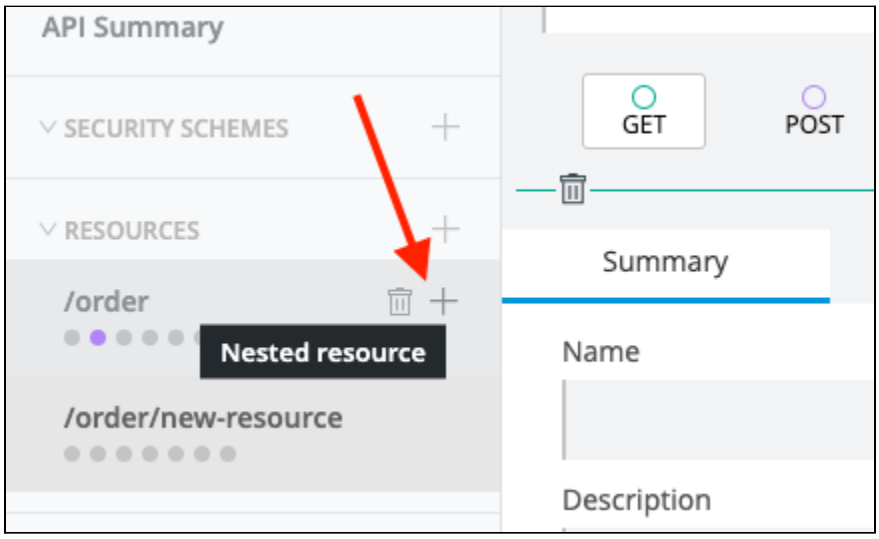
Congratulations! You have completed creation of an /order resource with a basic description.

Step 4: Create the Order Instance Resource

In the previous step we created an Order collection resource and POST method to enable the e-commerce application to create a new Order instance. What if we wanted to let the e-commerce application get the status of a single order? We need to create a new resource underneath the Order instance and use the GET method.

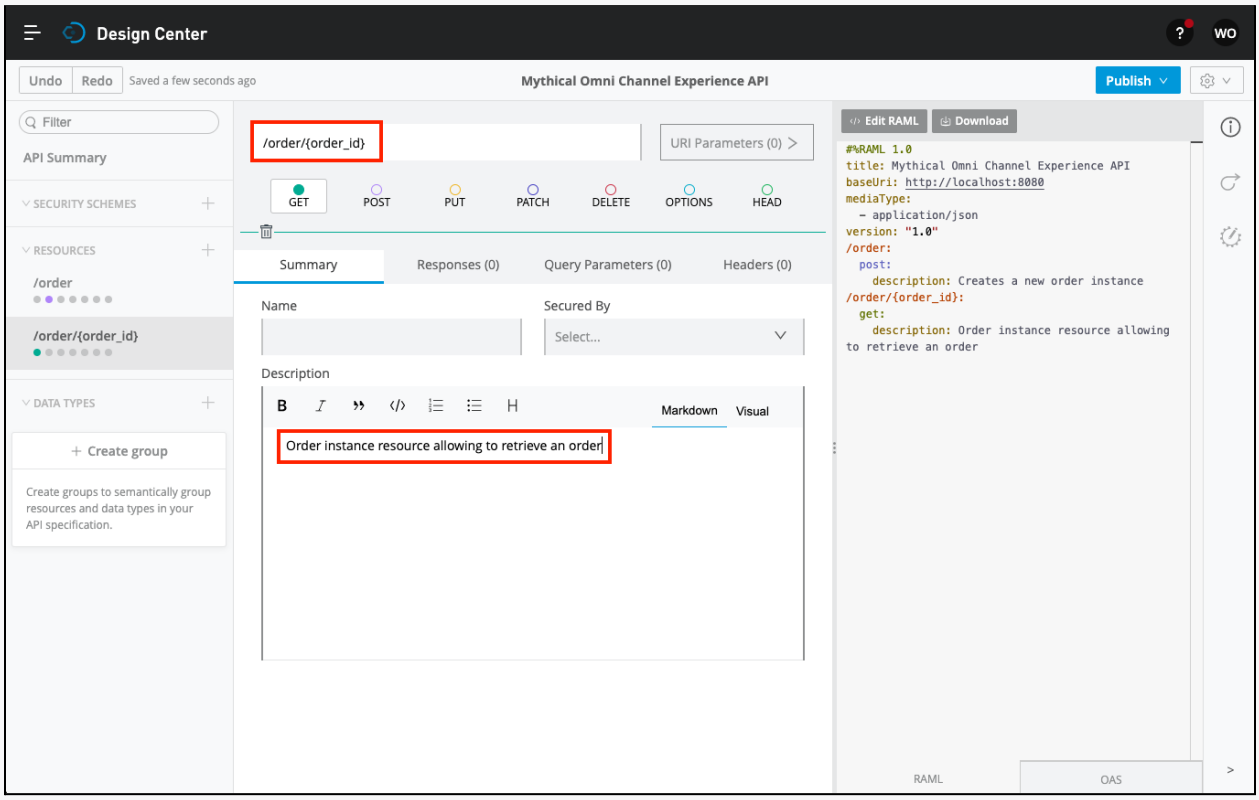
1. Hover the mouse cursor over the **/order** resource. You will notice that a `+` appears in the resource panel.

2. Press the `+` button to create a new, nested resource. Notice that a default name of "new-resource" has been created. In addition, the GET method has been selected by default.

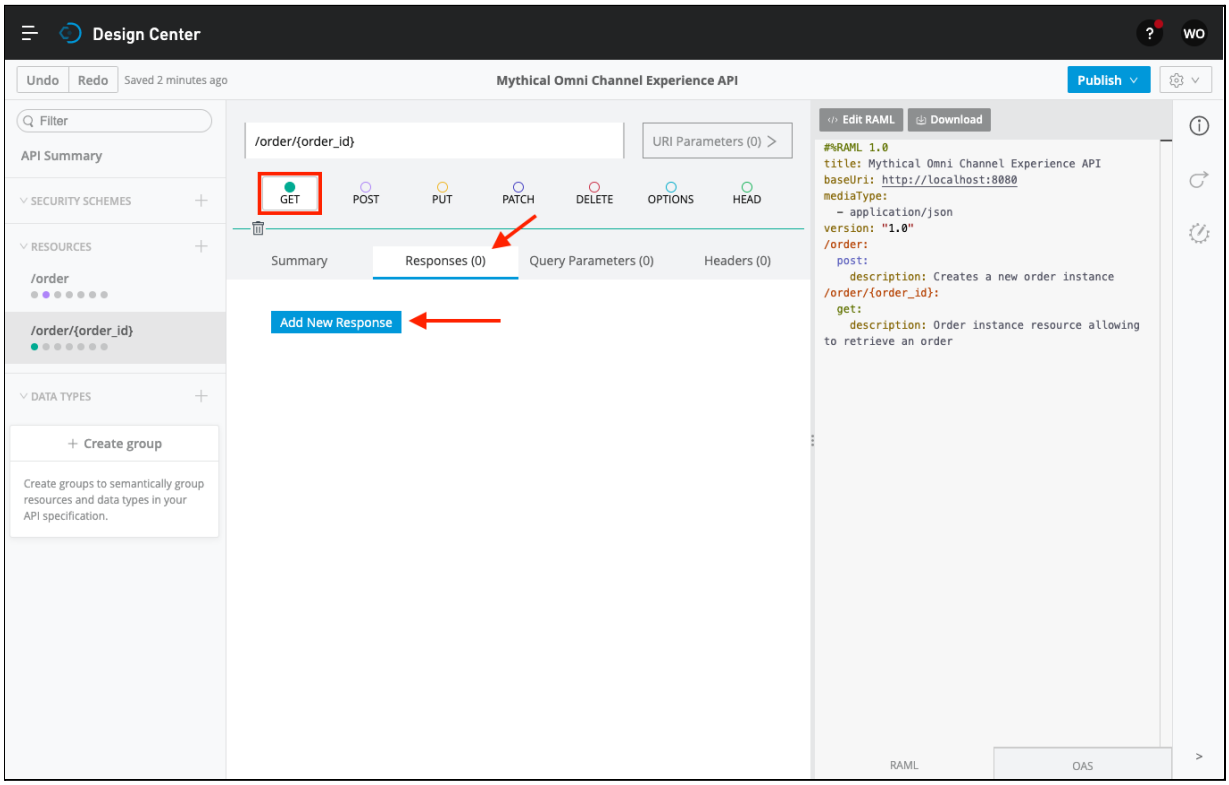


3. Rename `/order/new-resource` to `/order/{order_id}`. Curly brackets indicate dynamic variables where a value will need to be passed when called.

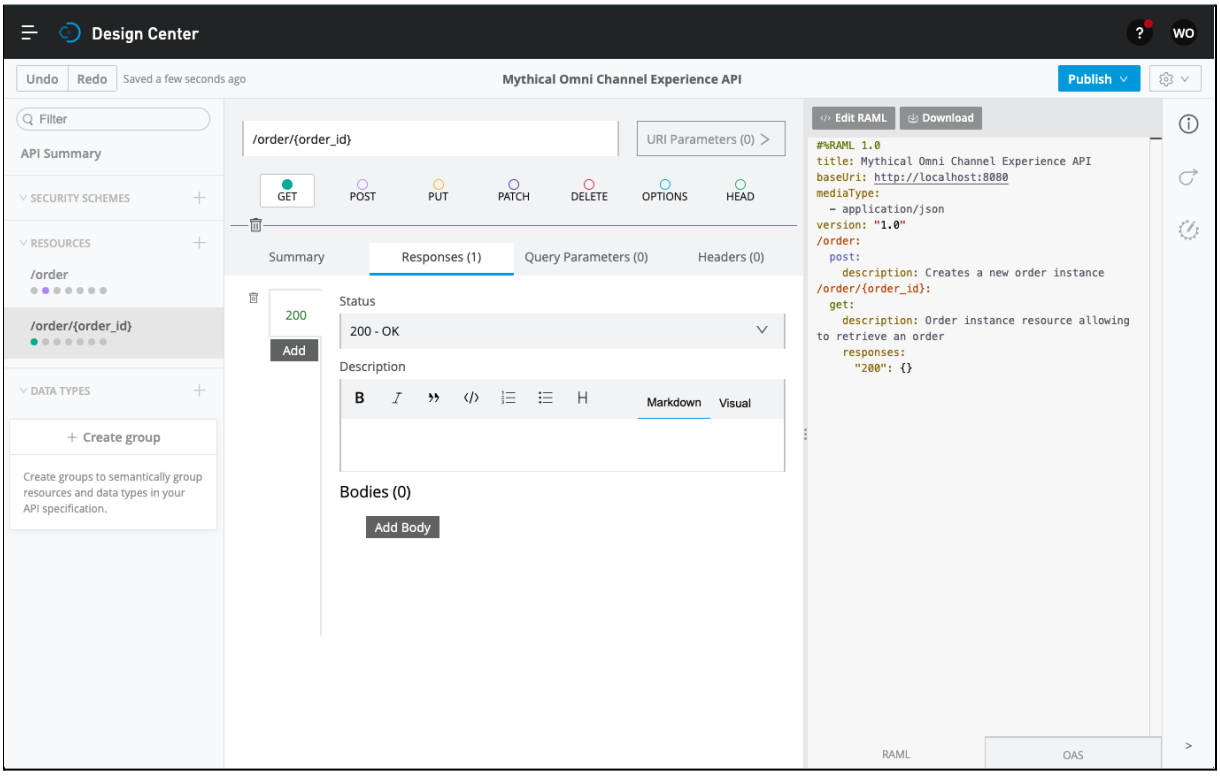
4. As we did in the previous step, enter a description: `Order instance resource allowing to retrieve an order`



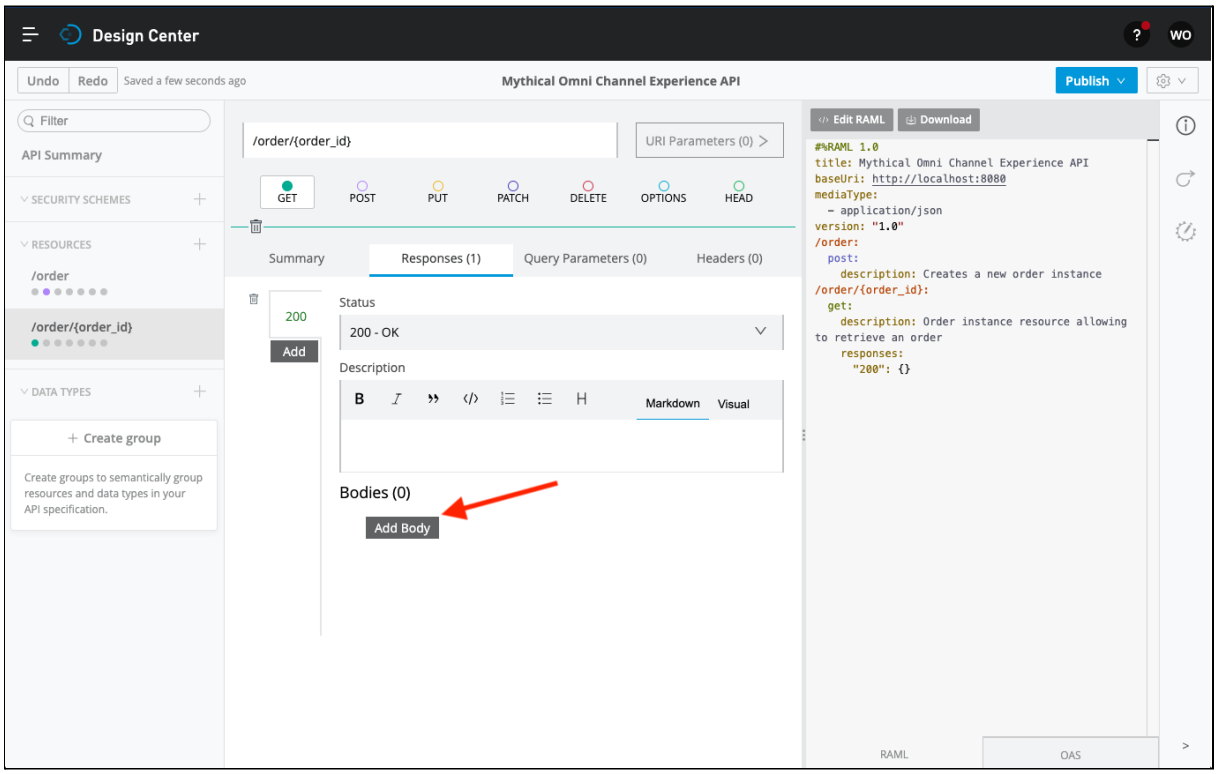
5. Under the GET method click on the **Responses** tab, then click on the `Add New Response` button.



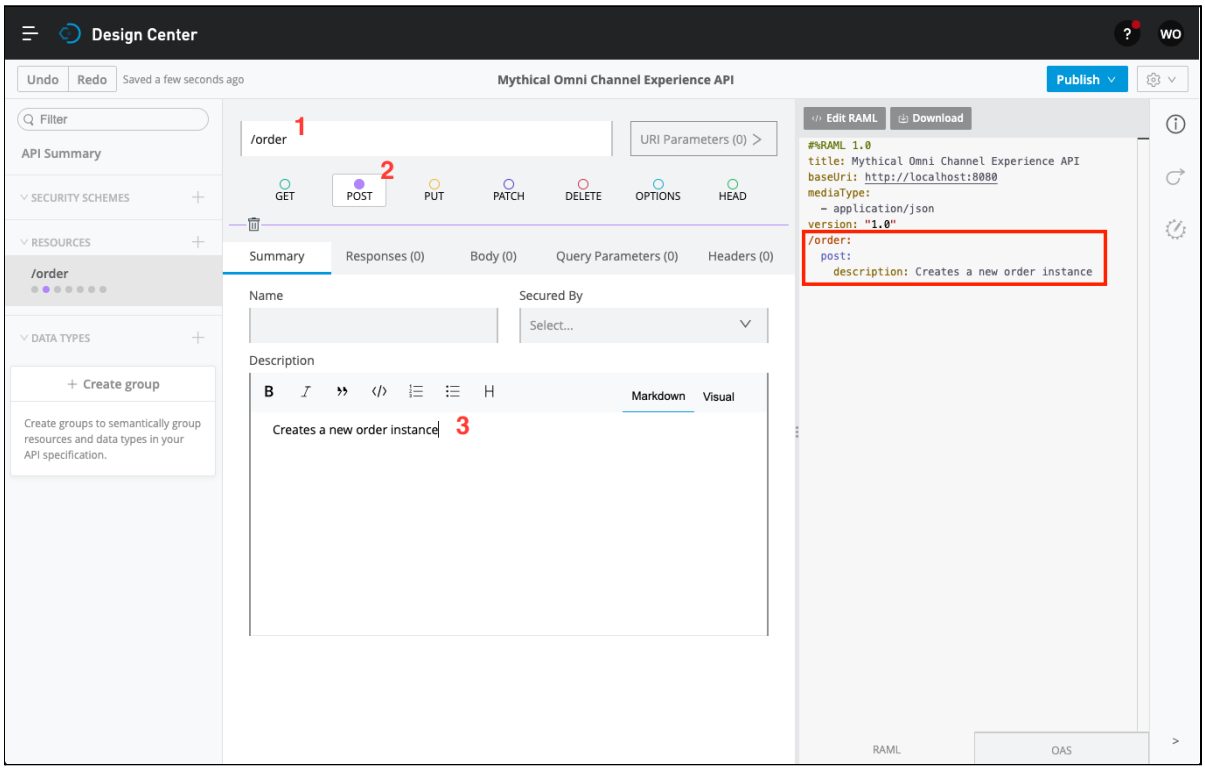
6. You should now see a response with a 200 OK response code.



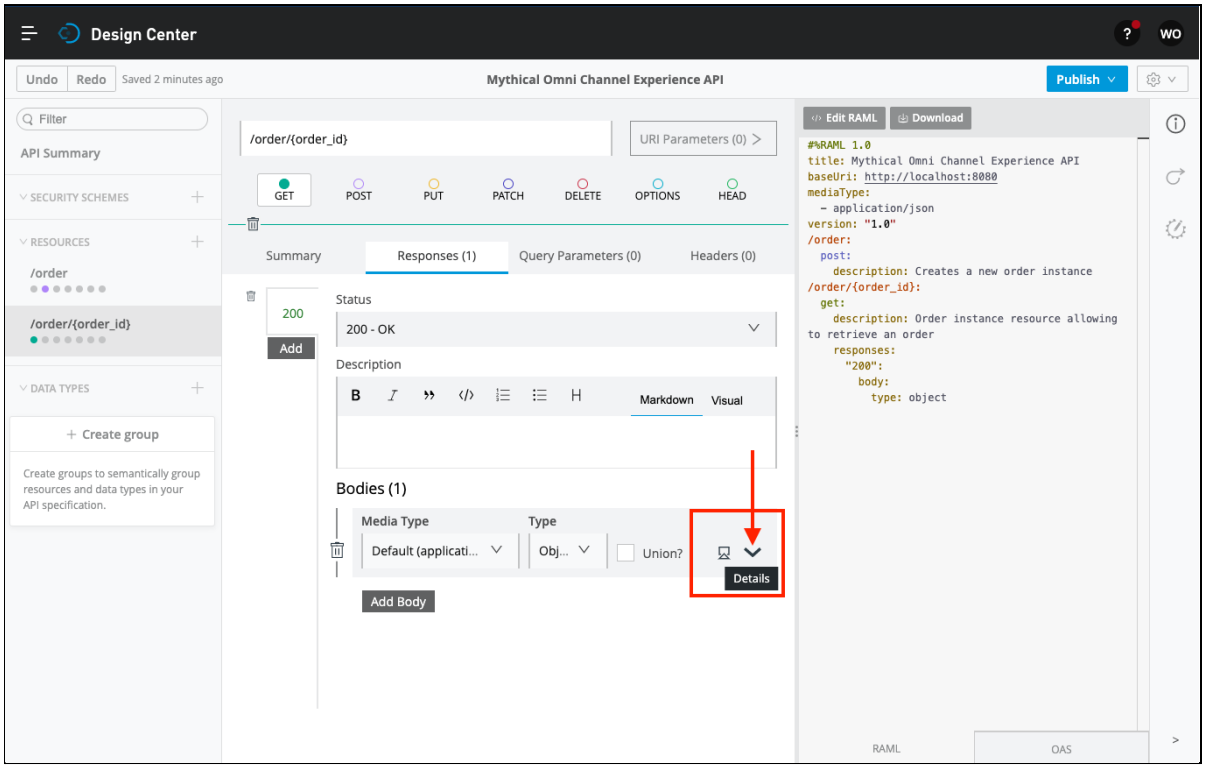
7. Under the response, click **Add Body** to add a JSON response to the GET method.



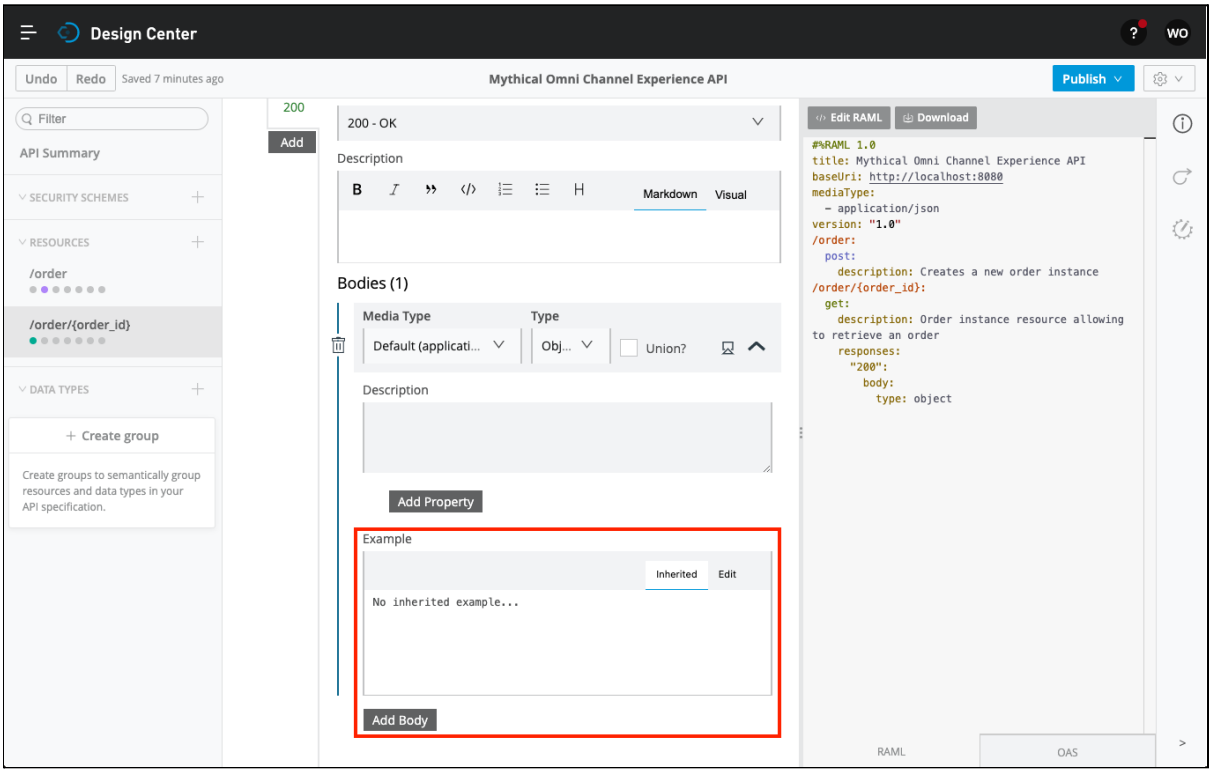
8. Notice the RAML specification on the right updated with the body.



9. Let's add an example JSON to the response. We need to expand the "bodies" section by clicking on "Details"

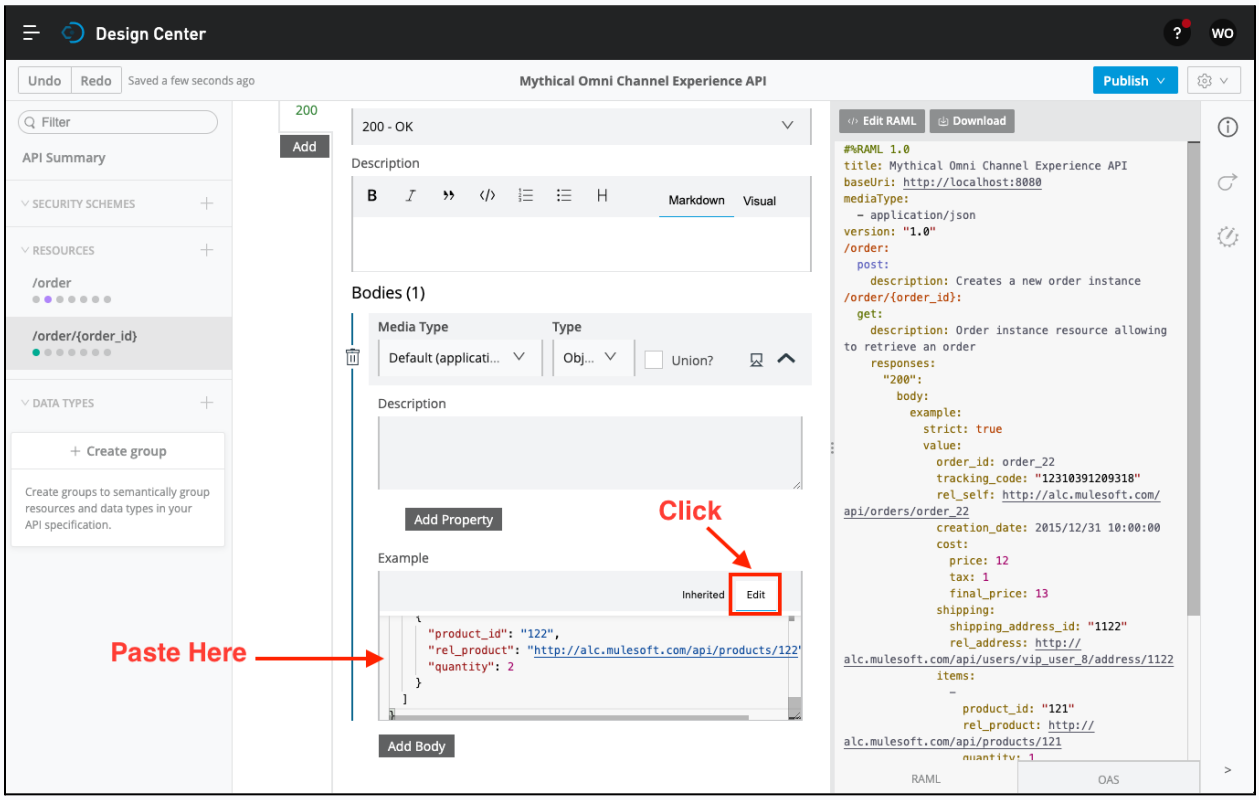


10. Once expanded, you will see the "Example" section.



Under “Example”, click the Edit tab and copy/paste the following JSON code

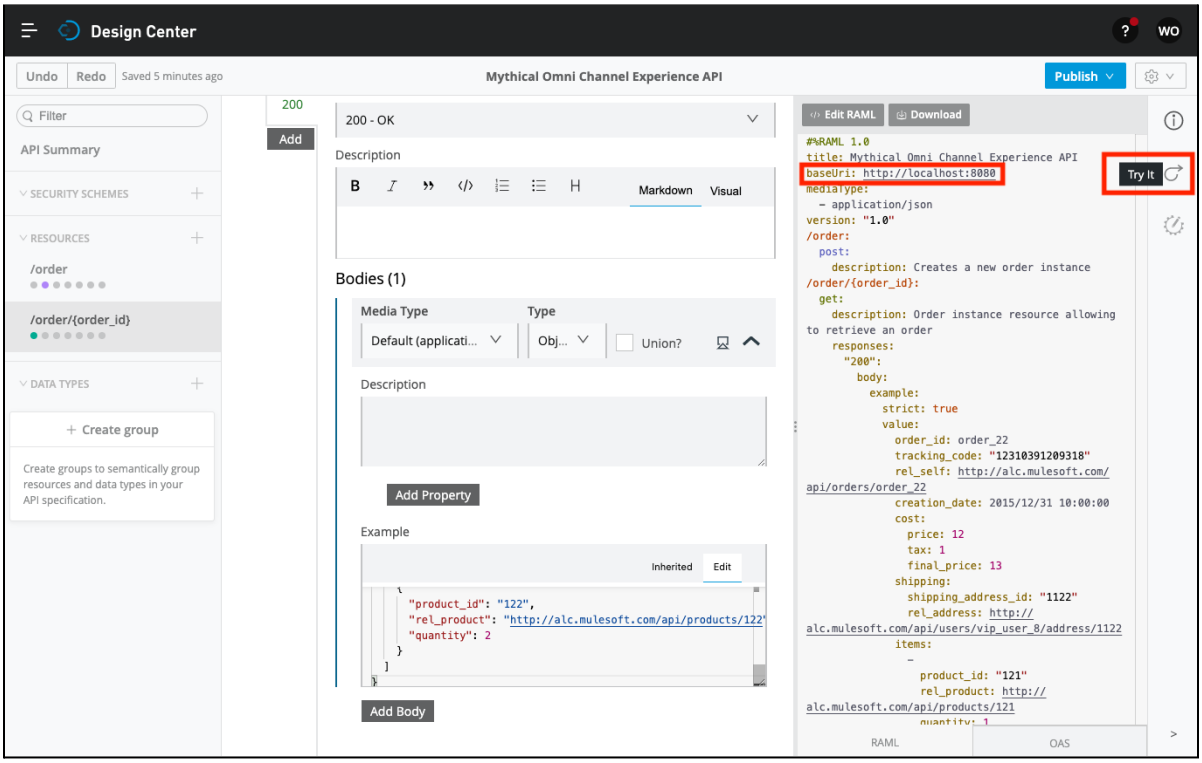
```
{
  "order_id": "order_22",
  "tracking_code": "12310391209318",
  "rel_self": "http://alc.mulesoft.com/api/orders/order_22",
  "creation_date": "2015/12/31 10:00:00",
  "cost": {
    "price": 12,
    "tax": 1,
    "final_price": 13
  },
  "shipping": {
    "shipping_address_id": "1122",
    "rel_address": "http://alc.mulesoft.com/api/users/vip_user_8/address/1122"
  },
  "items": [
    {
      "product_id": "121",
      "rel_product": "http://alc.mulesoft.com/api/products/121",
      "quantity": 1
    },
    {
      "product_id": "122",
      "rel_product": "http://alc.mulesoft.com/api/products/122",
      "quantity": 2
    }
  ]
}
```


Design Center
Undo
Redo
Saved a few seconds ago
Mythical Omni Channel Experience API
Publish
Filter
API Summary
SECURITY SCHEMES
RESOURCES
/order
/order/{order_id}
DATA TYPES
Create group
Create groups to semantically group resources and data types in your API specification.
200
Add
200 - OK
Description
Markdown
Visual
Bodies (1)
Media Type
Type
Default (applicati...
Obj...
Union?
Description
Add Property
Click
Example
Inherited
Edit
Paste Here
"product_id": "122",
"rel_product": "http://alc.mulesoft.com/api/products/122"
"quantity": 2
Add Body
Edit RAML
Download
#%RAML 1.0
title: Mythical Omni Channel Experience API
baseUri: http://localhost:8080
mediaType:
- application/json
version: "1.0"
/order:
post:
description: Creates a new order instance
/order/{order_id}:
get:
description: Order instance resource allowing to retrieve an order
responses:
"200":
body:
example:
strict: true
value:
order_id: order_22
tracking_code: "12310391209318"
rel_self: http://alc.mulesoft.com/api/orders/order_22
creation_date: 2015/12/31 10:00:00
cost:
price: 12
tax: 1
final_price: 13
shipping:
shipping_address_id: "1122"
rel_address: http://alc.mulesoft.com/api/users/vip_user_8/address/1122
items:
-
product_id: "121"
rel_product: http://alc.mulesoft.com/api/products/121
RAML
OAS

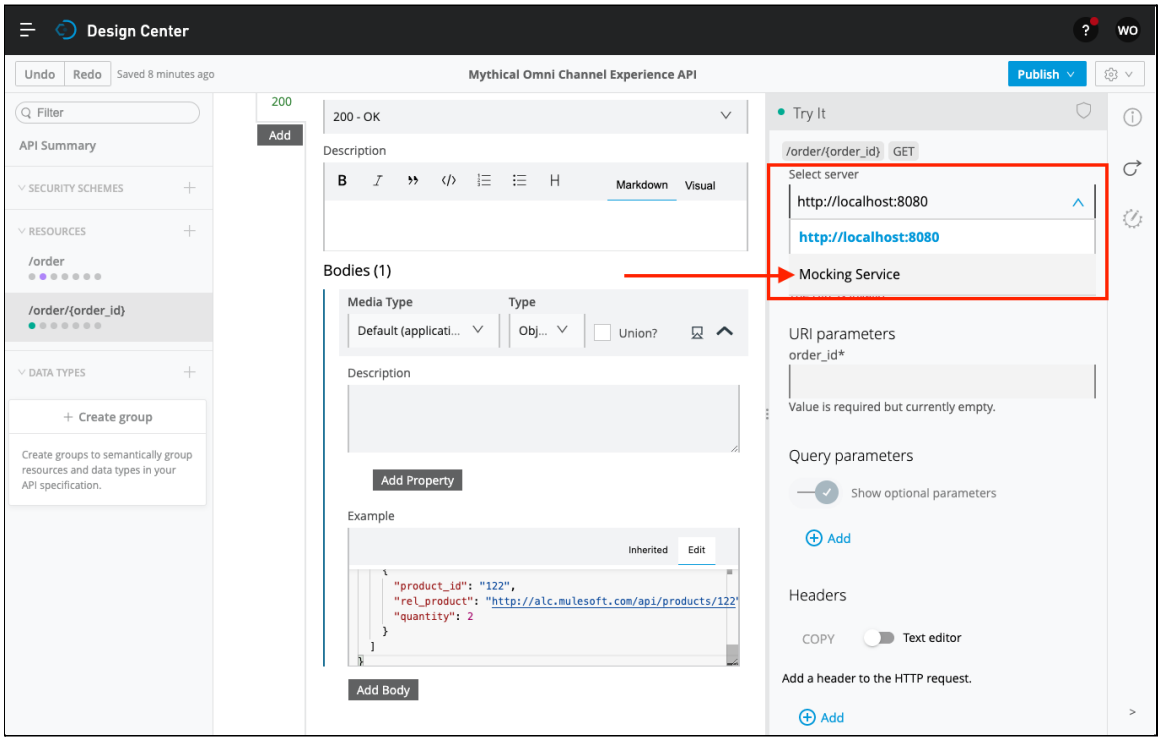
Step 5: Enable the Mocking Service

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 its consumer to finalize the contract.

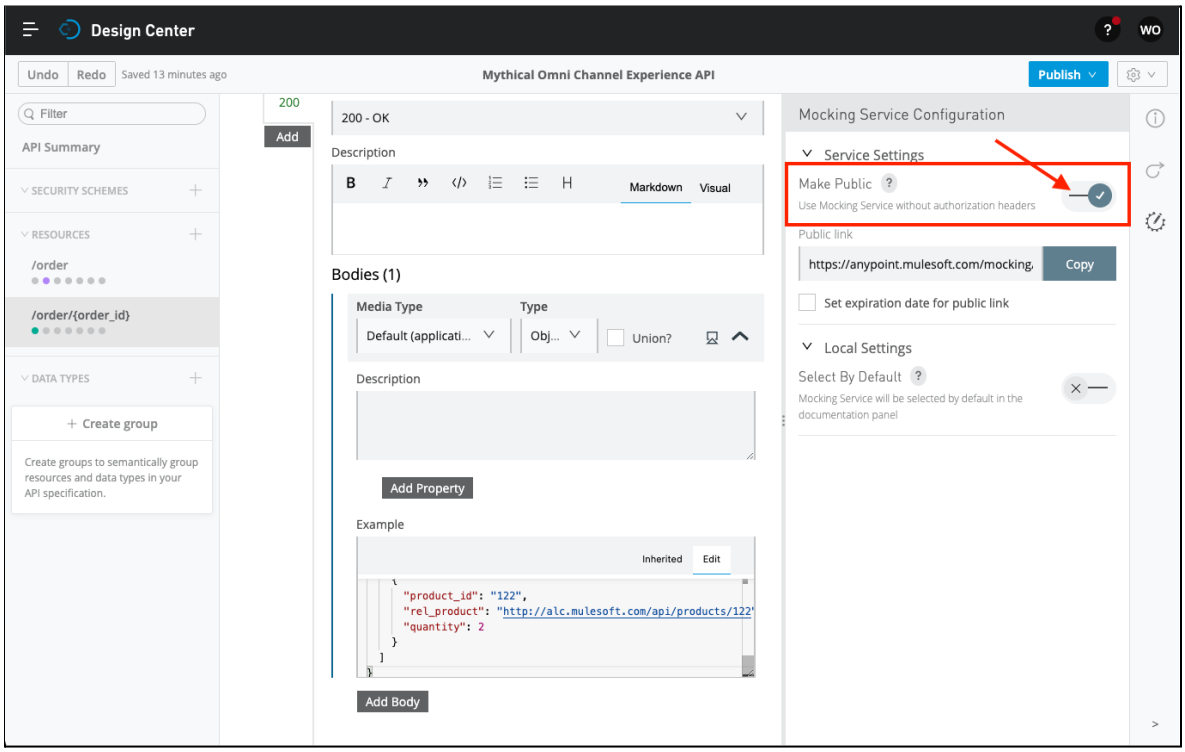
1. Look at the `baseUri` in the RAML Viewer. Notice it is set based on what we entered in the API Summary.
2. Enable the mocking service by selecting the “Try it” button on the right hand side of the RAML viewer:



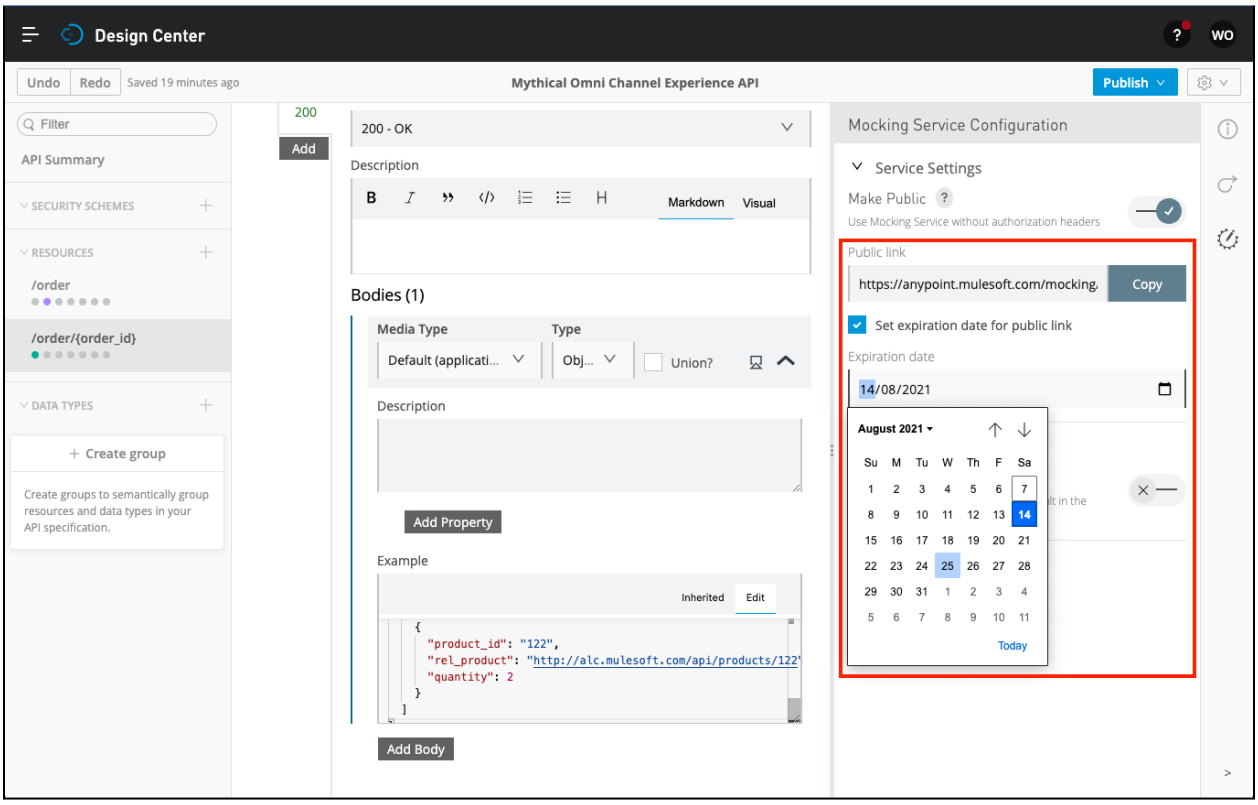
Select the **Mocking Service** in the Select server section



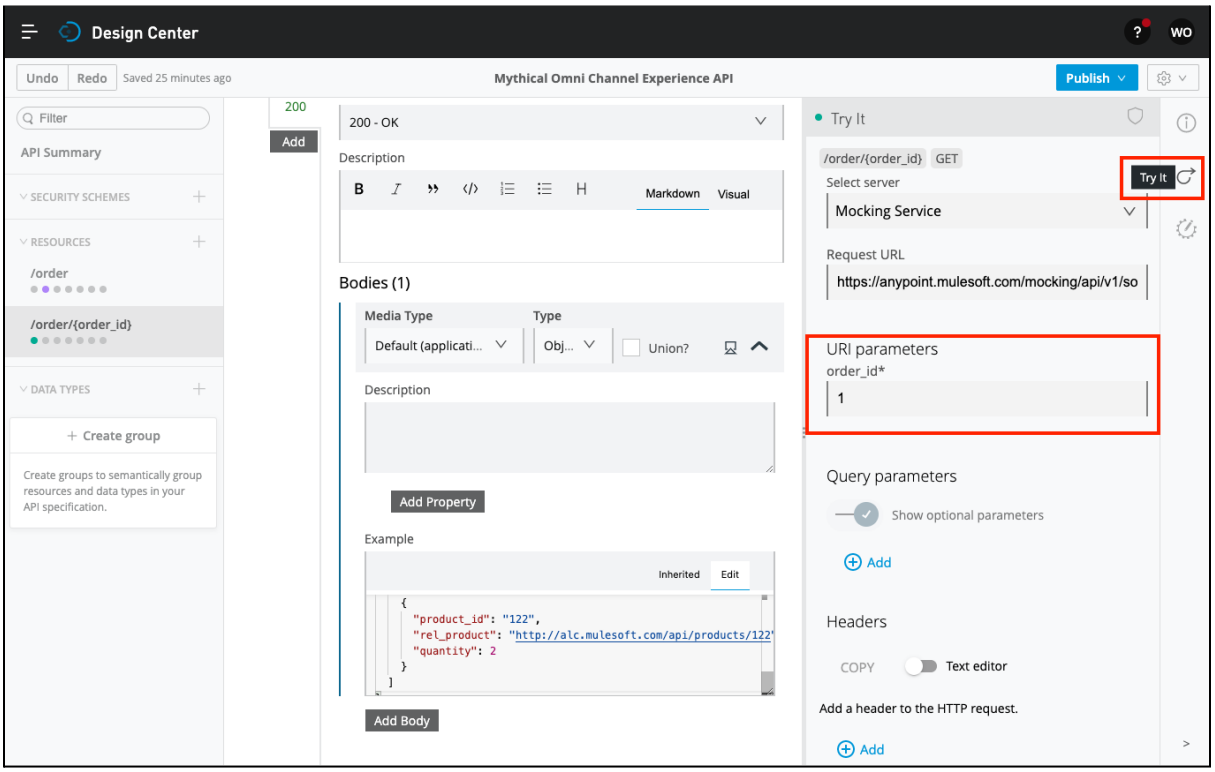
Next press on the Mocking Service Configuration where we will relax the authorization requirements in order to publicly access the Mocking Service by sliding the **Make Public** switch to ON.



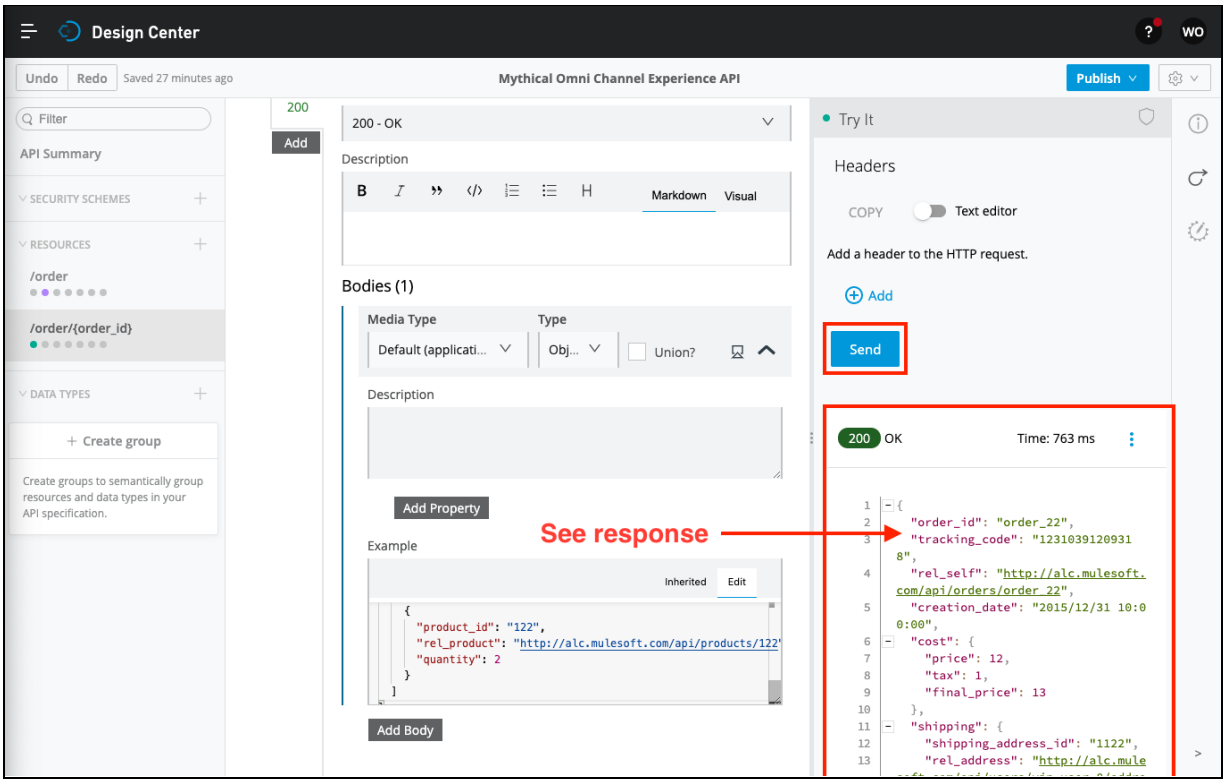
After a few seconds, the mocking service will start, creating a new URL for the baseUri. You can also set the expiration date for the public link. That way you don't leave the mocking service open unnecessarily.



Once the Mocking service is enabled, you can click **Try It** at the right hand of the RAML viewer to submit an API request and receive the example response you defined. Let's try to retrieve an Order using the Mock service. Update the **URI parameters** with **1**

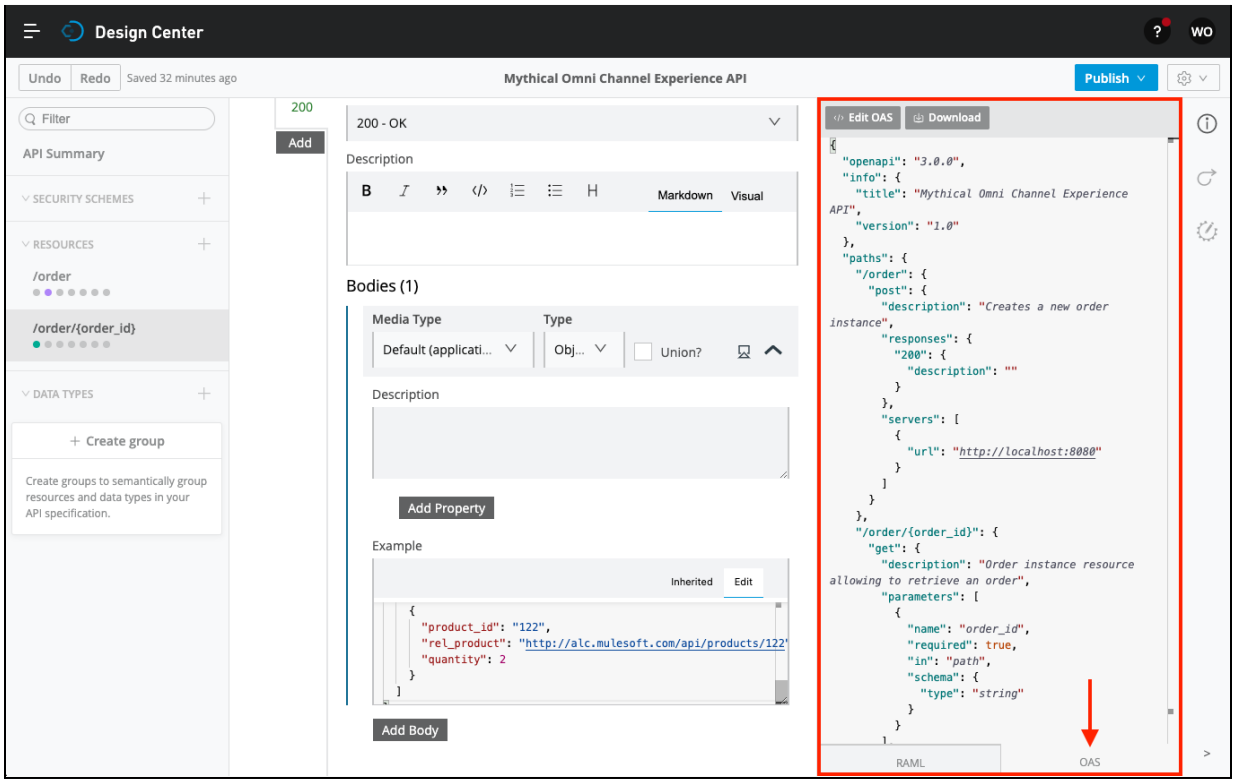


3. Click **Send** and you should see a 200 response with the Order record in a JSON format that adheres to the example you set in the API definition **example JSON payload data** that you configured earlier.

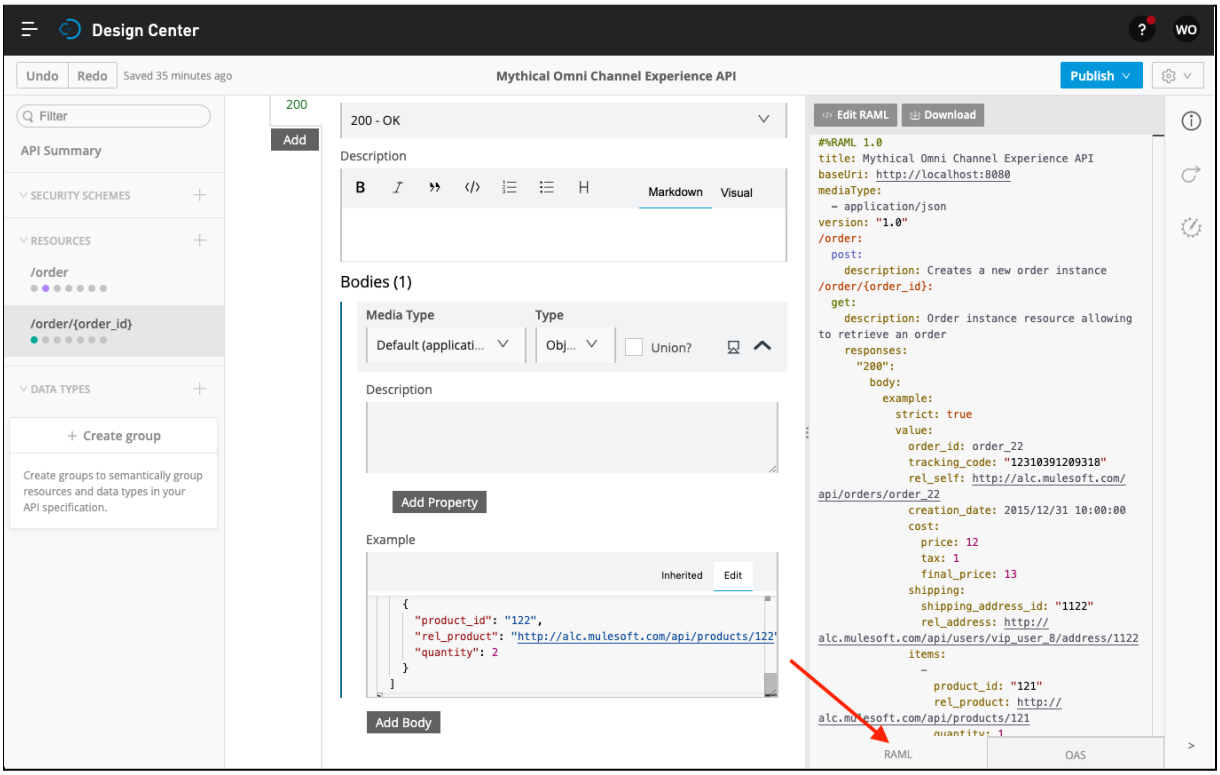

Design Center
Undo
Redo
Saved 27 minutes ago
Mythical Omni Channel Experience API
Publish
Filter
API Summary
SECURITY SCHEMES
RESOURCES
/order
/order/{order_id}
DATA TYPES
Create group
Create groups to semantically group resources and data types in your API specification.
200
Add
200 - OK
Description
Markdown
Visual
Bodies (1)
Media Type
Type
Default (applicati...
Obj...
Union?
Description
Add Property
Example
Inherited
Edit
"product_id": "122",
"rel_product": "http://alc.mulesoft.com/api/products/122"
"quantity": 2
Add Body
See response
Try It
Headers
COPY
Text editor
Add a header to the HTTP request.
Add
Send
200 OK
Time: 763 ms
"order_id": "order_22",
"tracking_code": "1231039120931 8",
"rel_self": "http://alc.mulesoft.com/api/orders/order_22",
"creation_date": "2015/12/31 10:00:00",
"cost": {
"price": 12,
"tax": 1,
"final_price": 13
},
"shipping": {
"shipping_address_id": "1122",
"rel_address": "http://alc.mule

Step 6: Switching between RAML and OAS view

Inside the Anypoint Platform you can work with RAML and OAS seamlessly. You can check that going to the right panel, where the specification code is shown, and selecting on the bottom the `OAS` tab.



Click the `RAML` tab to continue with the next step.

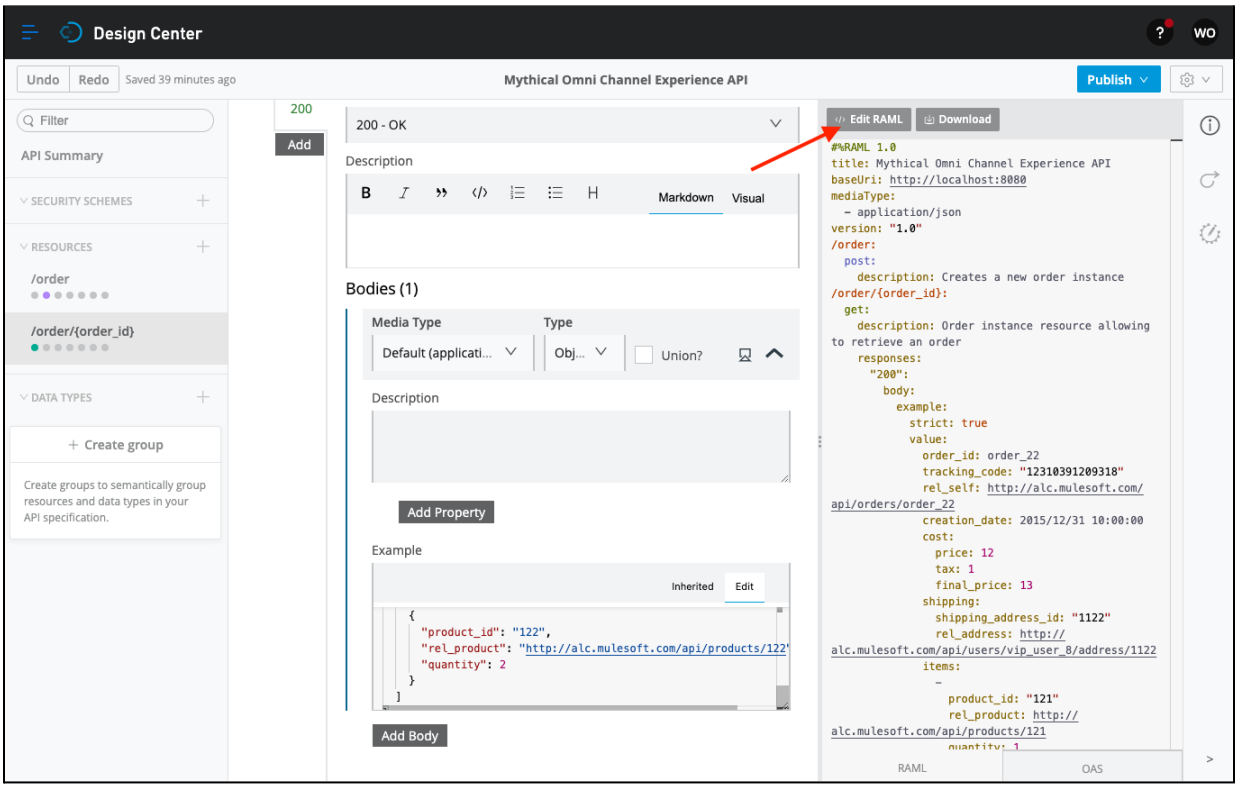


Step 7: Edit the RAML for more advanced RAML capabilities

In the Step 3 and Step 4 of this lab we created resources and methods for our Omni Channel Experience API utilizing Anypoint Visual API Designer. This is good when you are a beginner. As you get used to define your RAMLs, you would like to edit them by yourself.

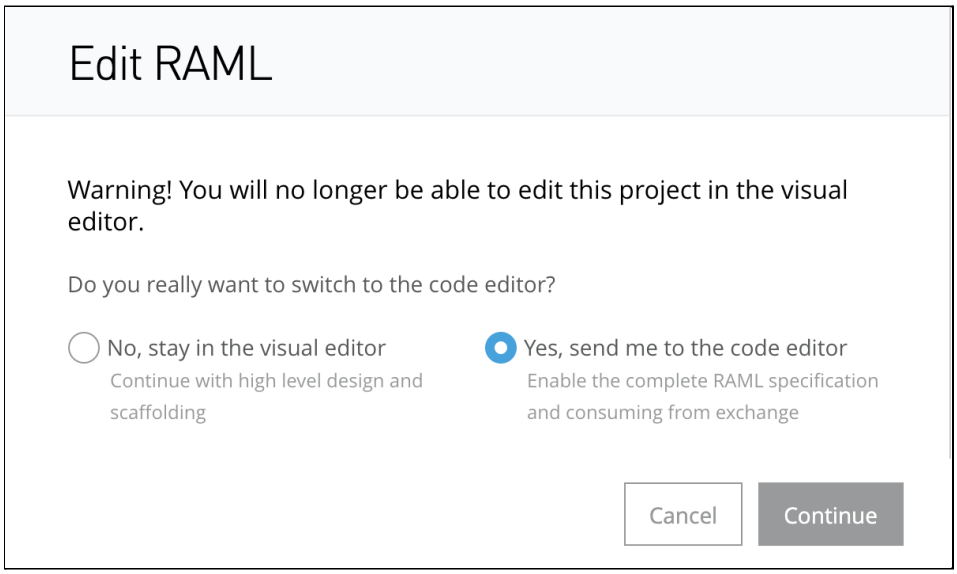
So first we need to change the view.

1. Click the Edit RAML button to go to the RAML Editor mode and confirm that you will be using the code editor instead of the visual editor.

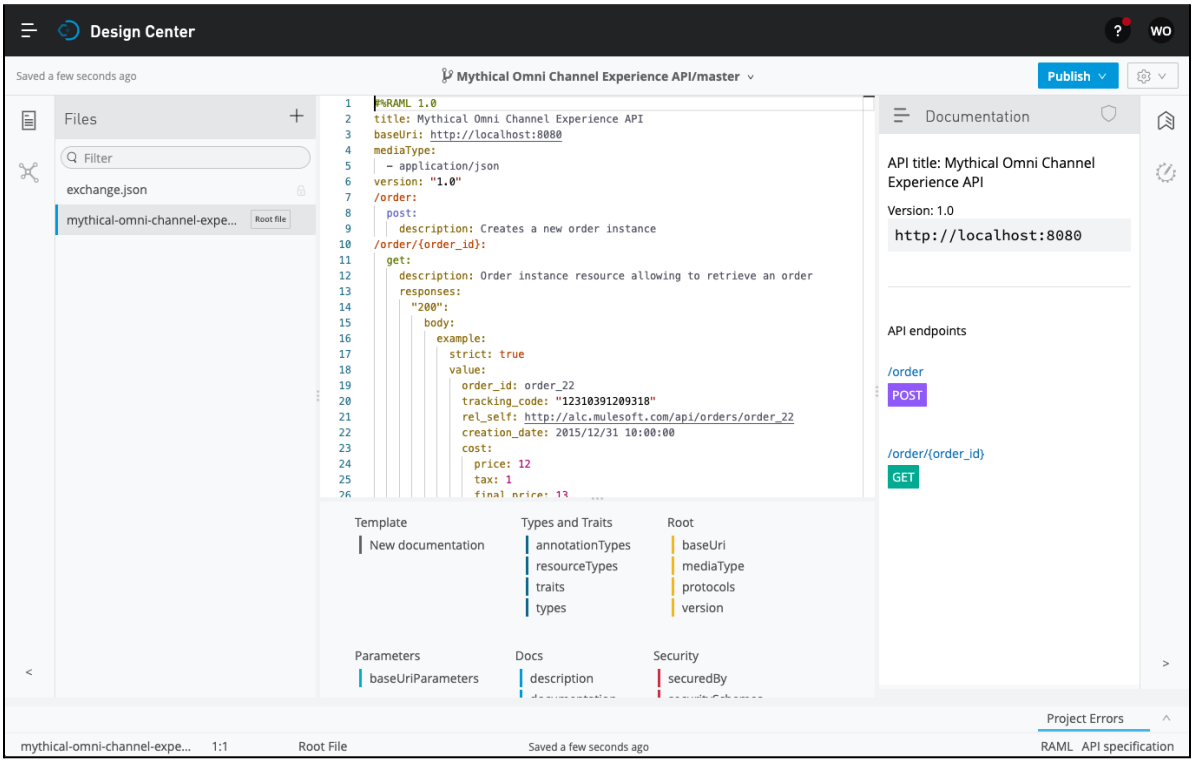


2. Read the Warning! On the popup window, please select **Yes, send me to the code editor**.

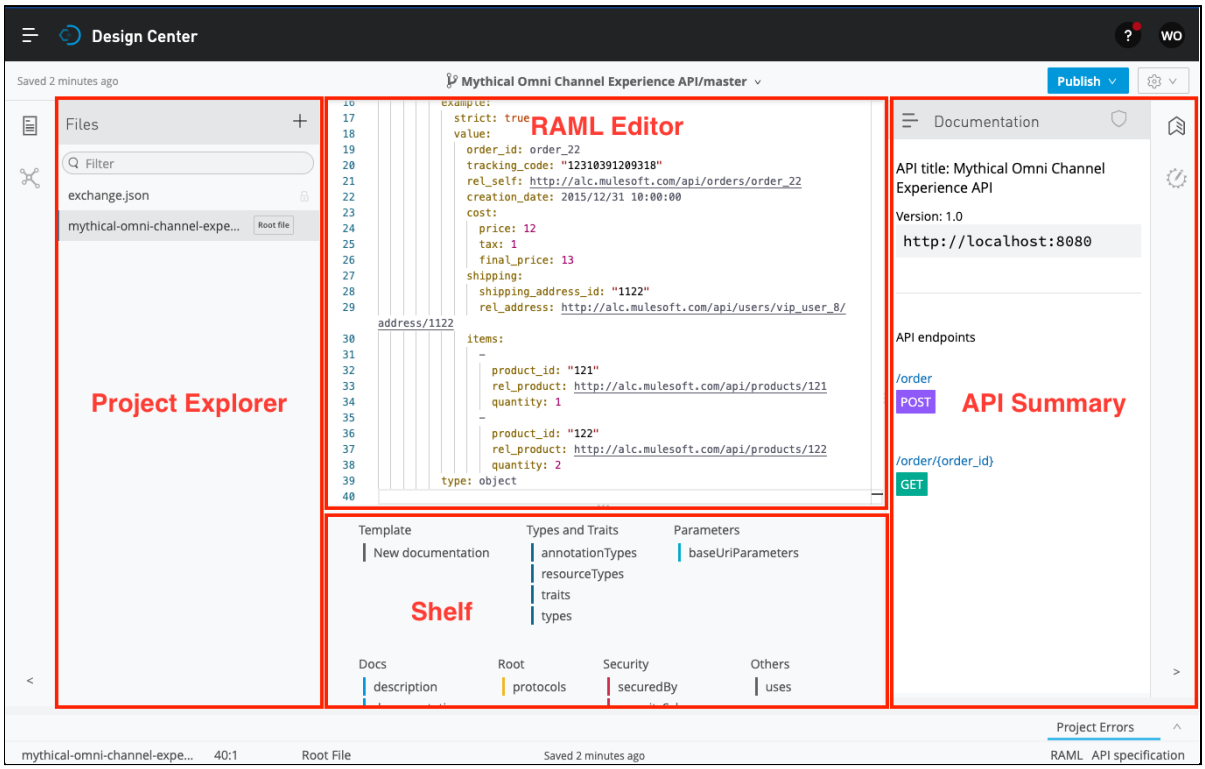
3. Press **Continue**



You should see the RAML editor.



4. Place the cursor at the end of the RAML specification on line 40.
   The screenshot and table below gives an overview of the various sections of the API Designer that you will use



5. Below is a table with a description of each section

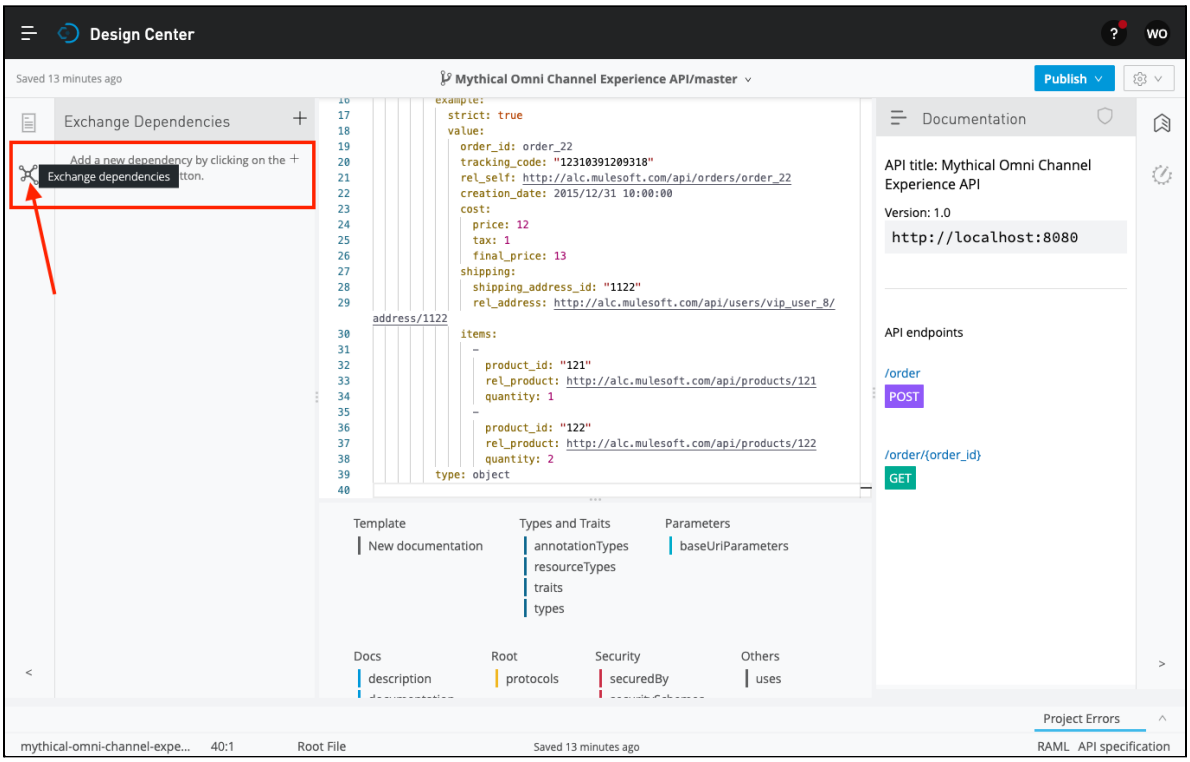
Project Explorer	Shows a list of all files in your API design project. File types supported include RAML, JSON, and any other custom file format.
RAML Editor	Where you define your API
Shelf	Where you have suggested commands available to use depending on where you are. This is contextual. You need to go to a new line to see it.
API Summary	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.

Step 8: Reusing an Asset from Exchange

In the Step 4 of this lab we created a GET methods for our Omni Channel Experience API and defined the response with a hard-coded JSON example. Wouldn't it be great if we could provide to developers an a reusable fragment that would be returned from this API instead of hard-coding? Well, that is exactly what the new version of Anypoint Exchange allows you to do.

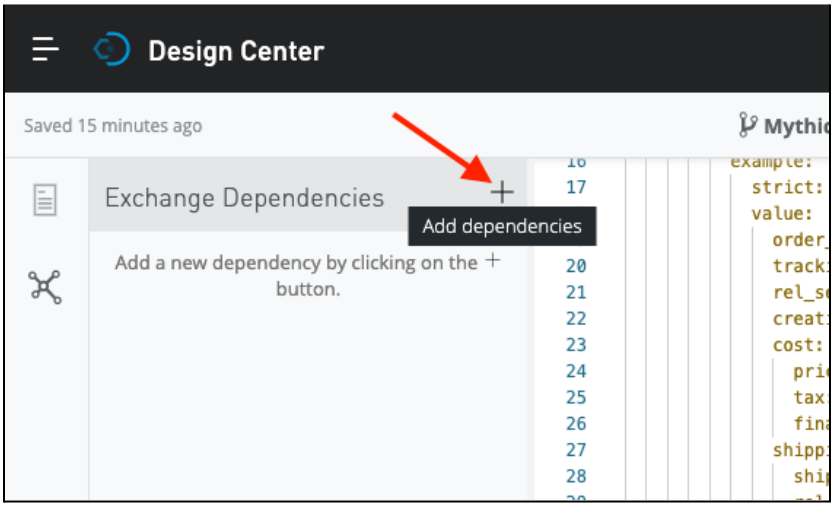
1. Let's provide an example of what the e-commerce application should expect when invoking the GET method on the /order/{order_id) resource. There are several ways to declare the example response in RAML, but lets see how easy it is to use Exchange to search for and incorporate a **RAML fragment** into our project.

   Click on the "Exchange dependency icon" on the left hand side of the API Designer:

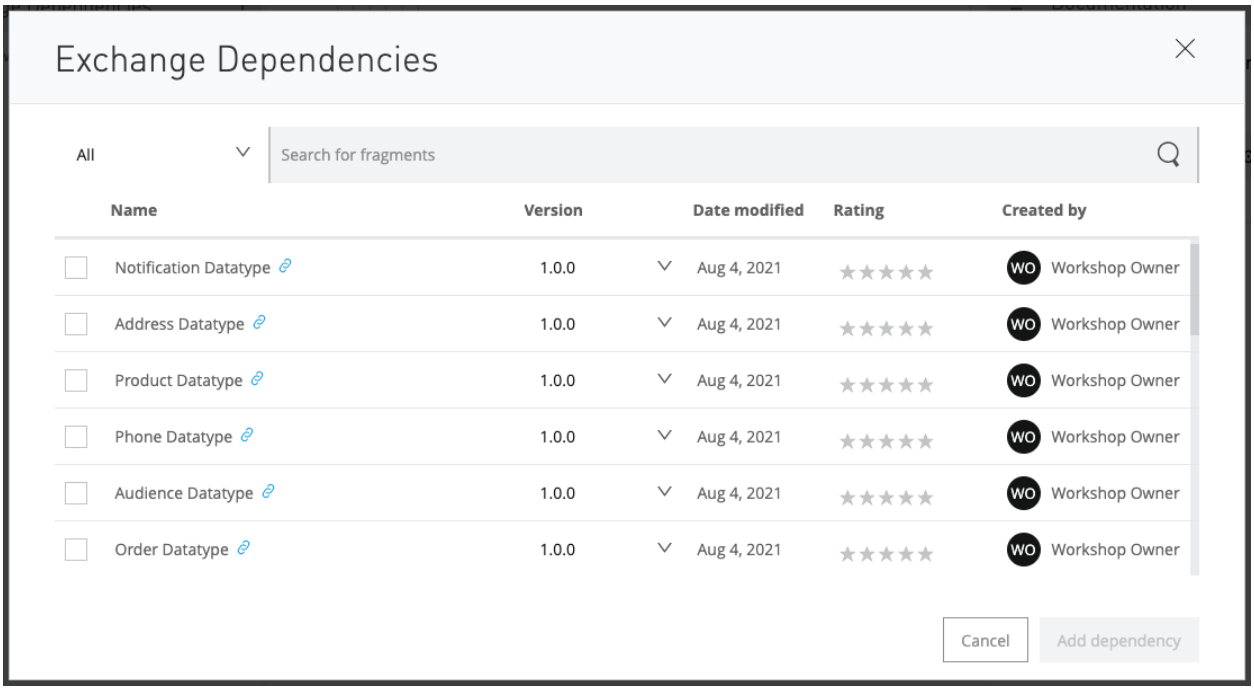


2. You should now see the project dependency section of your API specification. RAML allows API designers to not only define the resources, methods, and attributes of the API, but it also allows you to reference external files containing fragments of RAML. These fragments can define commonly used data definition, examples, traits, and resourceTypes, making the API design process easier and more consistent across the organization.

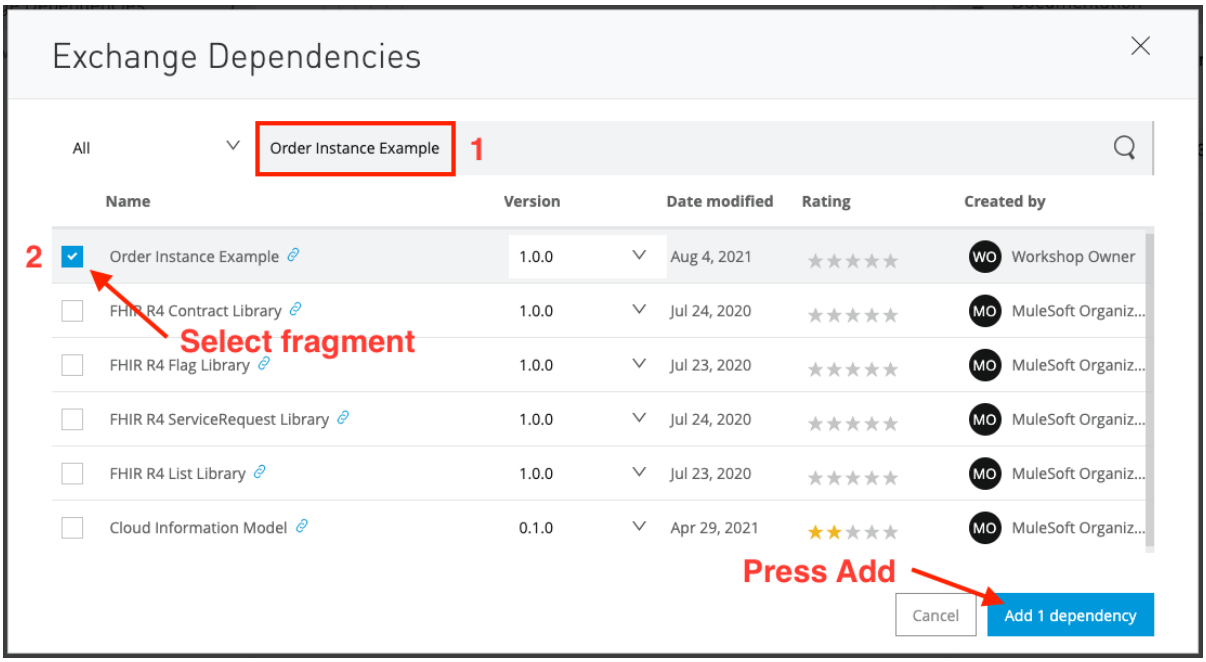
3. Click on the + button to add an Exchange dependency:



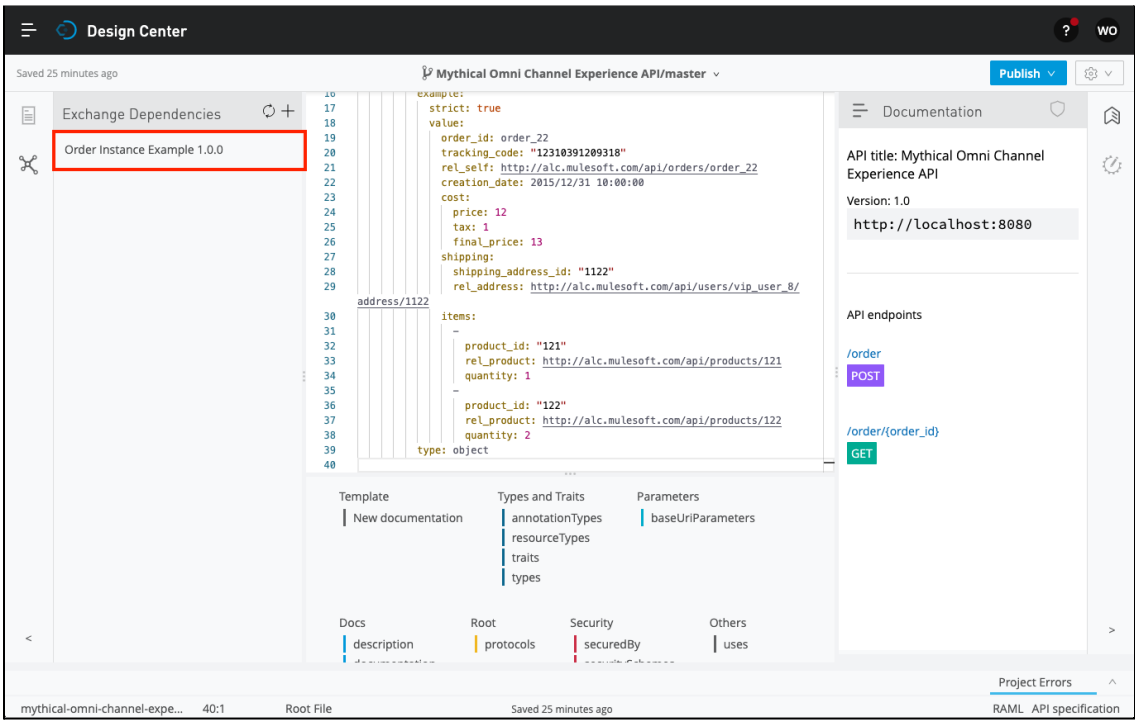
4. You should see a list of RAML fragments.



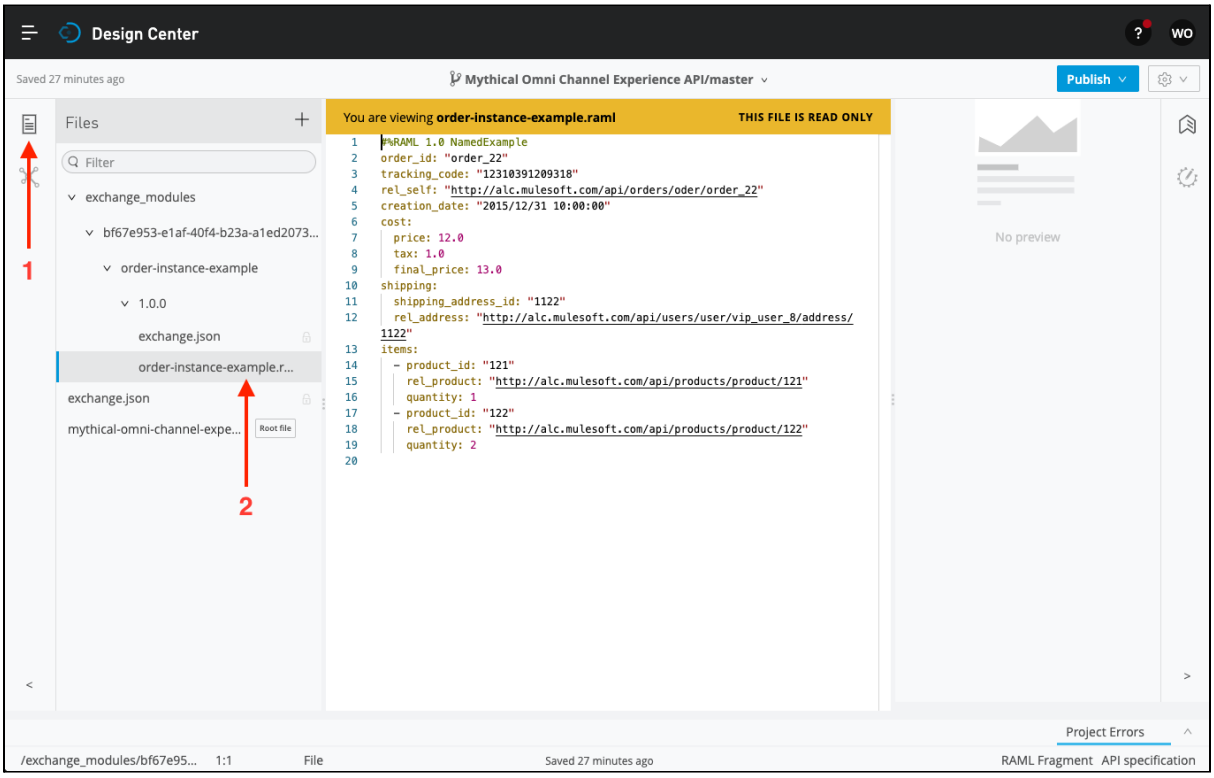
5. For the purposes of this workshop we have created a simple example of an order in Exchange. Use the search bar to find the asset titled **Order Instance Example** and add it to your project by **selecting the Order Instance Example fragment** and pressing **Add 1 dependency**:



6. The API Designer will now pull the **Order Instance Example** RAML fragment from Exchange.

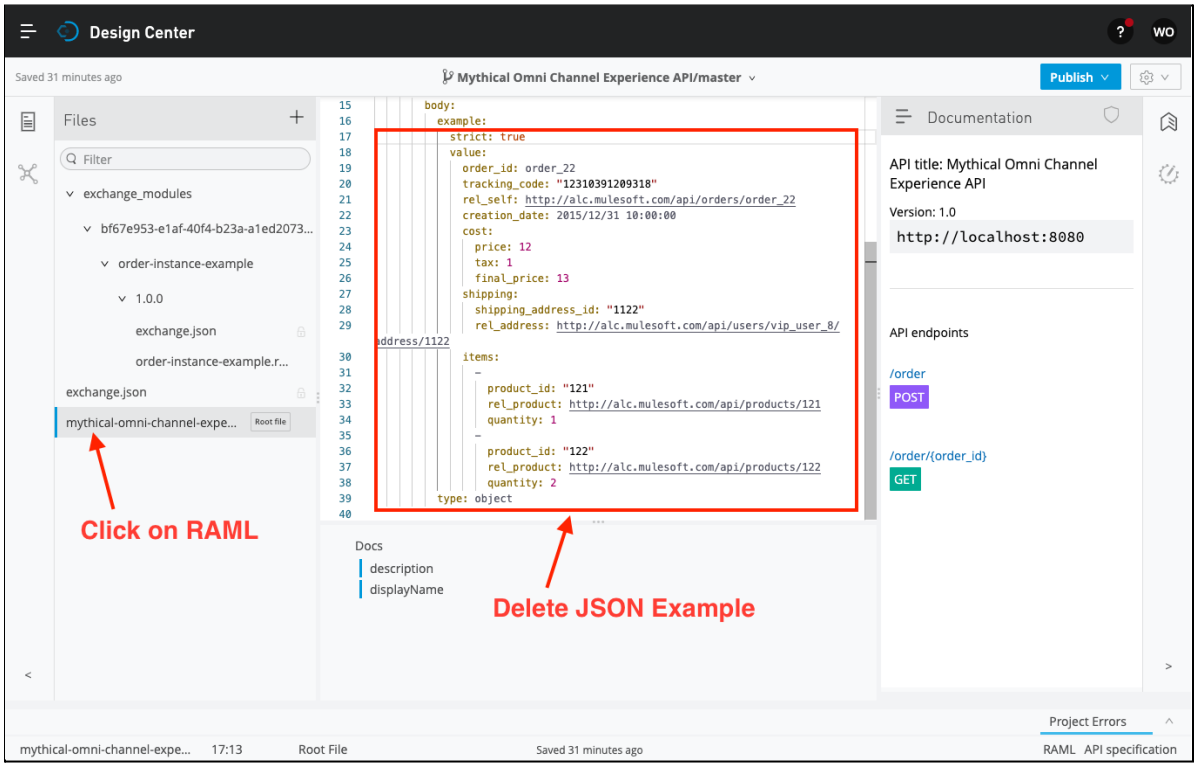


7. To see these files, click on the **Files** icon at the top left of the API Designer. Expand **exchange_modules** and select **order-instance-example.raml**. Your Project Explorer should look something like this:



8. The above RAML fragment on the right defines an example of what an Order instance might look like. By defining this example using a format-neutral language like RAML (instead of a JSON object which is what is expected to be returned) we give the API designer flexibility to change the method's data format (i.e. from JSON to XML) and still be able to create a valid example.

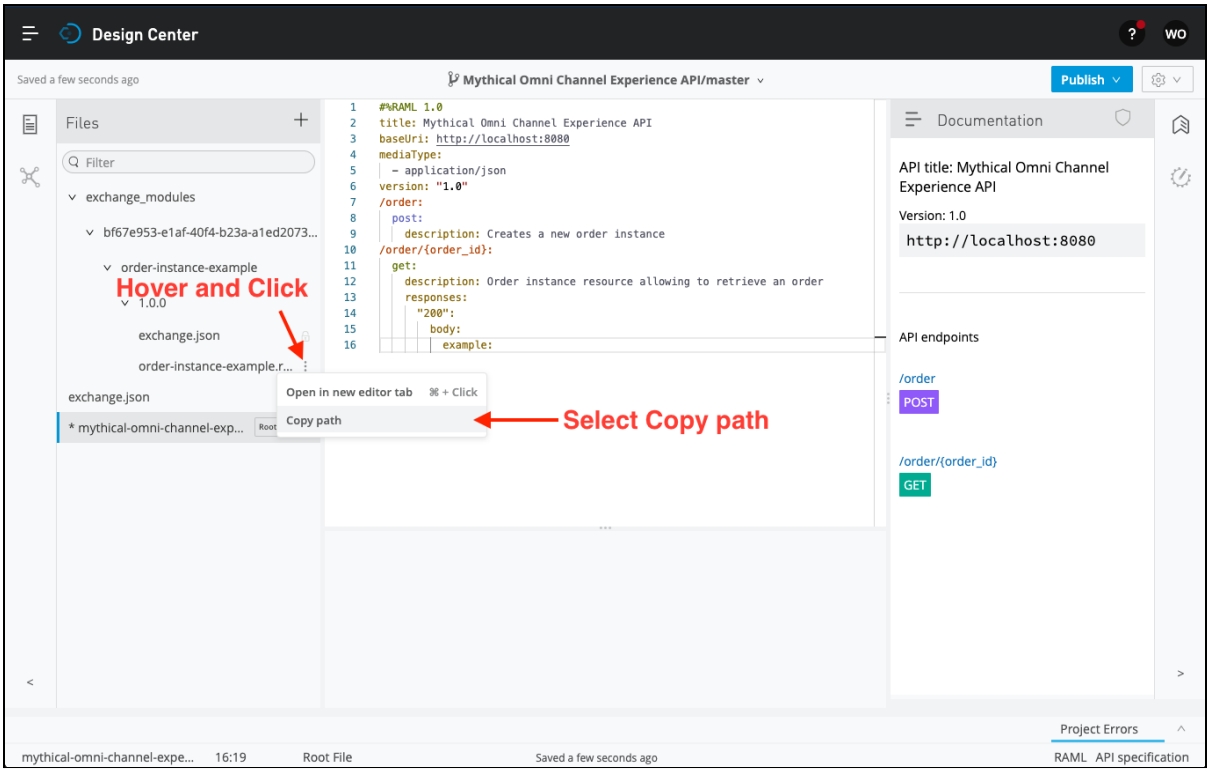
9. Remove the hardcoded example we wrote on Step 4



At this point we have only referenced the RAML fragment from our project. The last step is to designate the Order example as the **example:** data for the **/order/{order_id}:** resource.

Let's start by copy the Resource complete path into the clipboard.

10. Click on the ellipsis menu and select Copy path:

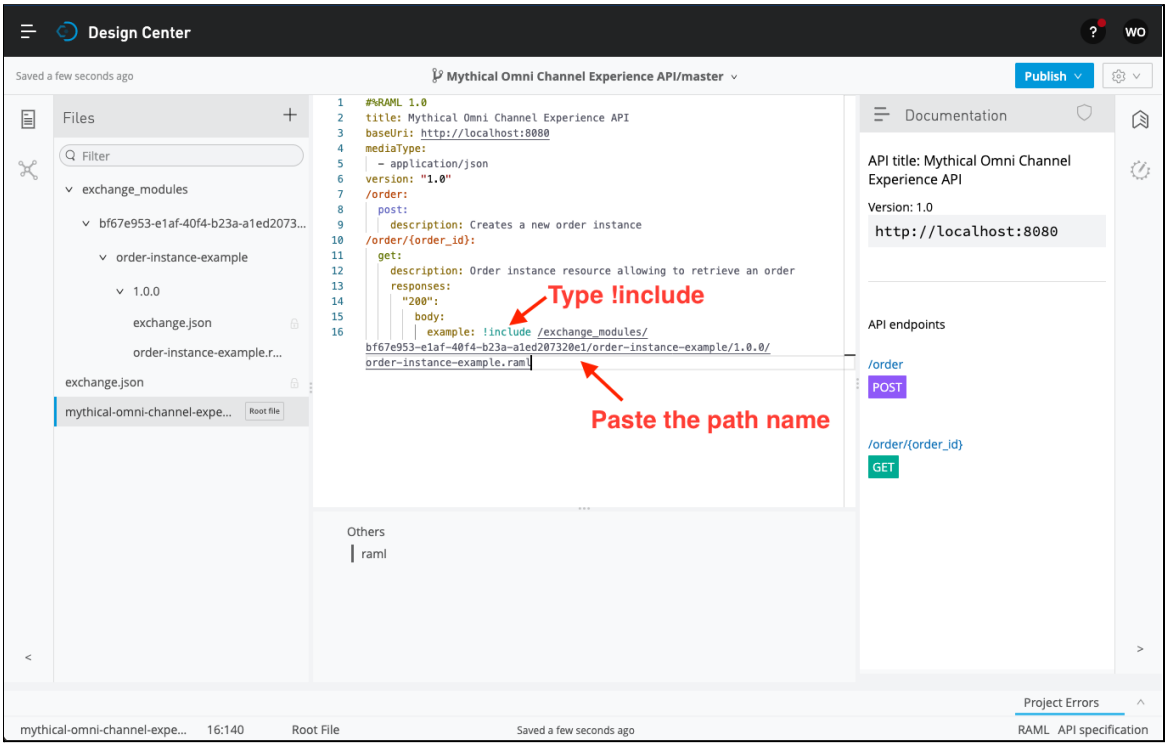


11. Now, return to the bottom of our main API specification file and insert the following text:

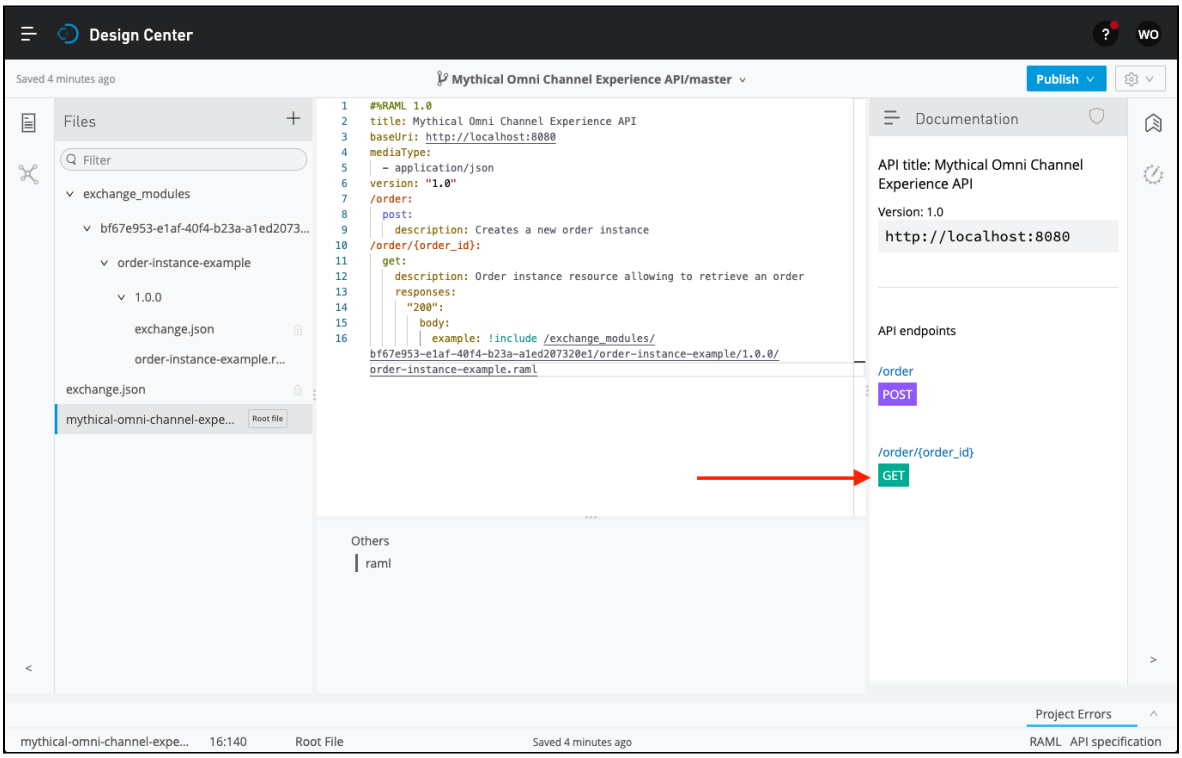
```
examples: !include
```

The **!include** directive allows you to create a relative path reference to other files in your RAMl project.

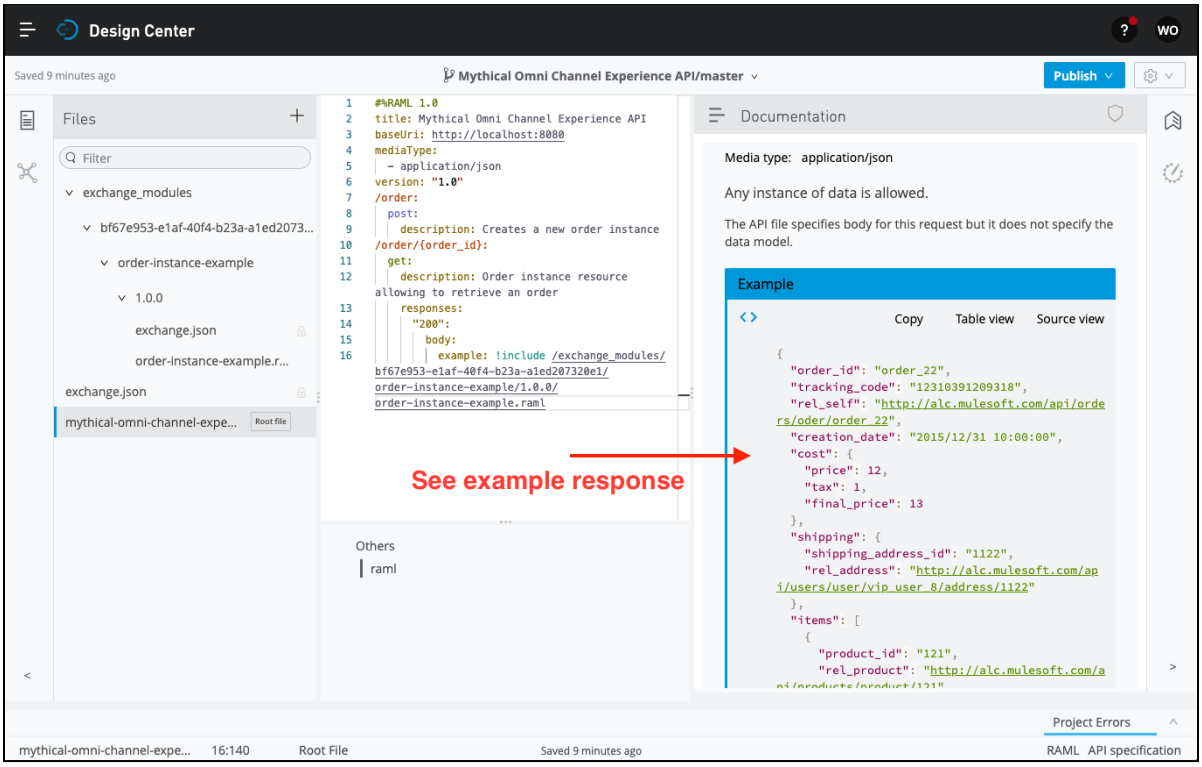
Paste the pathname next to the **!include** directive in the RAML file.



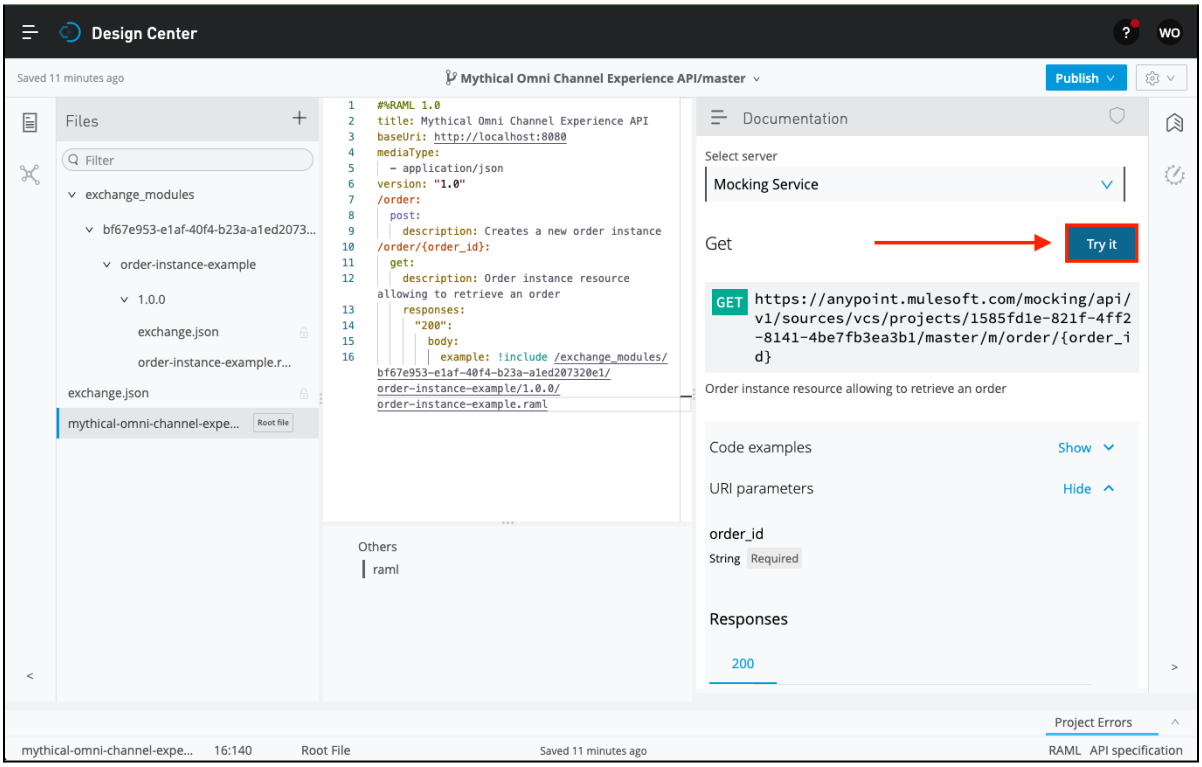
12. Take a look at the API Summary. You could see the Get resources under /{order_id}



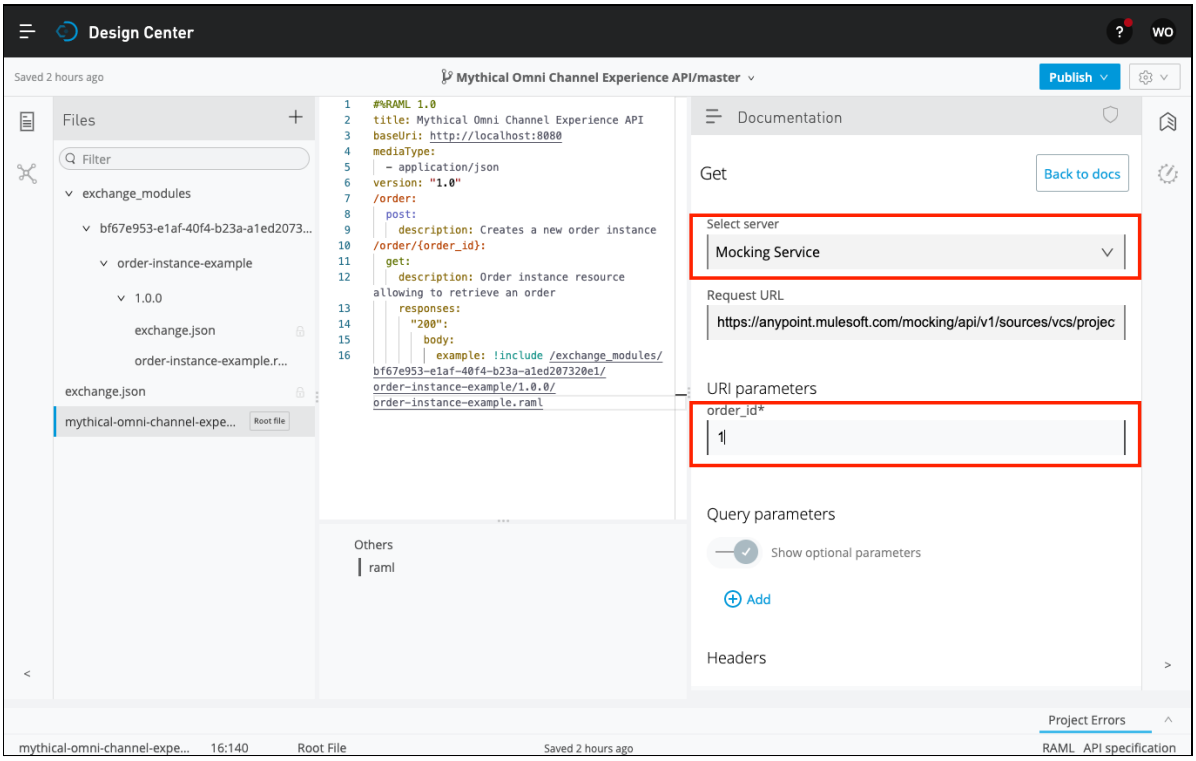
13. Click on the **Get** resource and you will see the description:
    - Operation description
    - URI Parameters
    - Response examples Scroll to the bottom of **Get** resource summary.



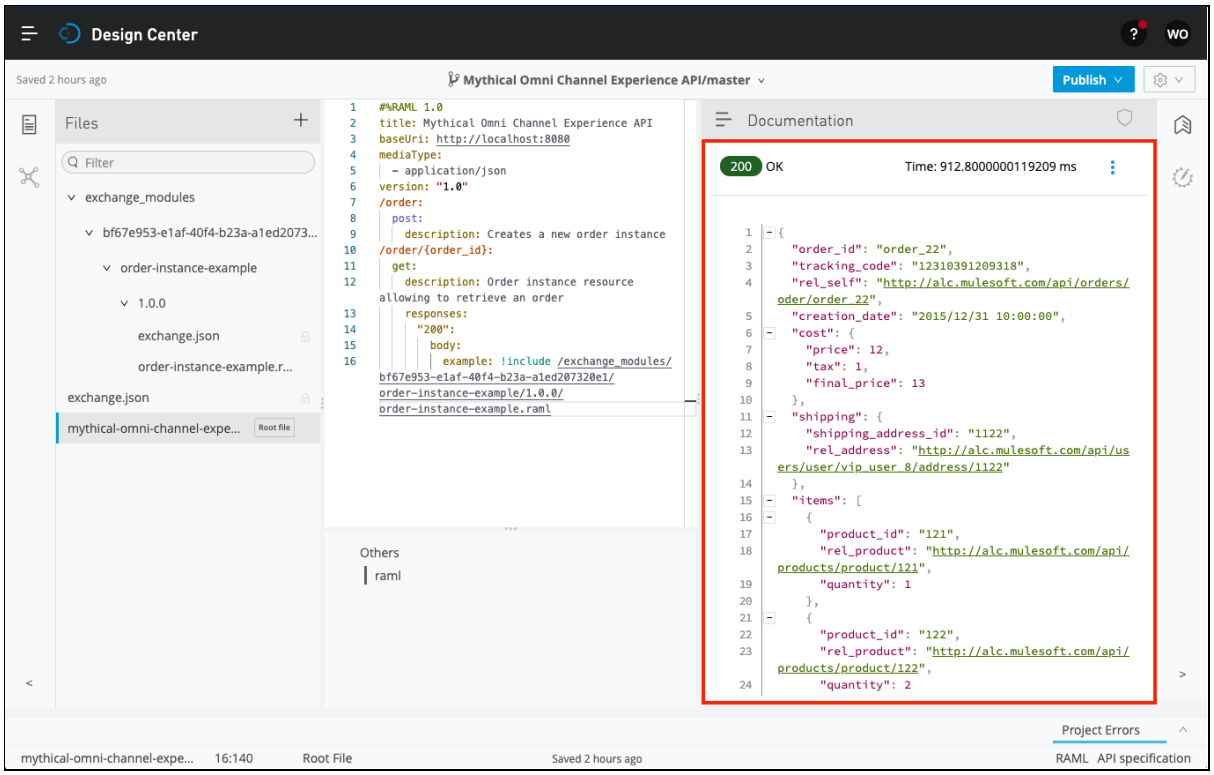
14. Press the Try **it** button and test the application again.



15. You will see the following page. In the **Select server**, select the **Mocking Service. Update the Order_ID to 1**:
16. Press send.



17. Scroll down and you should see a 200 response with the Order record in JSON format. Notice that the data displayed is from the example file you **!included**.



Summary

In this lab, you completed the following steps:

- Step 1: Review the E-Commerce Application Requirements
- Step 2: Create the Omni Channel Experience API
- Step 3: Create the Orders List Resource
- Step 4: Create the Order Instance Resource
- Step 5: Enable the Mocking Service
- Step 6: Switching between RAML and OAS view
- Step 7: Edit the RAML for more advanced RAML capabilities
- Step 8: Reusing an Asset from Exchange

We easily created and designed a new experience API for our e-commerce 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**.

We highly recommend you take the time to explore the API Specification for the full **Omni Channel Experience API** located in the Design Center. It provides a comprehensive example of an experience API specification and can be used as a proper example when designing new APIs.

To learn more about RAML follow this tutorial.

See Designing Your API for more information on API design.

Also review the following free ebooks on API design.

- Undisturbed REST: A Guide to Designing the Perfect API
- 7 Habits of Effective API & Service Mgmt
- 7 Steps to Building Your API Blueprint
- Ben Kepes on Why APIs are Critical to Business

Congratulations! You have completed Lab 2.

Please proceed to Lab 3

Lab 3: Publish your API to Exchange

Overview

In Lab 1 we examined how to use **Anypoint Exchange** to facilitate the API discovery process. In Lab 2 we learned how to use **Anypoint Design Center** to design the specifications for a modern RESTful API. In this lab we will learn how to publish our API to **Anypoint Exchange** so that API consumers and API developers within the organization are able to leverage the API through "self-service".

One of the biggest challenges encountered by early adopters of SOA was how to scale their development activities. SOA maintains many of the reuse principles that we have discussed in our "API-led" approach to software development. Unfortunately, most SOA efforts could not achieve their reusability objectives because of a few limiting factors including:

- Poorly documented services
- Lack of mature tools to facilitate service discovery
- Over-reliance on limited number of knowledgeable resources
- Burdensome processes for gaining access to services

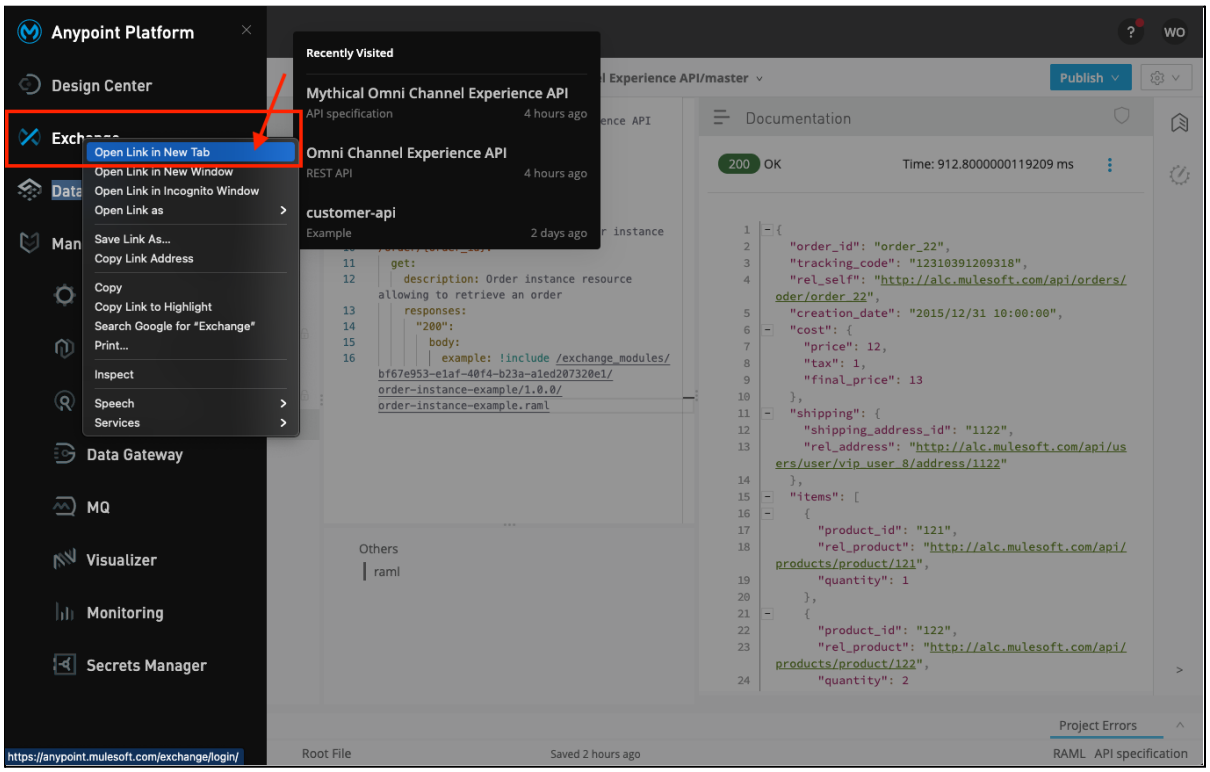
As a result, many SOA-based initiatives were only able to partially achieve their reusability objectives because developers would adopt the path of least resistance and simply recreate services rather than reuse them. The **MuleSoft Anypoint Platform** addresses all of these issues by providing innovative tools like **Anypoint Exchange** that alleviate these challenges.

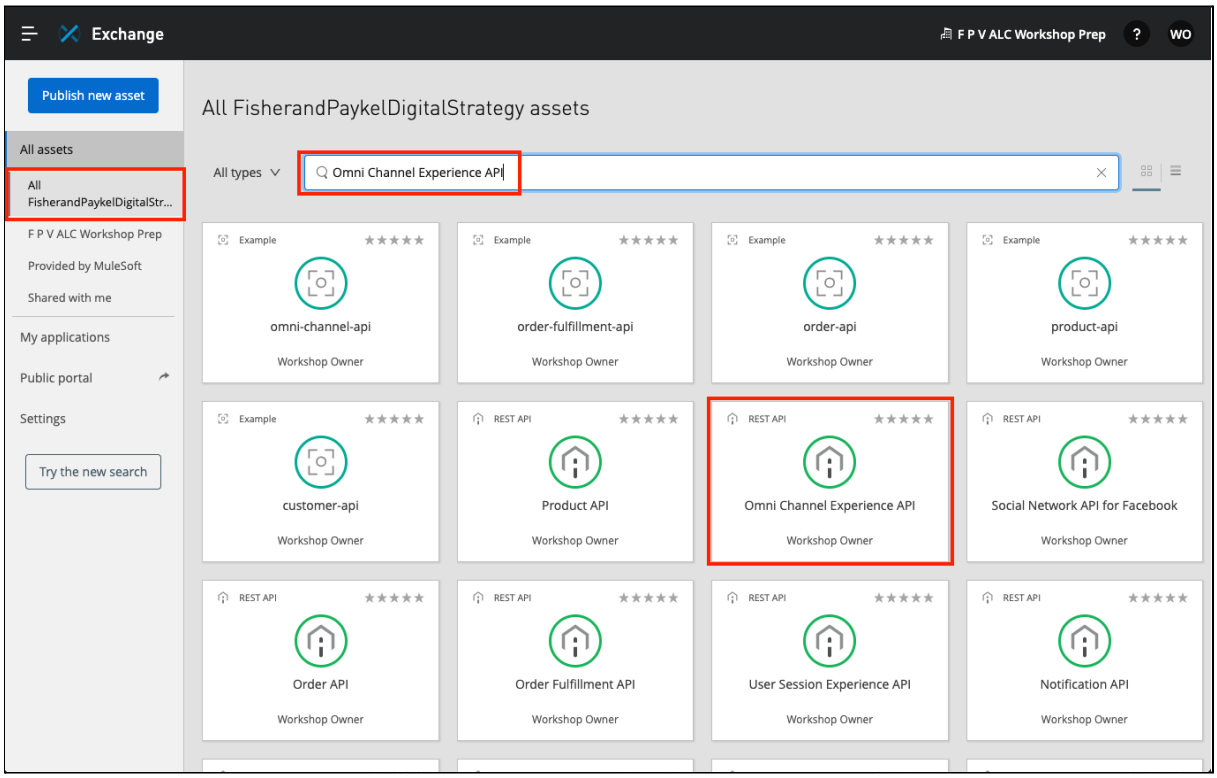
Let’s take a look at the process for publishing and documenting our reusable Omni Channel Experience API.

Step 1: Complete the Omni Channel Experience API Specification

Before we get started, make sure you are logged into the **Anypoint Platform** and viewing the **Anypoint Design Center** screen. You should have your "<username> - Omni Channel Experience API" open and visible. We have only defined a few resources and methods at this point and need to author the full API in order to complete the exercise. Rather than build it from scratch, we have the ability to import an API specification previously constructed for the workshop.

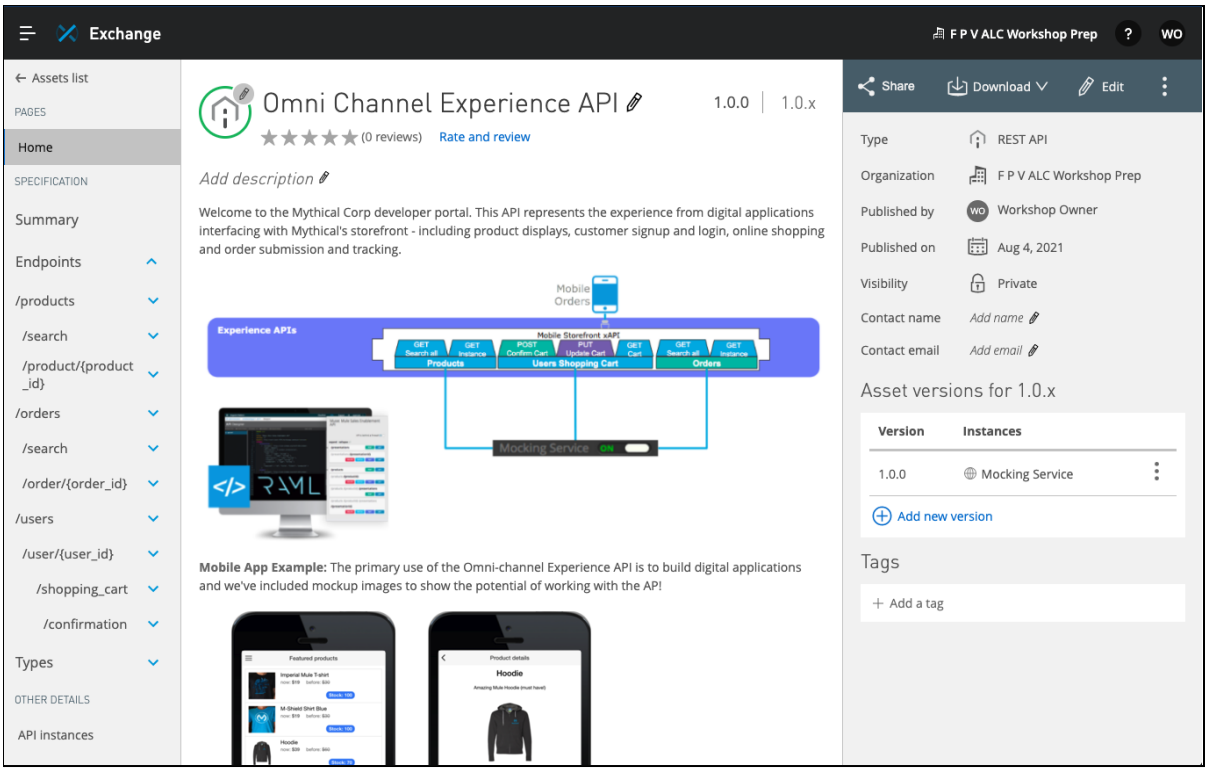
1. In a new browser tab, go to the **Anypoint Platform** and search **Anypoint Exchange** for the API named "Omni Channel Experience API".



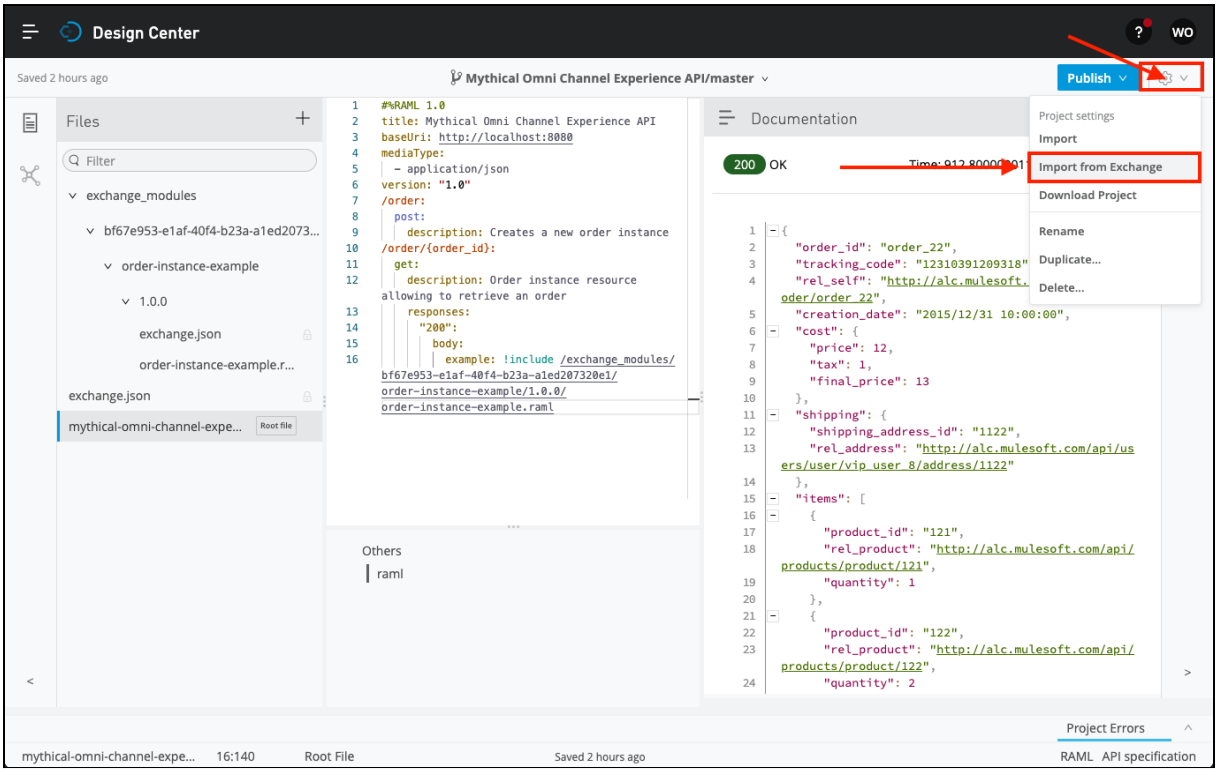


2. Select the Omni Channel Experience API once found

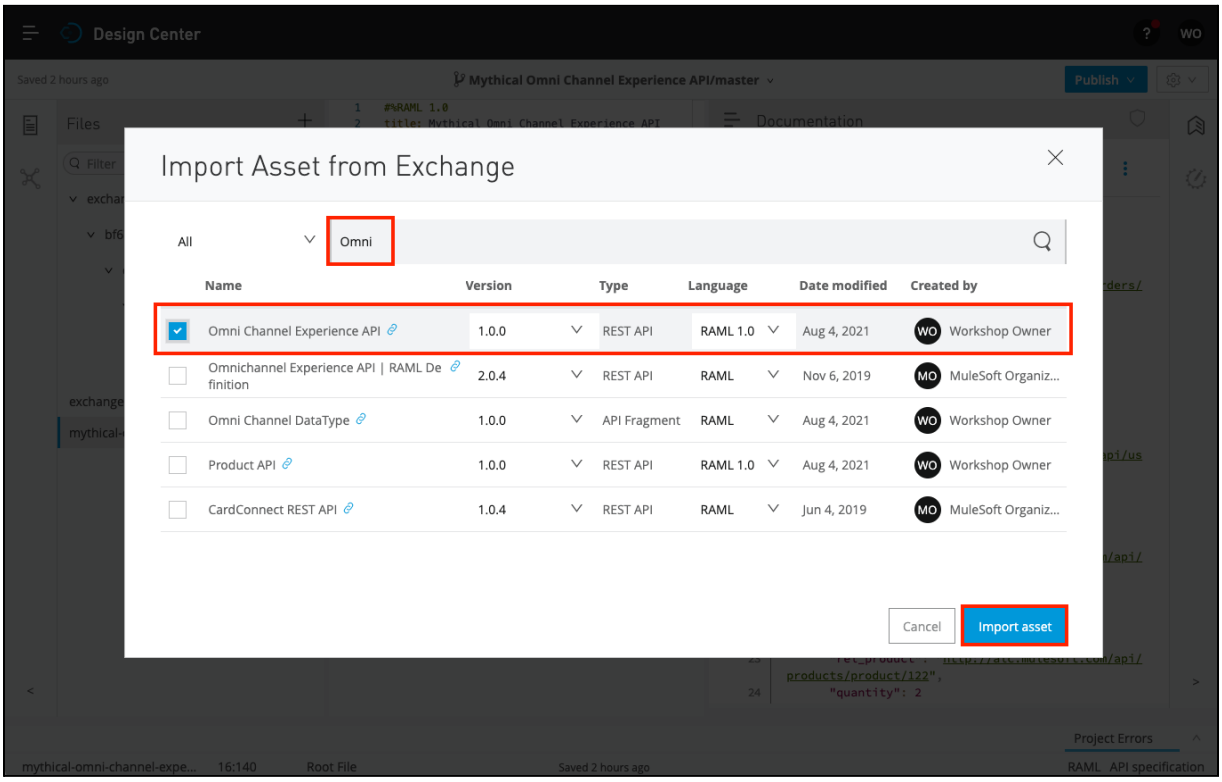
3. You should see the following once the API is selected. We will be using this API specification to overwrite the <yourinitials> Omni Channel Experience API in step 4



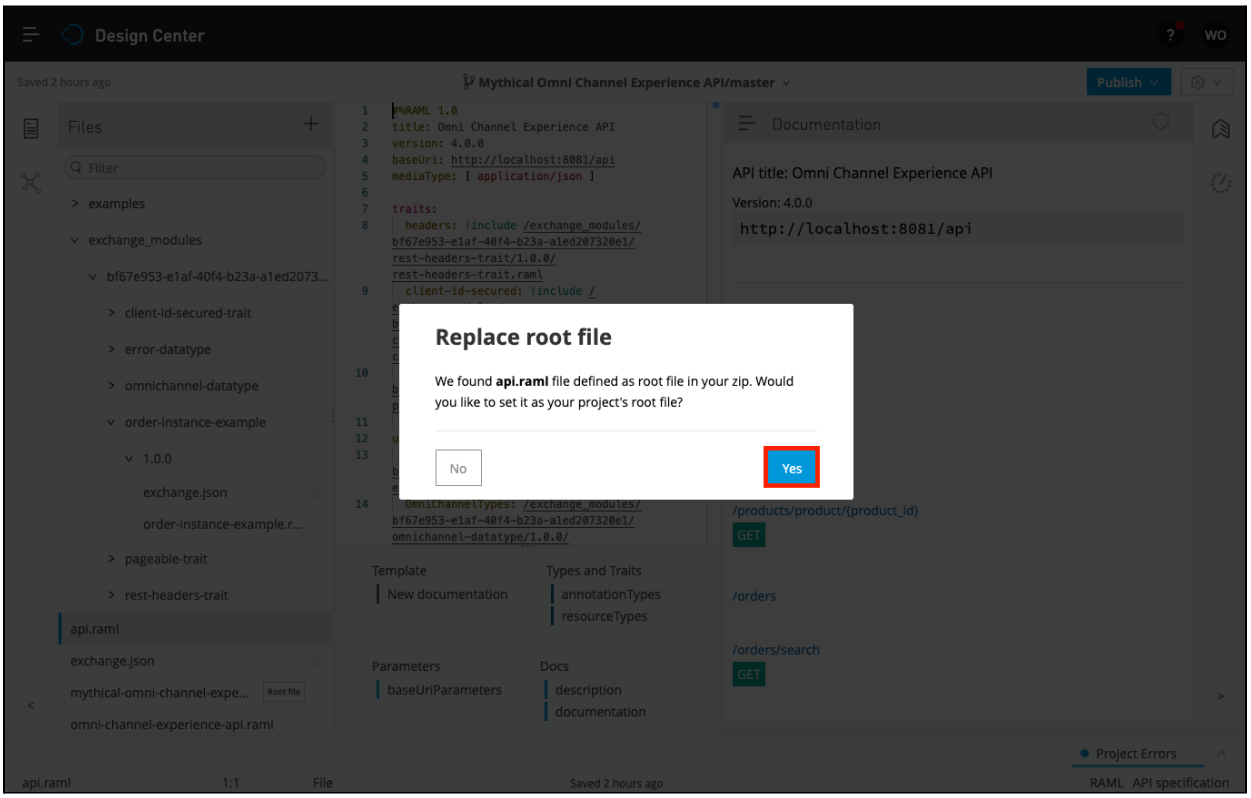
4. Click on the Design Center Tab, Click on the settings button on the right and select “Import from Exchange”



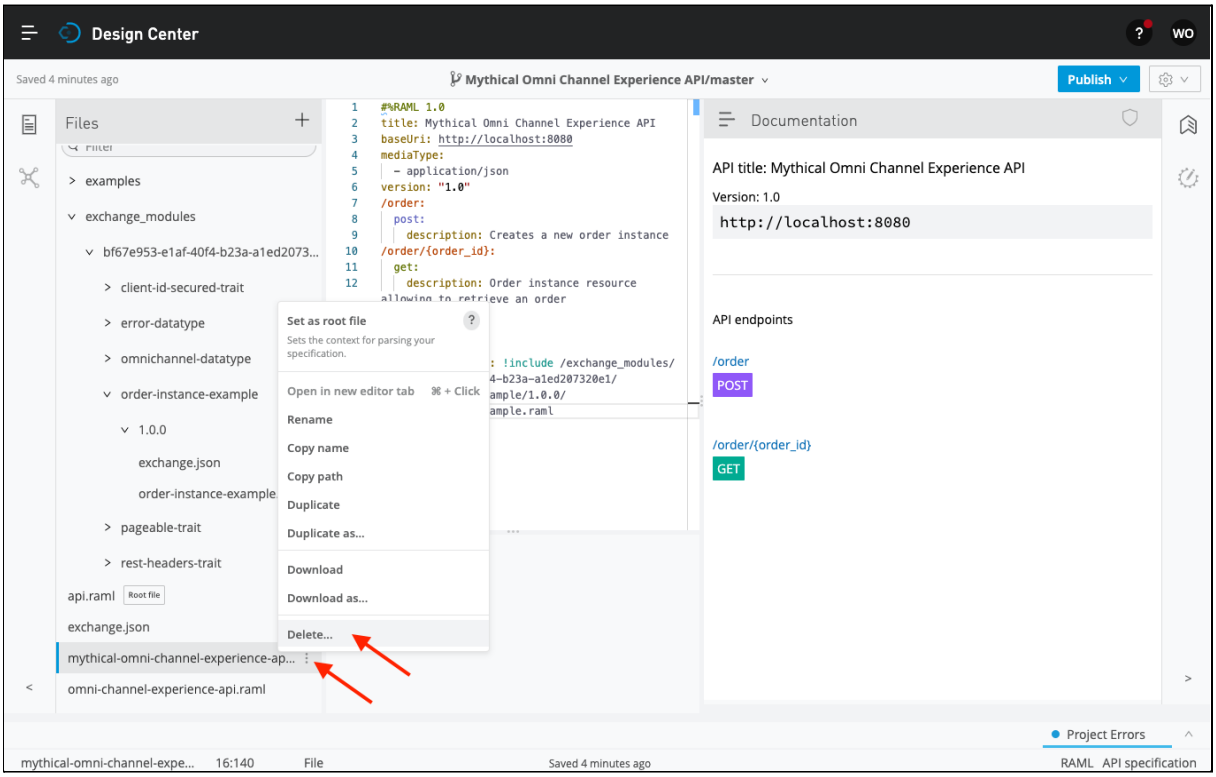
5. Search for **Omni Channel Experience API**, **select it** then press **Import asset**



6. You will be prompted to overwrite the root file api.raml - choose **yes**.



7. Delete the RAML file ***user-omni-channel-experience-api.raml*** from your project explorer that we were previously editing. We no longer need that file since the API we just imported replaces that specification. Click the "ellipsis" menu next to your file and then click "delete"



8. Delete the *omni-channel-experience-api.raml* file.

Design Center

Saved a few seconds ago

Files

> examples

> exchange_modules

api.raml Root file

exchange.json

omni-channel-experience-api.raml

```
1  #%RAML 1.0
2  title: Omni Cha
```

Set as root file

Sets the context for parsing your specification.

Rename

Copy name

Copy path

Clone file

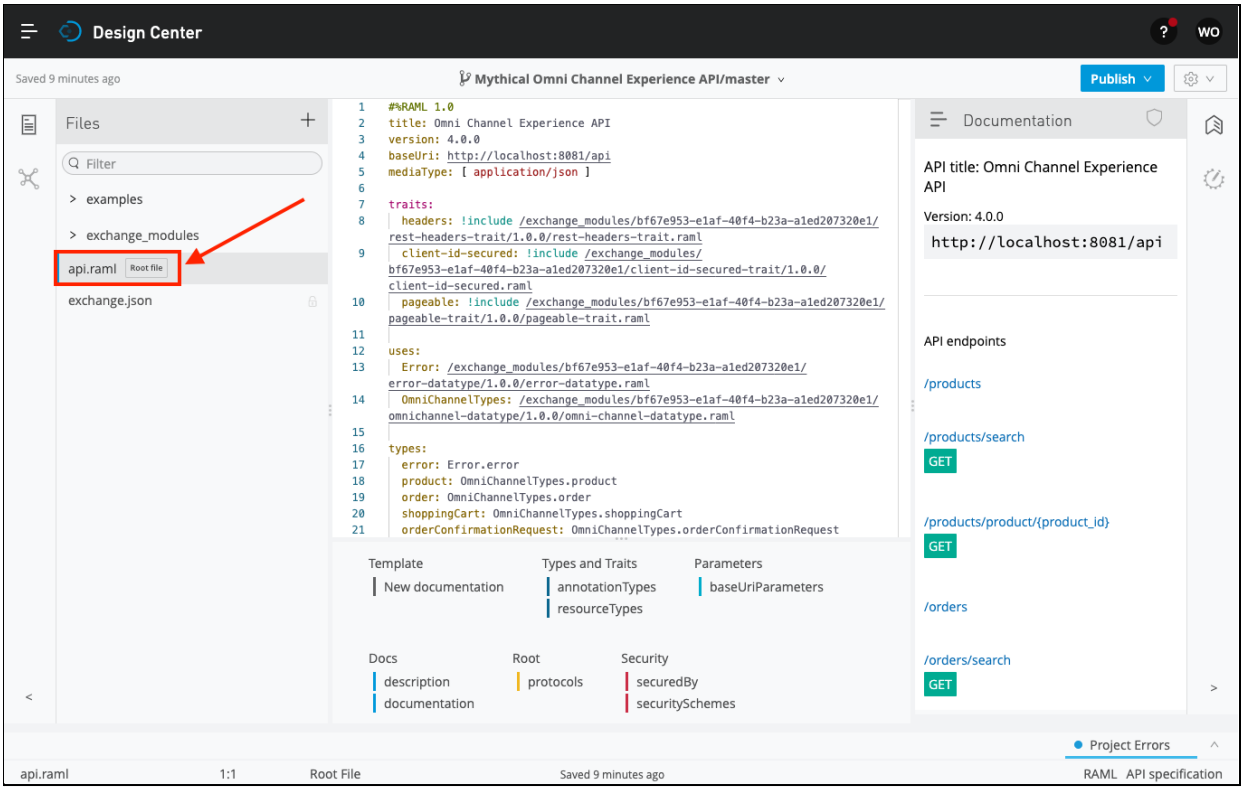
Download

Download as...

Copy as...

Delete...

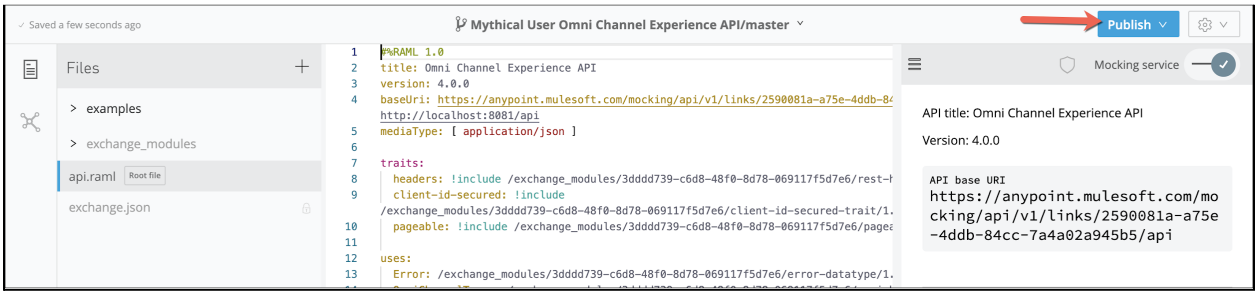
9. Select the new RAML file we imported. You should see the following complete API specification:



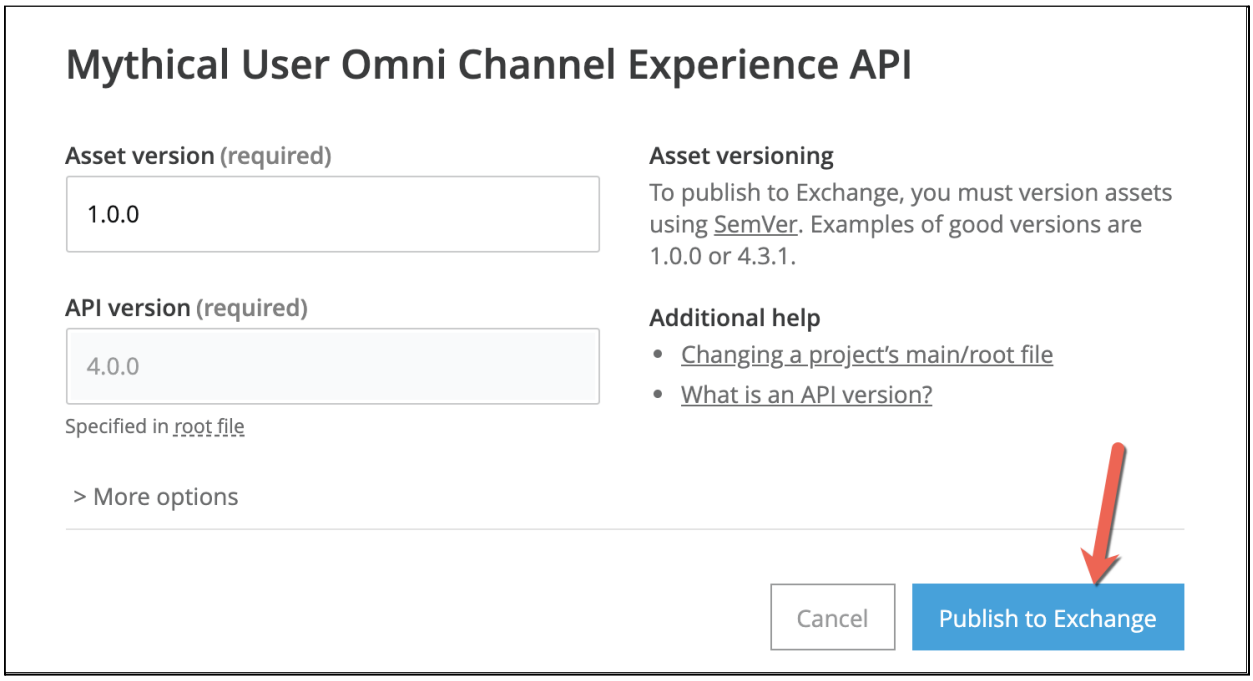
This fully formed API specification is ready to be published! Feel free to explore the API using the mocking service if desired.

Step 2: Publish Your API to Anypoint Exchange

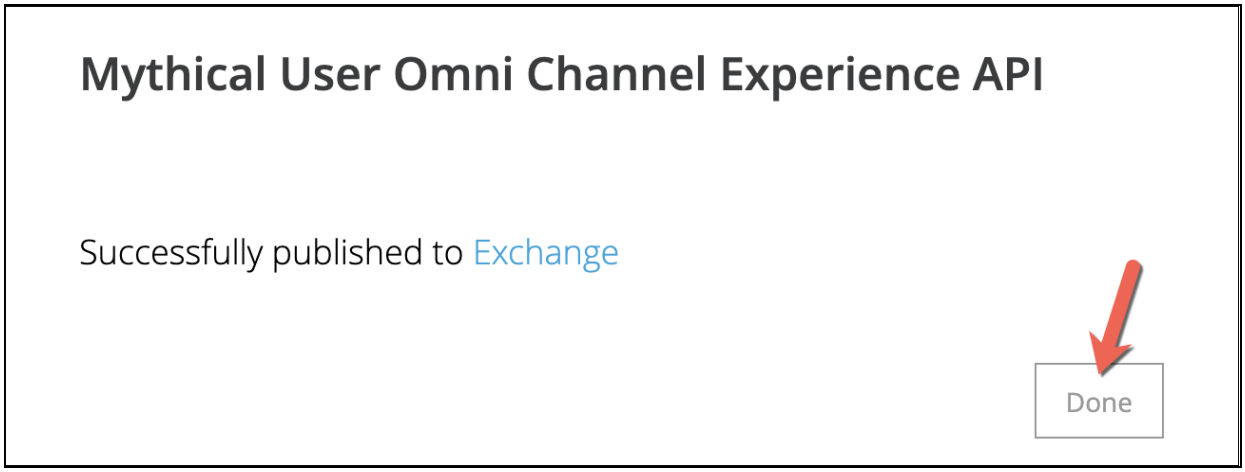
1. Click on the "Publish to Exchange" button



2. You will be presented with a popup window to capture the attributes that you want to publish to Exchange. Most of these fields will be pre-populated and the RAML will be verified to ensure that there are no errors.



3. Click the "Publish" button when you have finished reviewing the API name and version information. We have now published our Omni Channel Experience API to **Anypoint Exchange** for other developers to discover and consume!



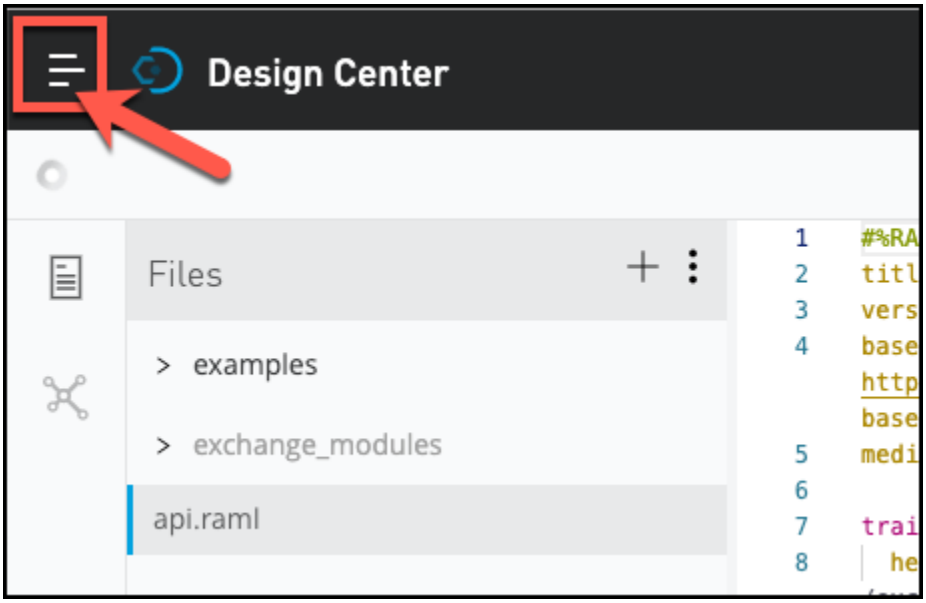
Each time you publish and API Specification you will generate two assets:

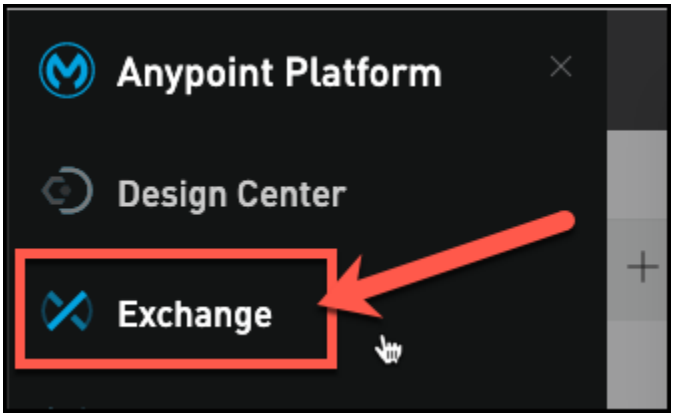
- RAML Spec: This is the RAML Specification. Everybody in the organization can discover the spec and documentation.
- Rest Connector: This is a connector automatically generated for the API you have just defined. We will cover this in more details on Module 2 and Module 9.

Step 3: Augment the Documentation for Your API

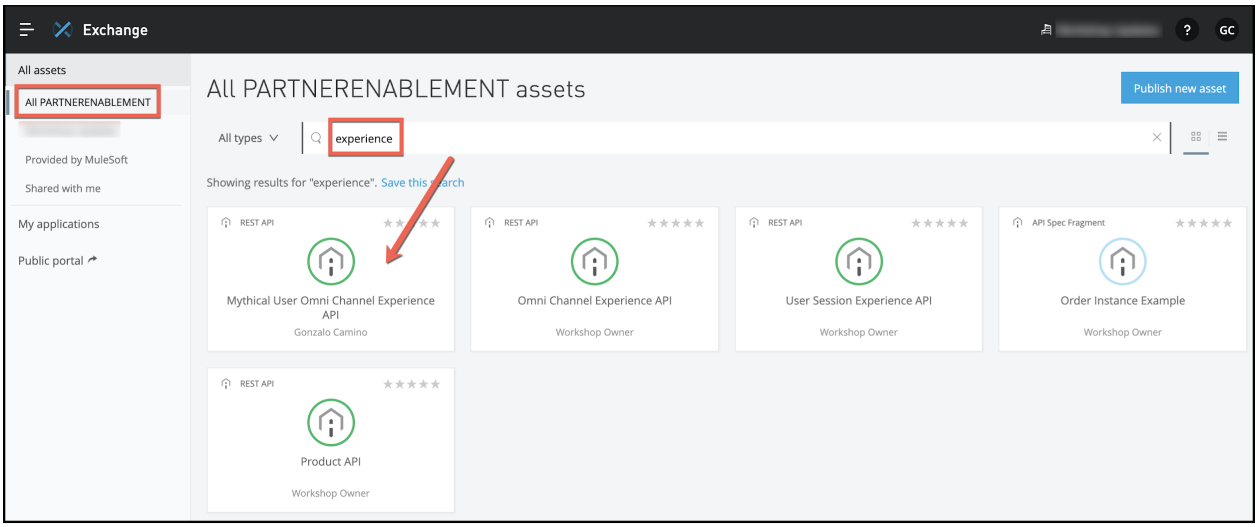
As demonstrated in Step 2, publishing your API to exchange is as easy as a few clicks. However, Exchange has really only just captured the basic attributes and documentation for your API at this point. Ensuring that your API is easy to find and "self-service ready" is your responsibility as an API designer. Let's find our API in Exchange and add more to our API documentation.

1. Navigate to Exchange 2.0 from the "hamburger menu" at the top left corner of Design Center.

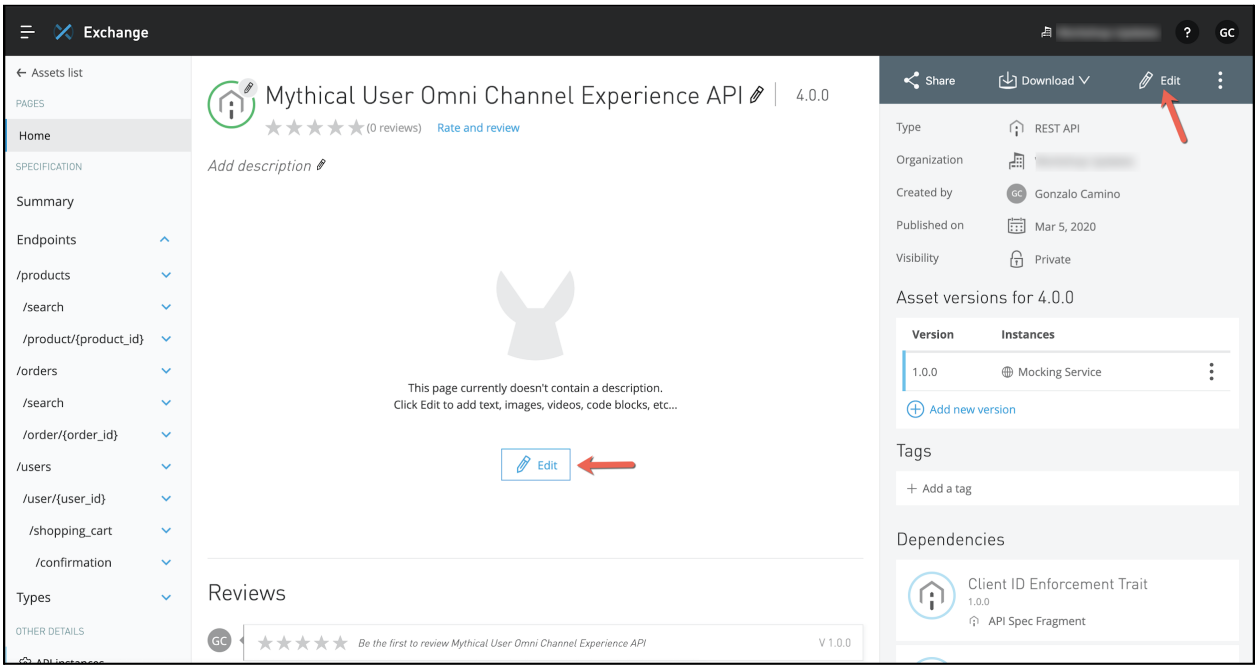




2. Use the search bar to find your API if it is not readily visible in the list of Exchange assets. Click on your API once you find it:



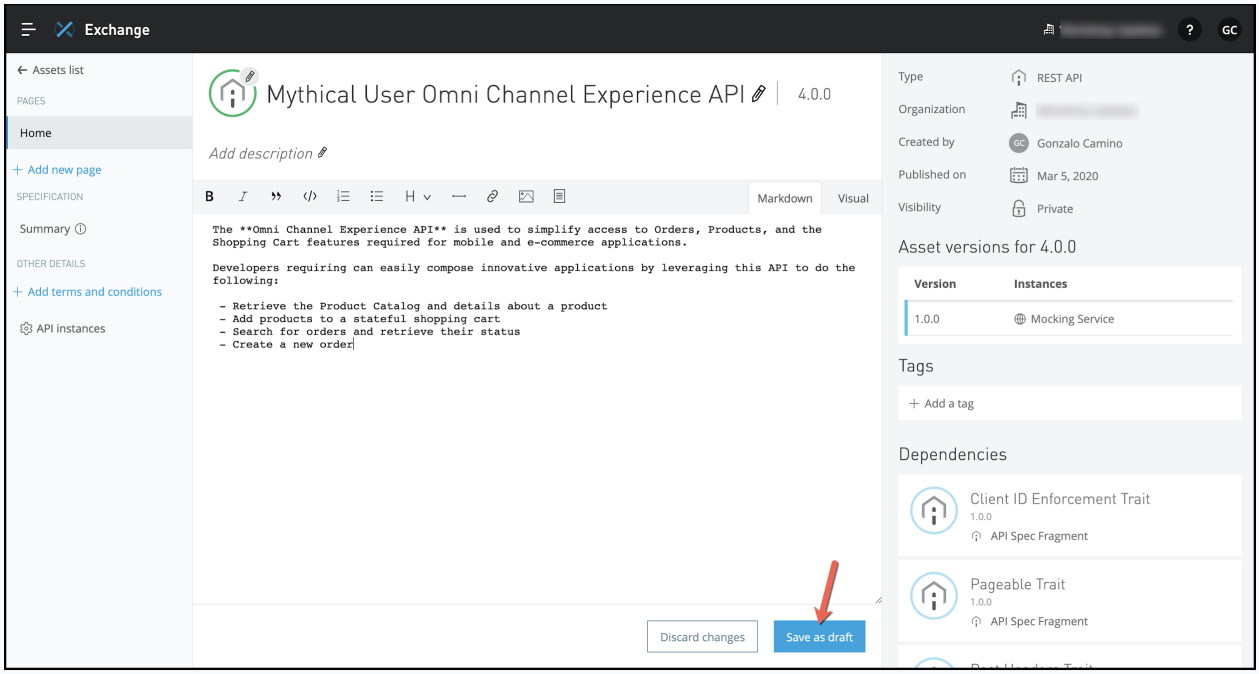
3. Click on either of the "Edit" buttons to launch into edit mode.



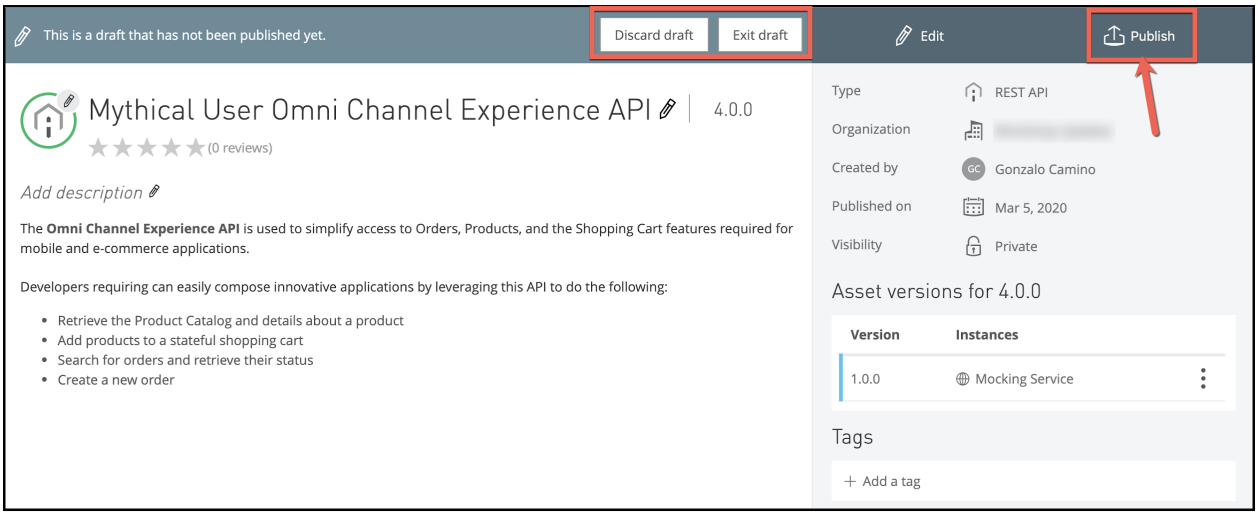
Exchange assets are documented using "markdown" language. Markdown language is a format-agnostic syntax used for creating documentation independent of how it will be rendered (i.e. HTML, PDF, text, etc). It is used by popular applications like GitHub as the standard way to create software documentation. For more information on markdown please refer to:
https://guides.github.com/features/mastering-markdown/
Create your own documentation for your API, or copy the following text into the editor:

```
The **Omni Channel Experience API** is used to simplify access to Orders, Products,
and the Shopping Cart features required for mobile and e-commerce applications.

Developers requiring can easily compose innovative applications by leveraging this
API to do the following:

 - Retrieve the Product Catalog and details about a product
 - Add products to a stateful shopping cart
 - Search for orders and retrieve their status
 - Create a new order
```

4. Click on the "Save as Draft" button at the bottom of the screen.

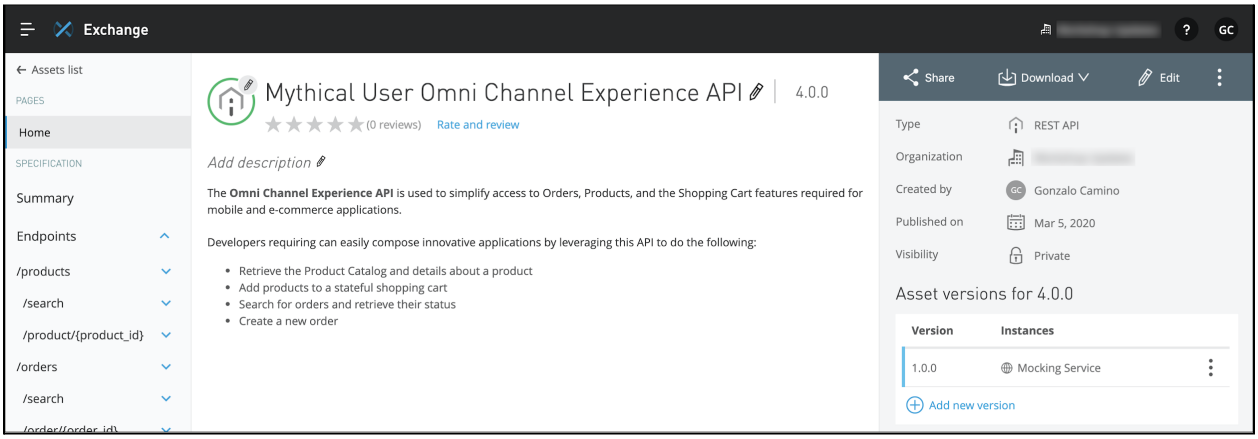


5. Your draft of the API documentation is saved but not published until you formally "Publish" the final updates to Exchange, so feel free to save multiple drafts as you create your documentation.



- You can press `Publish`, `Discard Draft` or `Exit Draft`.
  - Publish - Publish documentation changes to exchange
  - Discard Draft - Throw away changes
  - Exit Draft - Keep changes, but do not publish to Exchange

6. Press the `Publish` button. Your documentation changes are now visible on Exchange!

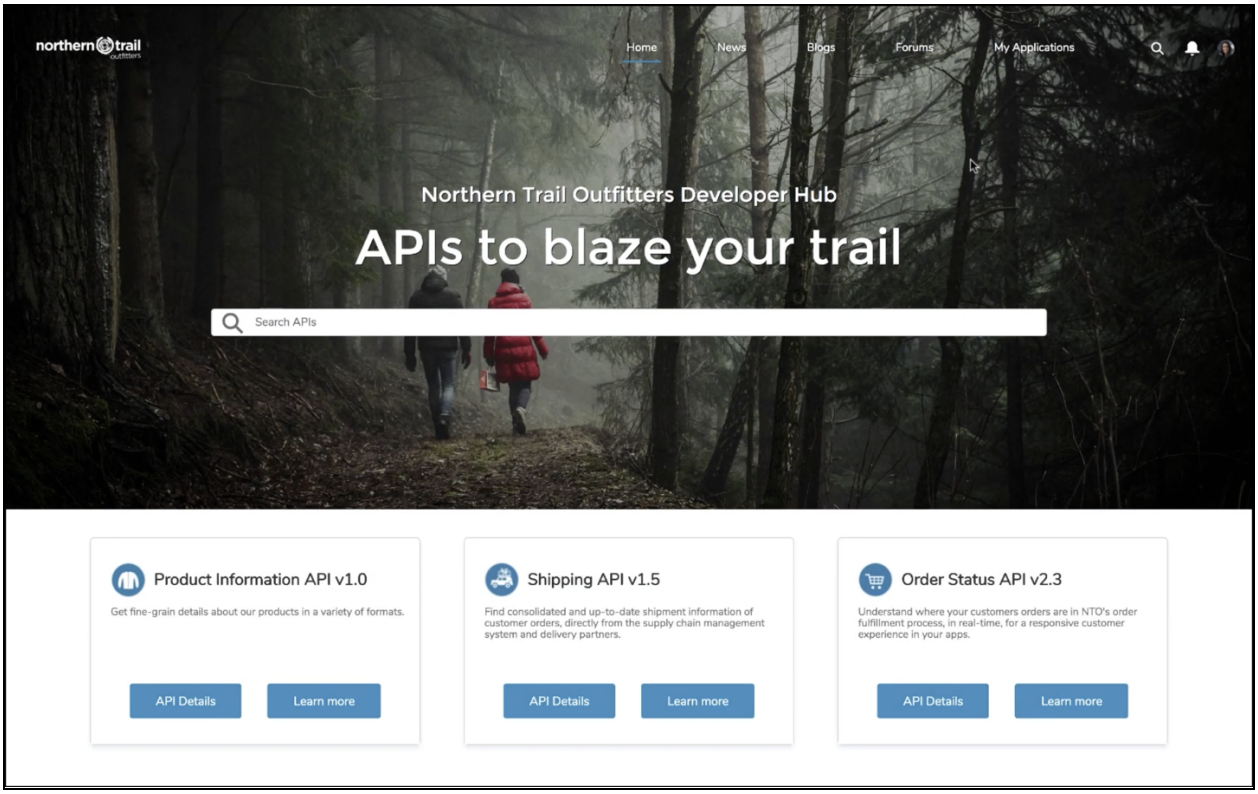


Exchange documentation can be multiple pages. So far we have just created the home page for our API. It is common practice to include links to other content like video tutorials in our Exchange documentation. This makes it easier for the interested developer to gain access to all of the documentation required to learn how to use the API.

Step 4: Explore API Portals

Anypoint Platform provides a default Public Portal via Exchange along with the possibility to create personalized digital experiences for the internal or external consumers of your API products via API Community Manager.

1. See this short video:



2. Also review the following API Communities from our customers:
   - Salesforce Commerce Cloud
   - KONE

Congratulations! You have completed Module 1.

Please proceed to Module 2.

Take me to the TOP