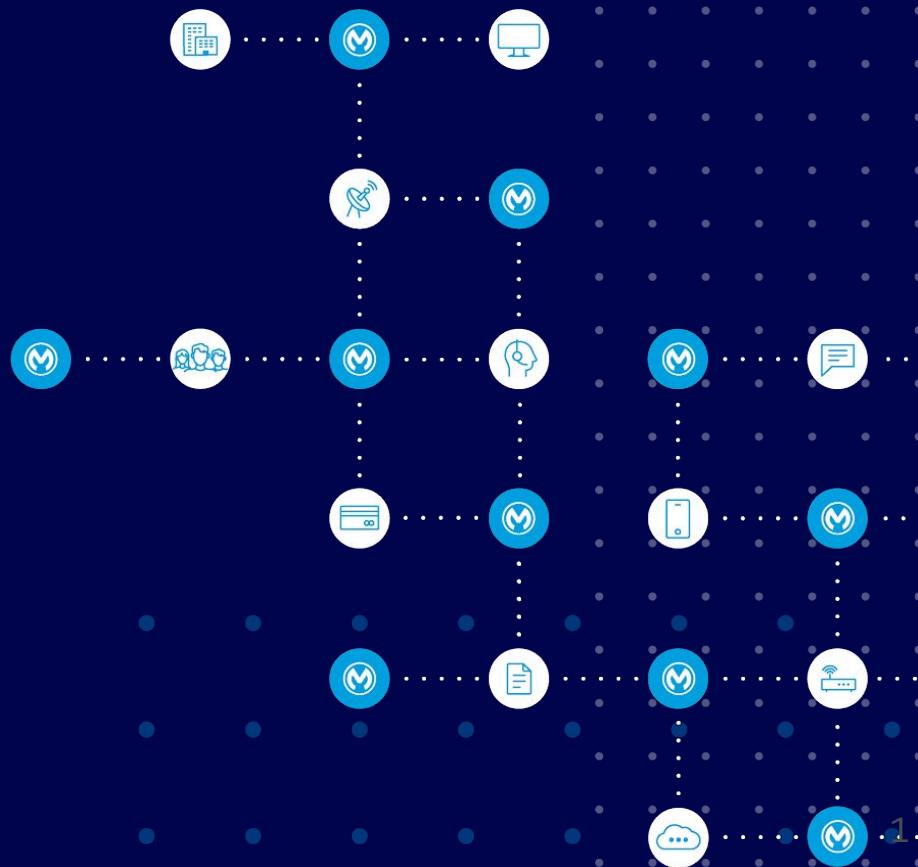




Welcome!

Thank you for joining this API-led Connectivity
Workshop for Direct to Consumer (DTC)

Module Lab Guide



Thank you!



Module 0: Prerequisites

Module 0: Prerequisites



Overview

Welcome to the MuleSoft API Led Connectivity Workshop, In this lab we will show you the prerequisites setup that are needed to start the workshop.

Following Workshop Instructions

Functions or actions to be performed by you (for example, menu selections, buttons to click, ...) are for the most part in **bold**.

Text to be entered will be in a *italic* font.

The general format for the workshop docs is to provide you instructions, followed by a screen capture of what you are doing.

1. Click on some button
2. Enter some text
3. Click Save
4. ...

Module 0: Prerequisites



Workshop Resources

1. **Guide Content:** The guide is also available at the following: <https://muleyoscar.github.io/dtc-workshop/>
2. **Lab Examples:** from code snippets to complete projects can be found in the Git repo:
<https://github.com/muleyoscar/dtc-workshop>
3. If you can not access your virtual machine and would still like to follow along you can download the **Anypoint Studio** here: <https://www.mulesoft.com/lp/dl/studio>



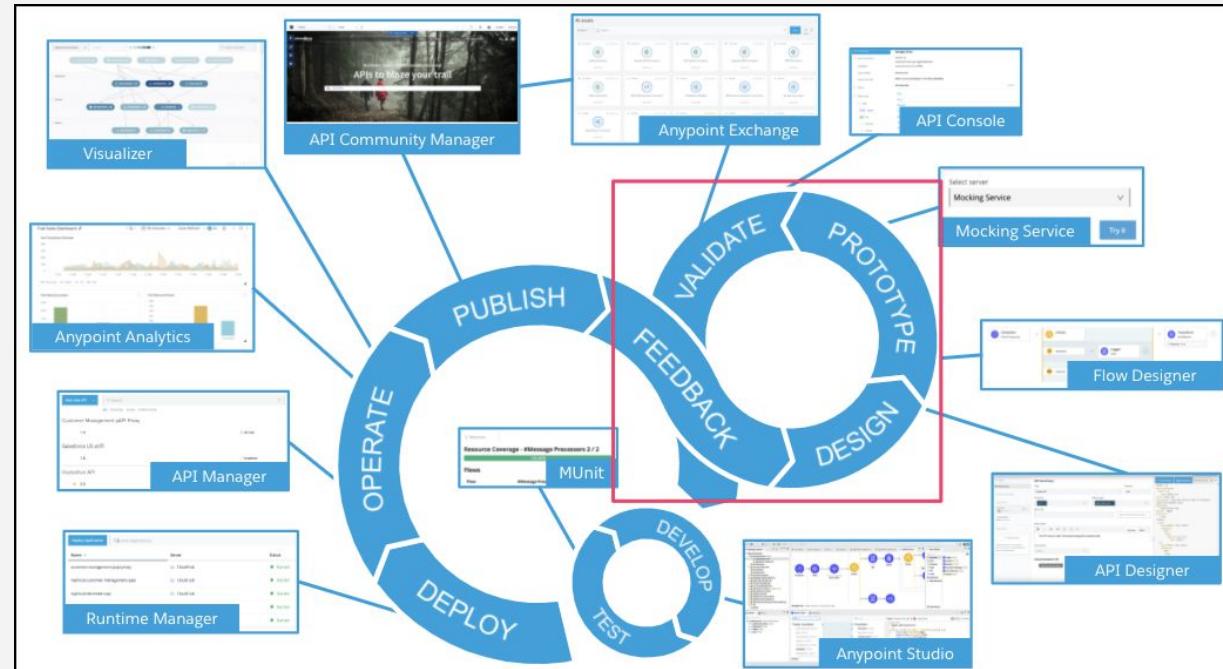
Module 1: Modern API Design: Overview

Module 1: The Modern API - Design: Overview



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 adopt a "design first" approach to creating API's. 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.

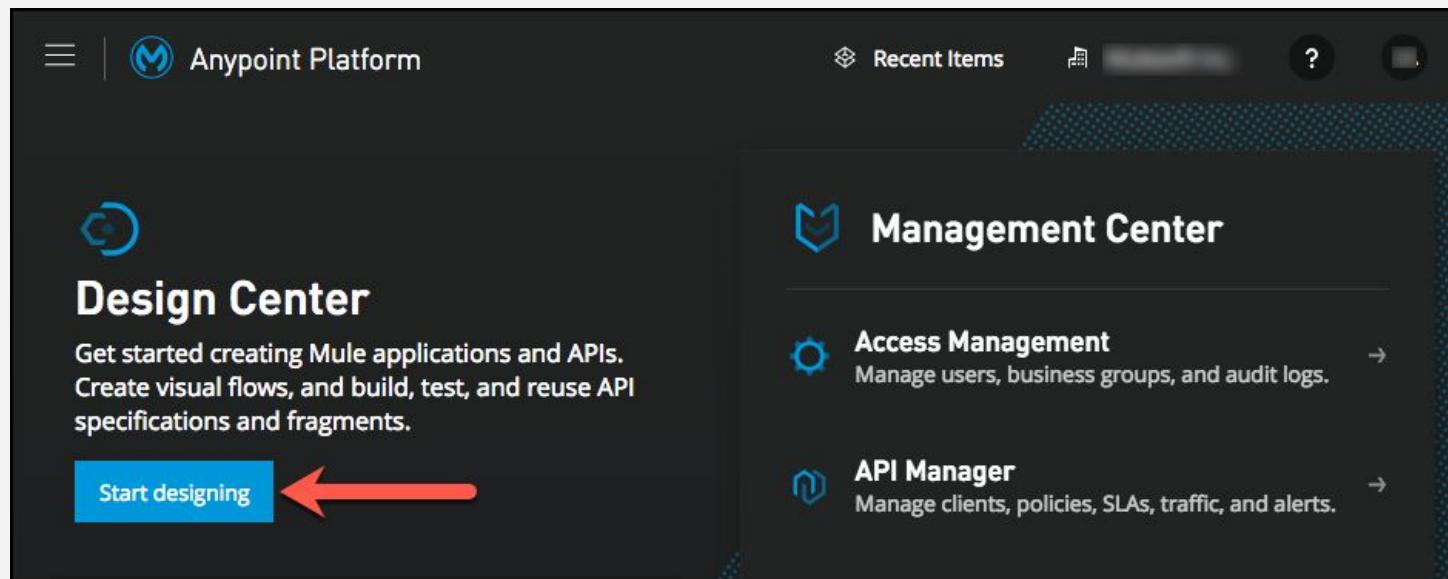


Module 1: The Modern API - Design: Overview



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

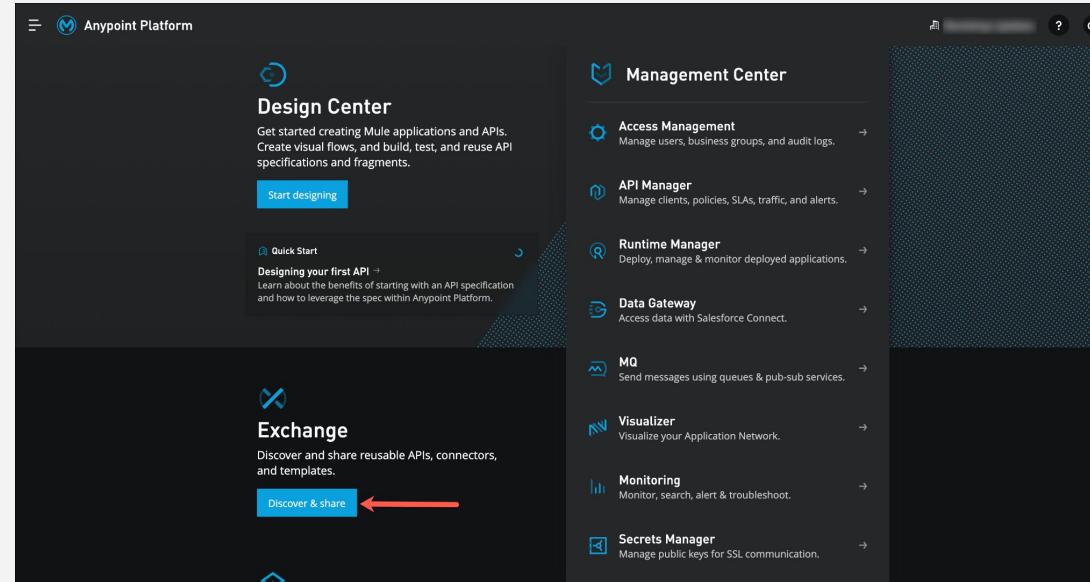
Anypoint Design Center provides a set of interactive tools that enable API designers to create or modify the specifications for an API. These specifications form the underlying API contract between the provider and the consumer of the API. **Anypoint Design Center** also enables the development of API Portals, documentation used to communicate to developers how the API should be consumed.



Module 1: The Modern API - Design: Overview



API discovery and collaboration is provided by **Anypoint Exchange** and simplifies the way that reusable assets are discovered and consumed across the enterprise. We will explore the features of **Anypoint Exchange** in detail in this first module.



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 3 labs in this unit are:

- [Lab 1: Search for an API in Exchange](#)
- [Lab 2: Design the Shopify API](#)
- [Lab 3: Publish the Shopify API to Exchange](#)

[Proceed to Lab 1](#)



Module 1: Lab 1 Search For an API in Exchange

Lab 1: Search for an API in Exchange

Overview

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

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

Module 1: The Modern API - Design: Lab 1 (2/14)



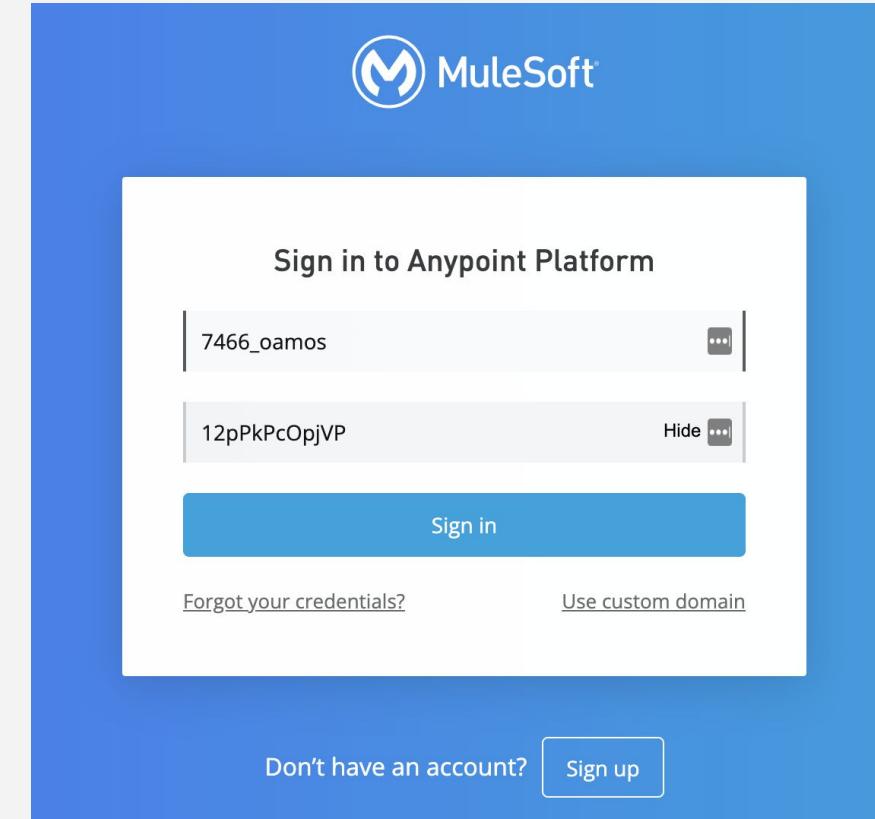
Step 1: Login to Anypoint Platform

1. Go to <http://anypoint.mulesoft.com>
2. Enter your credentials into the Username and Password fields.

You should have received an email from demos@mulesoft.com

with contents similar to the below:

The image shows an email inbox with one message titled "Workshop DTC-Workshop-DS Provisioning Result". The message is from "demos@mulesoft.com" and was sent 9:14 AM (5 hours ago). It contains the following text:
Your Anypoint Platform Credentials for the workshop are username:
7466_oamos and password: **12pPkPcOpjVP**



WARNING: If you already have an Anypoint login, the system will ask if you want to join the DTC Workshop with your existing account. DO NOT use your existing account select “Join with new account” option below the message.

Module 1: The Modern API - Design: Lab 1 (3/14)



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

A screenshot of the Anypoint Platform landing page. It features a dark header with the MuleSoft logo and the text "Anypoint Platform". Below the header, there are two main sections: "Design Center" and "Exchange". The "Design Center" section includes a "Start designing" button and a "Quick Start" card titled "Designing your first API". To the right, a sidebar lists various tools: Management Center, Access Management, API Manager, Runtime Manager, Data Gateway, MQ, Visualizer, and Monitoring. A large blue arrow points from the left towards the "Design Center" section.

4. Click on the Business Group that is on the right top corner of the screen. The name will vary with the class. Select sub group that has “DTC Workshop...” labelled.

Two screenshots illustrating the selection process. The left screenshot shows the Anypoint Platform landing page with the "Design Center" and "DataGraph" sections. The right screenshot shows the "Business Groups" interface, which displays a tree structure under the heading "- Master - PUBLIC". One node, "DTC Workshop DS", is circled in red. A red arrow points from the "DTC Workshop DS" label in the left screenshot to the same node in the right screenshot.

Module 1: The Modern API - Design: Lab 1 (4/14)



Step 2: Access Anypoint Exchange

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

The screenshot shows the Anypoint Platform dashboard. On the left, there's a sidebar with icons for Design Center, Exchange, and other tools. The Exchange section is highlighted with a red arrow pointing to the 'Discover & share' button. The main content area shows various management and developer tools like Management Center, API Manager, and Data Gateway.

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

The screenshot shows the Anypoint Exchange portal. The left sidebar has a 'Publish new asset' button and navigation for 'All assets' and 'All PUBLIC'. The main area is titled 'All PUBLIC assets' and displays a grid of 16 API cards. Each card includes a thumbnail, name, rating (5 stars), and owner information. Examples include 'Shopify Experience API', 'omni-channel-api', 'order-fulfillment-api', and 'order-api'.

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.

Module 1: The Modern API - Design: Lab 1 (5/14)



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.

A screenshot of the Anypoint Exchange web interface. The top navigation bar includes a 'DTC Workshop DS' organization name, a help icon, and a user icon. On the left, a sidebar shows navigation links like 'All assets', 'All PUBLIC' (which is circled in red), 'DTC Workshop DS', 'Provided by MuleSoft', 'Shared with me', 'My applications', and 'Public portal'. The main content area has a heading 'All PUBLIC assets' (also circled in red) and a search bar. Below are four cards representing software assets:

- Shopify Experience API: REST API, 5 stars, Workshop Owner
- omni-channel-api: Example, 5 stars, Workshop Owner
- order-fulfillment-api: Example, 5 stars, Workshop Owner
- order-api: Example, 5 stars, Workshop Owner

Module 1: The Modern API - Design: Lab 1 (6/14)



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 3 and Lab 4.

- **My applications:** These are the clients registered in the organization that are consuming the APIs that are deployed in the organization.
- **Public Portal:** You can define a public portal to let any user to consume any public org API that the organization may publish.

A screenshot of the MuleSoft Exchange interface. The top navigation bar includes 'Exchange', 'DTC Workshop DS', a help icon, and a workspace switcher. On the left, a sidebar shows organizational filters: 'All assets' (selected), 'All PUBLIC', 'DTC Workshop DS', 'Provided by MuleSoft', 'Shared with me', 'My applications' (circled in red), and 'Public portal'. The main area displays a grid of API assets under 'All PUBLIC assets'. Each asset card includes a thumbnail, name, rating, and owner information. The first asset is 'Shopify Experience API' (Workshop Owner).

Name	Type	Rating	Owner
Shopify Experience API	REST API	★★★★★	Workshop Owner
omni-channel-api	Example	★★★★★	Workshop Owner
order-fulfillment-api	Example	★★★★★	Workshop Owner
order-api	Example	★★★★★	Workshop Owner

Module 1: The Modern API - Design: Lab 1 (7/14)



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 API's. **Anypoint Exchange** is the enterprise repository for the following types of reusable software assets:

A screenshot of the Anypoint Exchange portal. The interface has a dark header with the title 'Exchange'. On the left, there is a sidebar with navigation options: 'All assets' (selected), 'All PUBLIC', 'DTC Workshop DS', 'Provided by MuleSoft', 'Shared with me', 'My applications', and 'Public portal'. The main area shows a grid of API assets. A red arrow points to the search bar, which is labeled 'Search' and has a dropdown menu 'All types'. The first asset listed is 'Shopify Experience API' with a green icon and a rating of 5 stars. Other visible assets include 'omni-channel-api', 'order-fulfillment-api', and 'order-api', all rated 5 stars and owned by 'Workshop Owner'.

Asset Name	Type	Rating	Owner
Shopify Experience API	REST API	★★★★★	Workshop Owner
omni-channel-api	Example	★★★★★	Workshop Owner
order-fulfillment-api	Example	★★★★★	Workshop Owner
order-api	Example	★★★★★	Workshop Owner

Module 1: The Modern API - Design: Lab 1 (8/14)



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

The screenshot shows the MuleSoft Exchange interface. On the left, there's a sidebar with various navigation options like 'Exchange', 'Publish new asset', 'All assets', 'All PUBLIC', etc. The main area is titled 'All PUBLIC assets'. It features a search bar at the top. Below it, there's a dropdown menu labeled 'All types' with several options: Connectors, Templates, Examples, Policies, API Groups, REST APIs, SOAP APIs, DataWeave Libraries (BETA), AsyncAPIs, HTTP APIs, API Spec Fragments, and Custom. The 'All types' option is highlighted with a red arrow. Below the dropdown, there are cards for two APIs: 'omni-channel-api' and 'customer-api', both listed as 'Workshop Owner'.

- **Connectors** - Packaged connectivity to an endpoint developed and deployed on MuleSoft's Anypoint Platform with third-party APIs and standard integration protocols.
- **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.
- **Examples** - Implementation projects that explain development elements within Anypoint Studio and how these can be leveraged to achieve specific API and integration objectives.
- **Policies** - Configuration modules to extend the functionality of an API and enforce capabilities such as security.
- **API Groups** - A set of APIs bundled into a single asset. Instead of requesting access to multiple APIs to satisfy a use case, a developer can access the group in one step.
- **REST APIs** - API descriptions in RAML format that make the consumption of REST API's faster and easier.
- **SOAP APIs** - API descriptions in WSDL format that make the consumption of a SOAP API's faster and easier.
- **HTTP APIs** - A placeholder for an endpoint for use by private Exchange users who want to manage the endpoint with API Manager
- **API Spec Fragments** - Shared fragments of RAML files that can be used to assemble new API's with common traits.
- **Custom** - A general category for sharing resources such as links, blogs, articles, videos and more.

Module 1: The Modern API - Design: Lab 1 (9/14)



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 "**shop**", then hit the **Enter Key**.
8. The list of Exchange assets should now be filtered to look something like this:

A screenshot of the MuleSoft Exchange interface. The top navigation bar shows the title "Exchange" and a workspace named "DTC Workshop DS". On the left, a sidebar menu includes options like "Publish new asset", "All assets" (selected), "All PUBLIC" (highlighted with a red border), "DTC Workshop DS", "Provided by MuleSoft", "Shared with me", "My applications", "Public portal", and "Settings". The main content area is titled "All PUBLIC assets". A search bar at the top has a dropdown labeled "All types" and a search input field containing "shop", which is also highlighted with a red border. Below the search bar, a message says "Showing results for \"shop\"." and a "Save this search" button. Three API assets are listed in cards:

- Shopify Experience API**: REST API, 5 stars, Workshop Owner
- Omni Channel Experience API**: REST API, 5 stars, Workshop Owner
- Product API**: REST API, 5 stars, Workshop Owner

Module 1: The Modern API - Design: Lab 1 (10/14)



9. Click on the tile named **Shopify Experience API**.

The screenshot shows the MuleSoft Exchange interface with a search bar containing 'shop'. The results list three items: 'Shopify Experience API', 'Omni Channel Experience API', and 'Product API'. The 'Shopify Experience API' tile is highlighted with a red border.

You should now see the page dedicated to the Shopify API definition:



The screenshot shows the detailed page for the 'Shopify Experience API'. It includes a 'PAGES' sidebar with 'Home' selected. The main content area features a diagram illustrating the API's integration with various systems such as Shopify, Order Fulfillment API, Notifications, Orders, Products, Customers, and more. A large blue arrow points from the previous screenshot to this detailed page.

Module 1: The Modern API - Design: Lab 1 (11/14)



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. We will learn how to create this content latter. But for now, let's do a quick survey of the key features of the Exchange portal:

A screenshot of the MuleSoft Exchange portal. The main page displays the 'Shopify Experience API' asset. The interface includes a sidebar with navigation links like 'Assets list', 'PAGES', 'Home', 'SPECIFICATION', 'Summary', 'Endpoints', and 'OTHER DETAILS'. The main content area shows a large diagram illustrating the architecture of the Shopify Experience API. It features a central 'Shopify API' node connected to various components: 'Shopify' (represented by a green shopping bag icon), 'Order Fulfillment API' (blue camera icon), 'Notification' (envelope icon), 'Orders' (document icon), 'Products' (cloud icon), 'Customers' (person icon), and 'Campaigns' (globe icon). Below this diagram, there are two mobile device screenshots showing the 'northern trail' app interface. At the bottom of the page, there is a 'Orders' section with a table showing order details. The top right of the page has buttons for 'Edit documentation', 'Share', 'Download', 'View code', and 'Add version'. Version information at the top right indicates '2.0.1 Private' and 'Latest: 1.0.0 Stable'.

Module 1: The Modern API - Design: Lab 1 (12/14)



The screenshot shows the NTO developer portal interface. At the top, there's a navigation bar with 'Exchange', 'PUBLIC', a question mark icon, and 'OA'. Below it, a header for 'Shopify Experience API' includes a green icon with a white 'S', a title, and a subtitle: 'Direct-to-Consumer Shopify API will allow NTO to reach their consumers directly!'. A red box labeled 'A.' highlights this area. Below the header, there are links for 'REST API', 'DTC Workshop DS', 'Oscar Amos', and 'Updated 3 hours ago'. A message from 'Workshop Owner' is shown, along with a timestamp and a 'Tags (2) v' link. On the left, a sidebar has 'Assets list' selected and 'Edit documentation' available. The main content area contains a welcome message: 'Welcome to the NTO developer portal. This API represents the Direct to Consumer Shopify Shopping Cart Application - that includes: order, product and inventory tracking.' A red box labeled 'B.' highlights the top right options: 'Edit', 'Share', 'Download', 'View code', and 'Add version'. Another red box labeled 'C.' highlights the version management section, which shows '2.0.1 Private' as the current version, '1.0.x' as the previous version, and a dropdown menu for 'Latest 1.0.0' with 'Stable' selected. A small arrow icon is next to the dropdown.

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

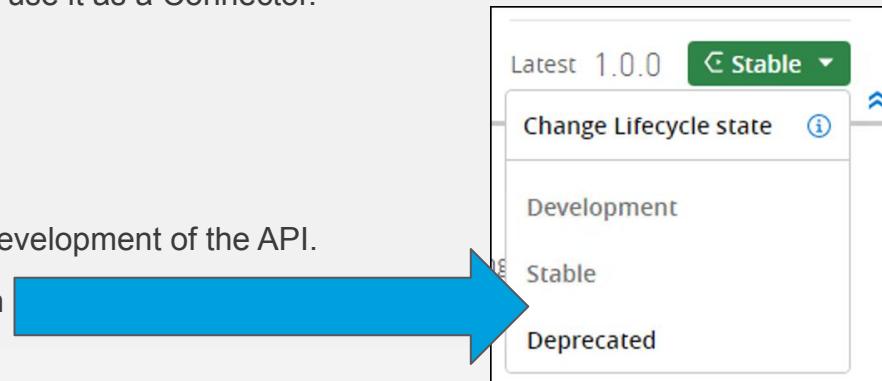
B. **Options Panel** - The options panel has three options.

- **Edit** (pencil icon): This option allows to edit the API portal.
- **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.
- **View Code**: Opens Design Center.
- **Add Version**: Lets you create a new version.

C. **Versions** It shows the different versions of the API.

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

- **Change Lifecycle State**: Lets you change the state of the version



Module 1: The Modern API - Design: Lab 1 (13/14)



The screenshot shows the MuleSoft Exchange interface for the 'Shopify Experience API'. The top navigation bar includes 'Exchange', 'PUBLIC', a question mark icon, and 'OA'. Below the title 'Shopify Experience API' (with a green icon), there's a description: 'Direct-to-Consumer Shopify API will allow NTO to reach their consumers directly!'. It lists 'REST API', 'DTC Workshop DS', 'Oscar Amos oamos@nto-demo.com', and 'Updated 3 hours ago'. On the right, there are buttons for 'Edit' (red box D), 'Share', 'Download', 'View code', 'Back to previous UI' (red box D), 'Add version' (red box E), and 'Tags (2)' (red box F). A 'Manage versions' section shows '2.0.1 Private' and '1.0.x'. Below it, 'Latest 1.0.0' is marked as 'Stable'. The left sidebar has 'Assets list' and a menu with 'PAGES' (selected) and 'Home'. The main content area contains a welcome message: 'Welcome to the NTO developer portal. This API represents the Direct to Consumer Shopify Shopping Cart Application - that includes: order, product and inventory tracking.' and 'The Shopify API is an Experience API that is a part of NTO's API led Architecture . This same architecture allows NTO to leverage a headless commerce platform that is highly composable.'

- D. **Back to previous UI** allows you to toggle between our classic or newer Portal UI (defaulted at newer UI)
- E. **Add Version** is where you can easily upload a newer version of the Shopify API specification via RAML or OAS file upload
- F. **Add Tags** - *Tags* make it easier to search and discover API's. Rather than relying on the title or description of the API, tags allow for you to create categories of API's related to each other by *tags*.
- G. **Edit Documentation** Lets you edit the documentation for your API.

Module 1: The Modern API - Design: Lab 1 (14/14)



Workshop Owner published 16 hours ago

Manage versions | 2.0.1 Private | 1.0.x | Latest 1.0.0 Stable

Tags (2) v

Assets list

Home H.

SPECIFICATION

Summary

Endpoints

/order

/products

/({id})

/inventory

/location/{id}/{quantity}

OTHER DETAILS

API Instances

Welcome to the NTO developer portal. This API represents the Direct to Consumer Shopify Shopping Cart Application - that includes: order, product and inventory tracking. The Shopify API is an Experience API that is a part of NTO's API led Architecture . This same architecture allows NTO to leverage a headless commerce platform that is highly composable.

Shopify App Example: The primary use of the ShopifyExperience API is to accelerate NTO's direct-to-consumer vision!

Shopify App Example: The primary use of the ShopifyExperience API is to accelerate NTO's direct-to-consumer vision!

API Tools

This API is designed using RAML which makes it easy to discover its capabilities and use with powerful tools.

- API Console is a graphical user interface for a RAML-defined API that visually exposes the API's structure and important patterns and serves as interactive API documentation.
- SDK Generator is a utility to generate an SDK in various languages. Developer SDKs are provided to make it easy to use the NTO Shopify API with your application.
- API Notebooks are javascript code snippets that provide an example of common API use cases. These notebooks help developers see how the Mythical API can be used to accomplish common tasks.

Reviews I.

Be the first to review Shopify Experience API 1.0.x

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

- You can expand the resource endpoints (e.g. orders) to view the operations that can be performed on that endpoint (e.g. GET, POST, DELETE):

Endpoints

/order

Overview

GET

PATCH

POST

I. 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.

Reviews

Oscar Amos | Apr 14, 2022, 7:45 AM

★★★★★ Just what I was looking for?

Awesome stuff!!!

Edit Delete

Proceed to Lab 2



Module 1: Lab 2 Design the Shopify API

Module 1: The Modern API - Design: Lab 2 (1/20)



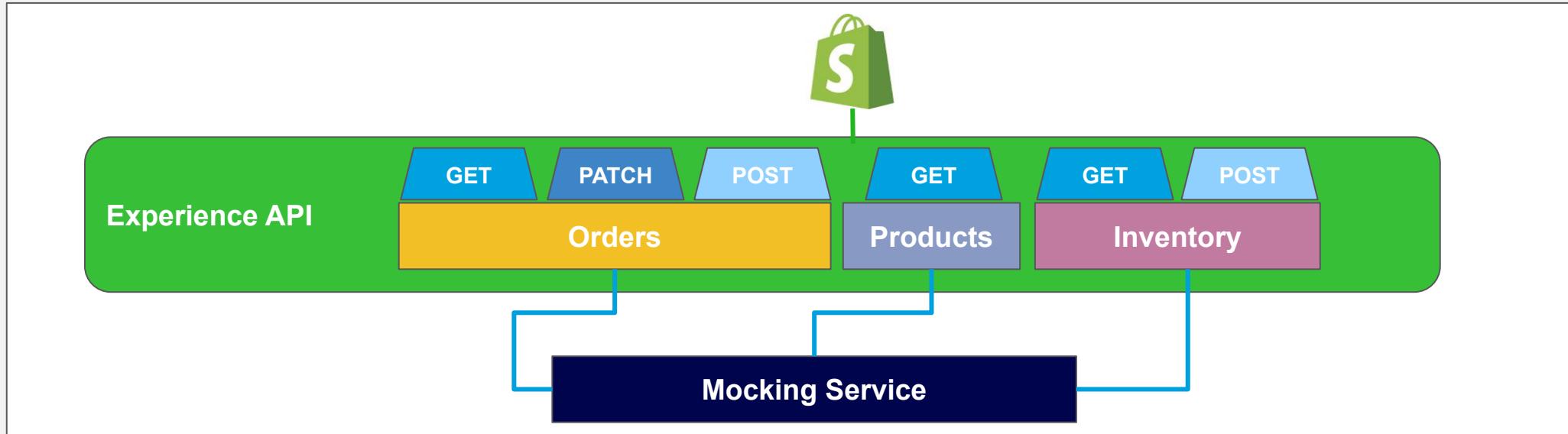
Overview

In the first lab of this module we learned how to discover the Shopify 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 Shopify 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 API's using REST. RESTful API's 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.

Module 1: The Modern API - Design: Lab 2 (2/20)



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.

Module 1: The Modern API - Design: Lab 2 (3/20)



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.

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

Module 1: The Modern API - Design: Lab 2 (4/20)



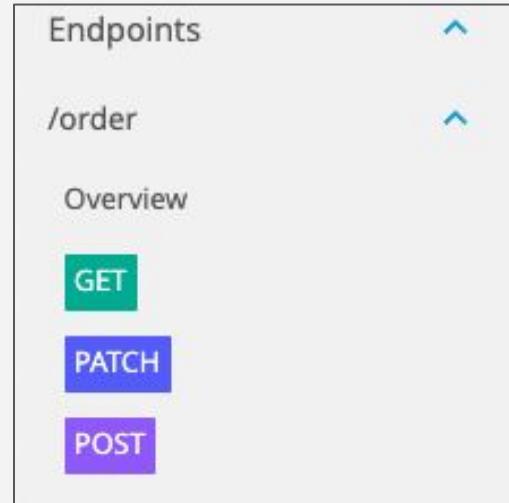
Collection Resources

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

- GET (get the list)
- PATCH (can both create new order records or modify existing order record to specified target)
- POST (will create order(s) at its intended target)



Instance Resources

Instance resources contain the details of a specific resource, i.e. the instantiation of a resource. For our inventory example that might include a product id and id of the store. Highly valuable are then links to other parameters (e.g. quantity) that might be part of that inventory management process.

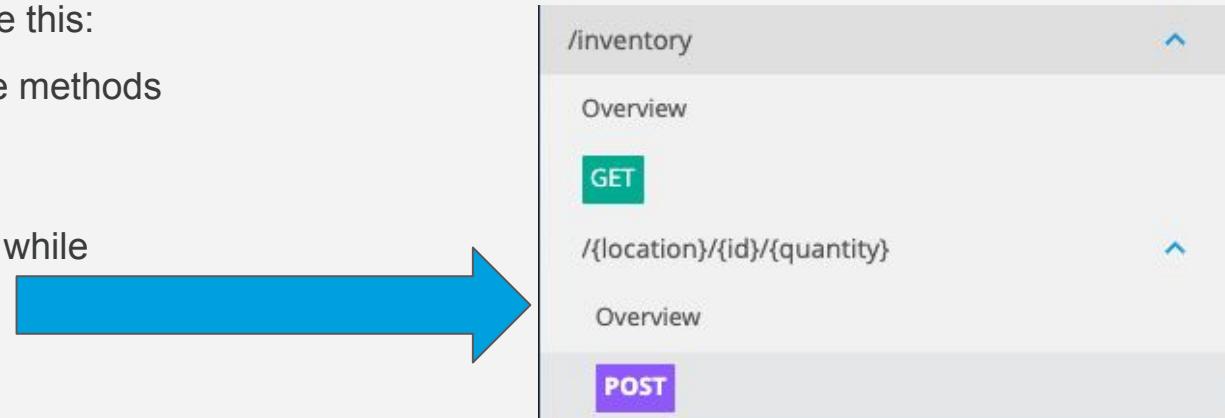
Instance resources must be uniquely identify-able and typically look like this:

`/inventory/{location_id}/{product_id}/{quantity_value}`. Typically the methods

- GET (retrieve an inventory list of products)
- POST (update inventory data by location and specific product while specifying quantity to update)

are available on the inventory resource.

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



Module 1: The Modern API - Design: Lab 2 (6/20)



Step 1: Review the E-Commerce Application Requirements

You have a new requirement for a new Shopify application. The NTO business team 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.

Functionality:

- Products:
 - List all the products available
 - Name
 - Description
 - Product Variants by SKU
 - Category/Brand
 - Get information about a specific product by identifier (ID)
- Orders:
 - List all consumer orders made in Shopify
 - Get information of an Order by its Order ID
 - Create new consumer order from other systems and post to Shopify for status updates via Shopify:
 - Bulk update and add new orders into Shopify
- Inventory:
 - List all product inventory across all NTO store locations
 - Update a specific product inventory level at certain store location

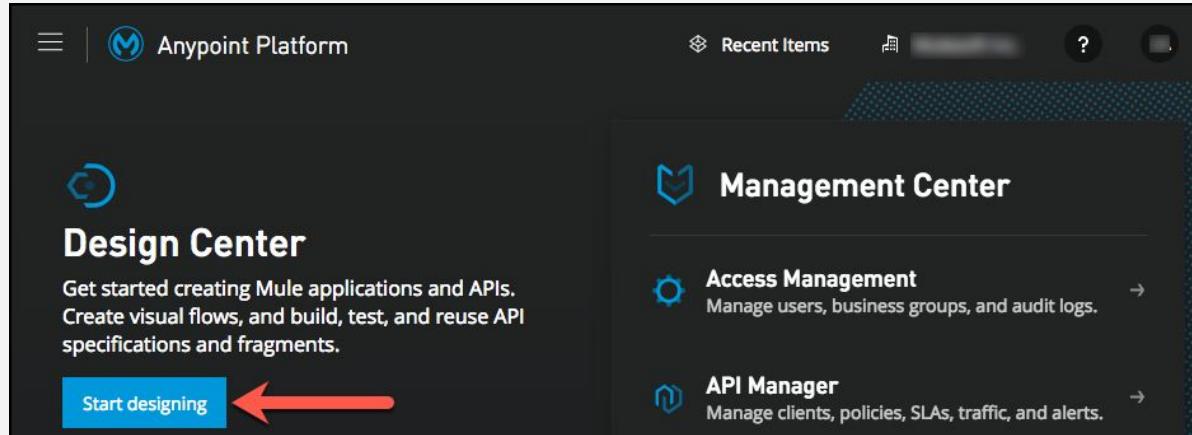
Module 1: The Modern API - Design: Lab 2 (7/20)



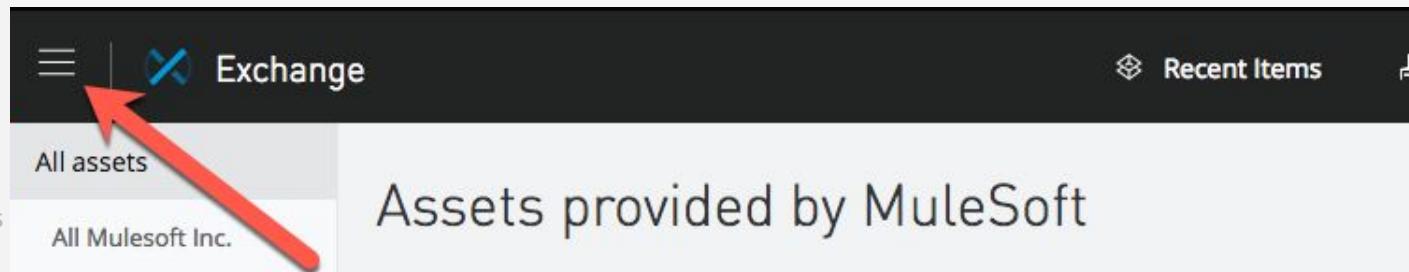
Step 2: Create the Shopify Experience API

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:



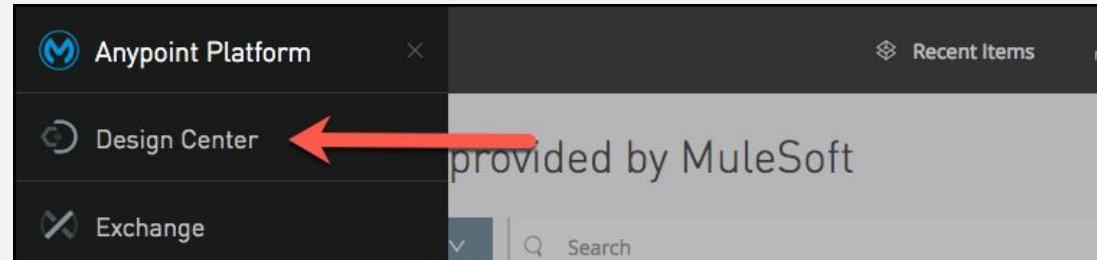
- Alternatively you can use the "hamburger menu" from any page within Anypoint Platform portal to navigate to Design Center:



Module 1: The Modern API - Design: Lab 2 (8/20)



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



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

A screenshot of the Anypoint Design Center. On the left, there is a table titled 'Projects' listing several API specifications. On the right, a detailed view of the 'Shopify Experience API' is shown, including its name, creation date, and creator information. A blue 'Open' button is visible at the bottom of the details panel.

Name	Project Type	Last Update	Created By
Shopify Experience API	API specification	April 13th, 2022	7466_oamos
Omni Channel Experience API	API specification	April 13th, 2022	7466_oamos
Product API	API specification	April 13th, 2022	7466_oamos
Social Network API for Facebook	API specification	April 13th, 2022	7466_oamos
Order API	API specification	April 13th, 2022	7466_oamos
Order Fulfillment API	API specification	April 13th, 2022	7466_oamos

Module 1: The Modern API - Design: Lab 2 (9/20)



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

A screenshot of the MuleSoft Design Center interface. At the top, there's a dark header bar with the 'Design Center' logo and some status icons. Below it is a search bar labeled 'Filter by project name' and a 'Create new' button. A large red arrow points to the 'Create new' button. The main area shows a table of 'Projects' with columns for 'Name', 'Project Type', 'Last Update', and 'Created By'. Two projects are listed: 'Shopify Experience API' (API specification, last updated April 13th, 2022, created by 7466_oamos) and 'Omni Channel Experience API' (API specification, last updated April 13th, 2022, created by 7466_oamos). To the right of the table is a small graphic icon.

Name	Project Type	Last Update	Created By
Shopify Experience API	API specification	April 13th, 2022	7466_oamos
Omni Channel Experience API	API specification	April 13th, 2022	7466_oamos

4. Select “New API Spec”:

A screenshot of a dropdown menu titled 'Create new'. The menu items are: 'New Project from scratch', 'New API Spec' (which has a red arrow pointing to it), 'New Fragment', and 'New Mule App'. Each item has a brief description below it.

- New Project from scratch
- New API Spec**
Define how the API will behave and how it communicates with other systems & APIs
- New Fragment
Compose snippets that can easily be reused across multiple APIs
- New Mule App
Design a Mule integration flow

Module 1: The Modern API - Design: Lab 2 (10/20)



5. Set the following values in the Add API form:

- A. API name: <your name or initials> - Shopify Experience API.
- B. How do you want to draft the API Spec: Select Guide me through it radio button.
- C. Select Create API Spec button.

New API Spec

API Title
A.

How do you want to draft the API Spec?

I'm comfortable designing it on my own
A complete code editing experience with interactive documentation

Specification Language

B. Guide me through it
Use a visual interface scaffolding the API Specification (can generate both RAML & OAS)

C.

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.

Module 1: The Modern API - Design: Lab 2 (11/20)



6. After clicking **Create API Spec**. 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.

The screenshot shows the MuleSoft API Designer interface. The title bar reads "Design Center" and "OAA-Shopify Experience API". The main area is titled "API Summary" and contains fields for "Title" (set to "New API"), "Version" (empty), "Protocols" (dropdown set to "Select..."), "Media type" (dropdown set to "Select..."), "Base URI" (empty), and "Description" (rich text editor in Markdown mode). On the right, a sidebar displays the RAML code: "#%RAML 1.0 title: New API". The left sidebar includes sections for "SECURITY SCHEMES", "RESOURCES", "DATA TYPES", and "Documentation (0)". A bottom note says "Create groups to semantically group resources and data types in your API specification." The top right features "Publish" and "Download" buttons.

Module 1: The Modern API - Design: Lab 2 (12/20)



Step 3: Create the Orders List Resource

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 add some general details about the API.

- A. Name the API by setting the title to:

<your name or initials>- Shopify Experience API.

- B. Since this is our first design let's set API Version to **“1.0”**

- C. Select protocol as **“HTTP”**

- D. Set the media type to **“application/json”**.

- E. 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 something like following in your API design:

The screenshot shows the Anypoint Design Center interface with the title "Design Center" at the top. The main area is titled "API Summary". A RAML code snippet on the right side defines the API details. The configuration fields are highlighted with red boxes and labeled A through E:

- A.** Title: OAA-Shopify
- B.** Version: 1.0
- C.** Protocols: HTTP
- D.** Media type: application/json
- E.** Base URI: http://localhost:8081/api

The RAML code on the right is:

```
#%RAML 1.0
title: "OAA-Shopify"
baseUri: http://localhost:8081/api
mediaType:
  - application/json
version: "1.0"
protocols:
  - HTTP
```

Module 1: The Modern API - Design: Lab 2 (13/20)



Step 3: Create the Orders List Resource

- 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**.

The screenshot shows the MuleSoft Anypoint Studio API Designer interface. The top navigation bar includes 'Design Center', 'API Summary', 'OAA-Shopify Experience API', 'Publish', and settings. The left sidebar has sections for 'API Summary', 'SECURITY SCHEMES', 'RESOURCES' (which is highlighted with a red box and has a red arrow pointing to the '+ Add resource' button), 'DATA TYPES', and 'Documentation'. The main area displays the 'API Summary' configuration with fields for Title ('OAA-Shopify'), Version ('1.0'), Protocols ('HTTP'), Media type ('application/json'), and Base URI ('http://localhost:8081/api'). To the right, the RAML code is shown:

```
%%RAML 1.0
title: "OAA-Shopify"
baseUri: http://localhost:8081/api
mediaType:
  - application/json
version: "1.0"
protocols:
  - HTTP
```

Module 1: The Modern API - Design: Lab 2 (14/20)



Step 3: Create the Orders List Resource

3. To create the order resource for the Shopify API we will need to do the follow:

- A. Rename the resource from **/new-resource** to **/order**.
- B. Select the **GET** option and under summary → description, give it a description
- C. Scroll to and select **Responses** tab below. It will have a default **200** Status (which signals successfully operation) preselected and hit **Add**
- D. Notice the generated RAML on the right.

The screenshot shows the MuleSoft Anypoint Studio Design Center interface for creating an API resource. The main area displays the 'OAA-Shopify Experience API' with a resource named '/order'. The resource details show a single GET method selected (labeled A). The 'Responses' tab is active (labeled B), showing a single 200 OK response with a status message and a description field. A red arrow points to the 'Add' button in the responses table (labeled C). To the right, the generated RAML code is displayed, and another red box highlights the RAML editor area (labeled C).

```
%RAML 1.0
title: "OAA-Shopify"
baseUri: http://localhost:8081/api
mediaType:
  - application/json
version: "1.0"
protocols:
  - HTTP
/order:
  get:
    responses:
      "200": {}
```

Module 1: The Modern API - Design: Lab 2 (15/20)

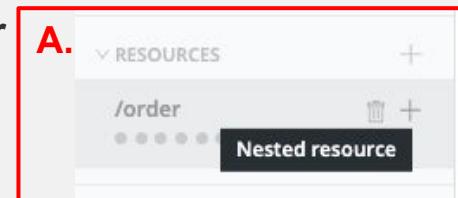


Step 4: Create the Order Instance Resource

In the previous step we created an Order collection resource and GET method to grab all Shopify Orders. But in most cases for NTO, information on a single order will be needed. We need to create a new resource underneath the Order instance and use the GET method.

1. So we will do the following:

- A. By the **order** resource hover your mouse on the resource and you should see a trash can and “+” icon. Under the “+” icon, **Nested resource** call out is visible to indicated we can nest child resource under **order**



- B. Click the “+” icon and you will see that another resource, **/order/new-resource** has been added under **order**.
- C. Change “**/order/new-resource**” to “**/order/{id}**” in the title pane
- D. Scroll to and select **Responses** tab below. It will have a default **200 Status** (which signals successfully operation) preselected and hit **Add**

The screenshot shows the MuleSoft Anypoint Studio Design Center interface. The top navigation bar includes 'Design Center', 'API Summary', 'OAA-Shopify Experience API', 'Edit RAML', 'Download', and 'Publish'. The main area displays the RAML specification for the API. On the left, there's a 'RESOURCES' tree with nodes for '/order' and '/order/{id}'. The '/order/{id}' node is expanded, showing its methods (GET, POST, PUT, PATCH, DELETE, OPTIONS, HEAD) and a 'Responses' tab. The 'Responses' tab for the GET method shows a 200 OK status with a 'Status' field containing '200 - OK' and a 'Description' field. A red box labeled 'C.' highlights the URL path '/order/{id}'. A red box labeled 'D.' highlights the 'Responses' tab and the 200 OK status entry.

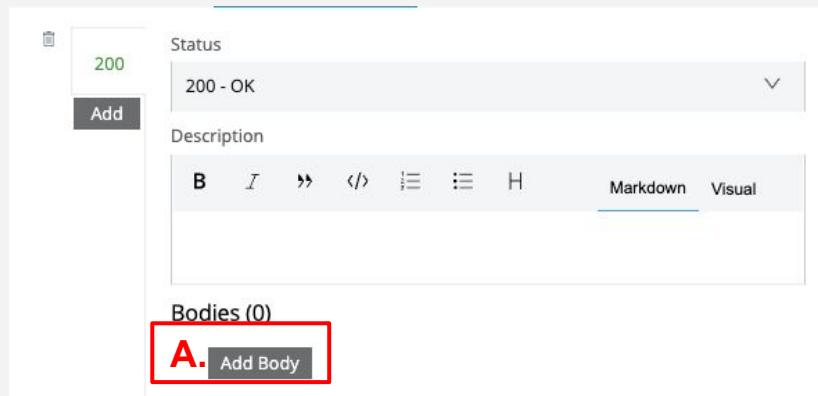
Module 1: The Modern API - Design: Lab 2 (16/20)



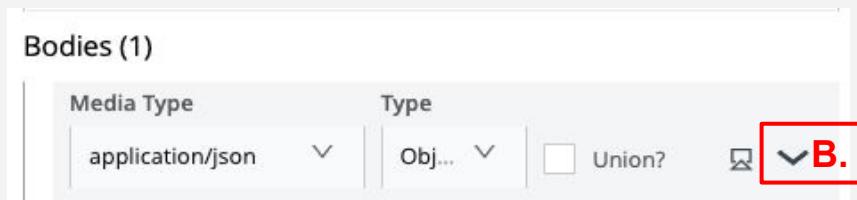
Step 4: Create the Order Instance Resource

2. Now let's add a specific body to the 200 response for `order/{id}` instance.

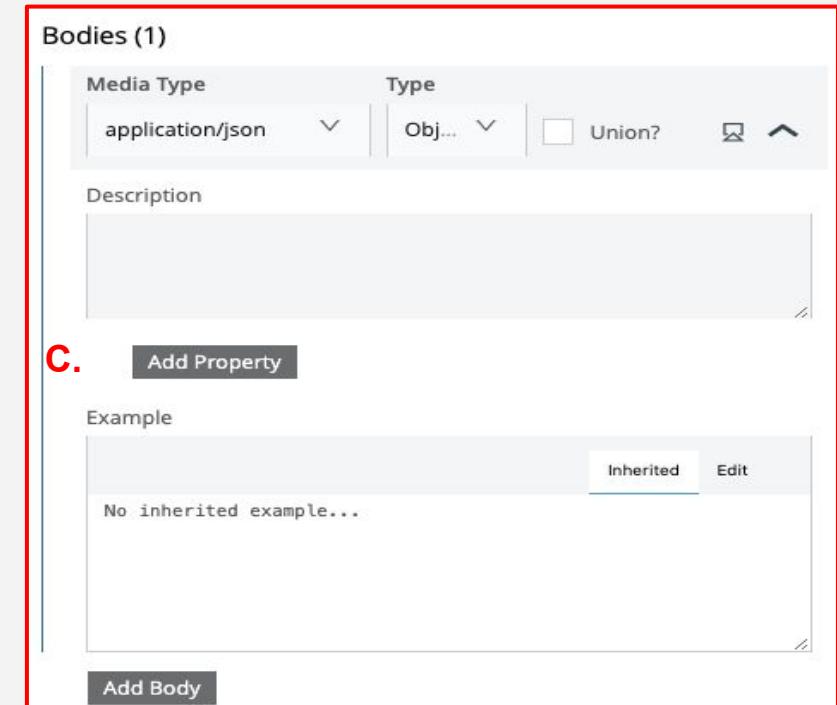
A. Select Add Body



B. Keep application/json as the Media Type and hover to the arrow icon at the right to expand options.



C. Once expanded it should look like this:



Module 1: The Modern API - Design: Lab 2 (17/20)



Step 4: Create the Order Instance Resource

3. Click **Edit**

4. Now let's copy and paste the following code into the body:

Bodies (1)

Media Type Type

Default (application/json) Object Union?

Description

Add Property

Example

Paste Here

Inherited Edit

{
 "id": 413413890297800,
 "confirmed": true,
 "fulfilled": false,
 "created": "2021-12-09T09:05:34-05:00",
 "currency": "USD",
 "total": 75.55,
 "customer": {
 "id": 1616616,
 "email": "jane@gmail.com",
 "address1": "567 Main St",
 "city": "Stafford",
 "state": "VA",
 "country": "USA",
 "zip": "22554",
 "phone": "555-710-9325",
 "line_items": [
 {
 "id": 10417352802515,
 "sku": "841483107332",
 "name": "Waist Cincher - Soft Nude / L",
 "quantity": 1,
 "gift card": false,
 "price": 75.55,
 "tax_code": "PC040100"
 }
]
 }
}

Add Body

If you have problems copying the code you can find example [here](#)

```
{  
  "id": 413413890297800,  
  "confirmed": true,  
  "fulfilled": false,  
  "created": "2021-12-09T09:05:34-05:00",  
  "currency": "USD",  
  "total": 75.55,  
  "customer": {  
    "id": 1616616,  
    "email": "jane@gmail.com",  
    "address1": "567 Main St",  
    "city": "Stafford",  
    "state": "VA",  
    "country": "USA",  
    "zip": "22554",  
    "phone": "555-710-9325",  
    "line_items": [  
      {  
        "id": 10417352802515,  
        "sku": "841483107332",  
        "name": "Waist Cincher - Soft Nude / L",  
        "quantity": 1,  
        "gift card": false,  
        "price": 75.55,  
        "tax_code": "PC040100"  
      }  
    ]  
  }  
}
```

Module 1: The Modern API - Design: Lab 2 (18/20)



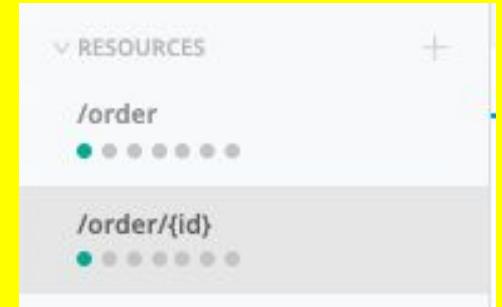
Step 5: Use Mocking Service

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 mocking service is a feature of the Anypoint Platform. You can simulate calls to the API in API Designer before publishing the API specification to Exchange or in Exchange after publishing the API specification. The mocking service will read the RAML specification of your API, create the API/service and return example data responses. This 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.

First we are going to enable the mocking service for internal use.

1. Click the Circular Arrow icon in the right-hand toolbar it should expand the pane like so:

If you see an empty pane check to make sure the `/order/{id}` resource is highlighted in the left pane like so:



Module 1: The Modern API - Design: Lab 2 (19/20)



Step 5: Use Mocking Service

2. From **Select Server** dropdown change the selection from <http://localhost:8081/api> to **Mocking Service**

3. Under **URI parameters (id*)** you can add any number you would like to.

4. Then click the **Send** Button.

5. You will notice at the bottom the sample response with the sample JSON code we copy and pasted in the order/{id} resource response body is showing. The mock response also shows a **200 OK** and **Time** output to simulate a the request and reply process for Shopify API.

Use the mocking service and check the examples is updated too.

The API design is ready. In the next lab we are going to publish the API so it can be discoverable by anybody in the organization.

```
1  [{"id": 413413890297800, "confirmed": true, "fulfilled": false, "created": "2021-12-09T09:05:34-05:00", "currency": "USD", "total": 75.55, "customer": {}}]
```

Module 1: The Modern API - Design: Lab 2 (20/20)



Summary

In this lab, you completed the following steps:

- Step 1: Review the Shopify Requirements
- Step 2: Create the Shopify Experience API
- Step 3: Create the Orders List Resource
- Step 4: Create the Order Instance Resource
- Step 5: Use the Mocking Service

We easily designed the framework for our Shopify API, providing the ability to get orders and check single order. 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.

The screenshot shows the MuleSoft Anypoint Platform Design Center. On the left, the 'API Summary' section displays details for the 'OAA-Shopify Experience API' version 1.0. It specifies 'HTTP' as the protocol and 'application/json' as the media type. The base URI is set to 'http://localhost:8081/api'. The 'Protocols' section lists 'HTTP' and 'application/json'. The 'Resources' section shows two resources: '/order' and '/order/{id}'. The '/order' resource has three methods: GET, POST, and PUT. The '/order/{id}' resource has four methods: GET, POST, PUT, and DELETE. The 'Data Types' section contains a 'Create group' button. The 'Documentation' section is currently empty. On the right side of the interface, the RAML specification for the API is displayed in a code editor. The RAML code defines the API structure, including resources like /order and /order/{id}, their methods, and data types.

To learn more about RAML follow this [tutorial](#).

- See [Designing Your API](#) for more information on API design.
- See [Design Mocking Service](#) for more information on API design

Congratulations! You have completed Lab 2.

Proceed to Lab 3



Module 1: Lab 3 Publish the Shopify API To Exchange

Module 1: The Modern API - Design: Lab 3 (1/17)



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 **Shopify Experience API**.

Module 1: The Modern API - Design: Lab 3 (2/17)



Step 1: Complete the Shopify 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 "**<your name or initials> - Shopify 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. Go to and click **Edit RAML** located at the top right corner.

The screenshot shows the Anypoint Design Center interface. At the top, there's a navigation bar with 'Design Center' and other options like 'Publish'. Below the bar, the title 'OAA-Shopify Experience API' is displayed. On the left, there's a sidebar with sections for 'API Summary', 'SECURITY SCHEMES', and 'RESOURCES'. Under 'RESOURCES', there's a list for '/order' with several status indicators. The main area shows the '/order/{id}' resource with various HTTP methods (GET, POST, PUT, PATCH, DELETE, OPTIONS, HEAD) and their corresponding icons. A red box highlights the 'Edit RAML' button in the top right corner of the resource panel. To the right of the resource details, a large panel displays the RAML code for the API. The code includes basic metadata like title, baseUri, mediaType, version, and protocols, along with a specific '/order' resource definition that includes a 'get' method with a 'responses' section.

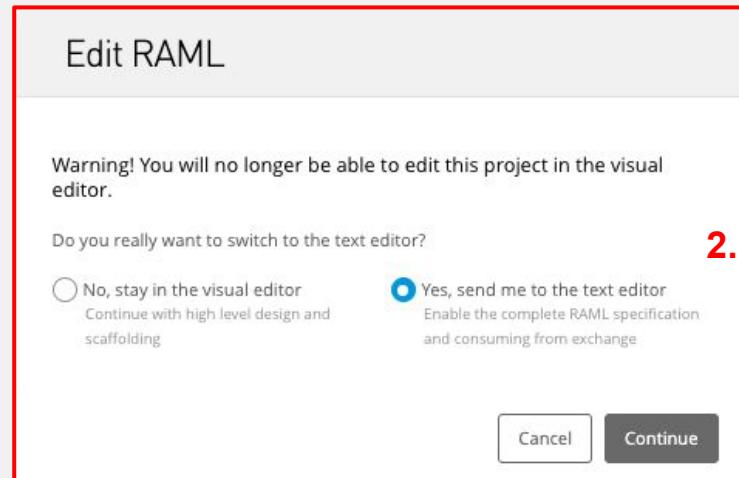
```
#%RAML 1.0
title: OAA- Shopify Experience API
baseUri: http://localhost:8081/api
mediaType:
  - application/json
version: 1.0.0
protocols:
  - HTTP
/order:
  get:
    responses:
```

Module 1: The Modern API - Design: Lab 3 (3/17)



Step 1: Complete the Shopify Experience API Specification

2. An **Edit RAML** window should appear with a warning. Select the radio button “**Yes, send me to the text editor**” and then the **Continue** Button.



3. You will notice that we are no longer in the **Visual Editor**, but are now in **Text Editor**:

```
%RAML 1.0
title: OAA- Shopify Experience API
baseuri: http://localhost:8081/api
mediaType:
| - application/json
version: 1.0.0
protocols:
| - HTTP
/order:
get:
responses:
"200": {}
/order/{id}:
get:
responses:
"200":
body:
example:
strict: true
value:
id: 413413890297800
confirmed: true
fulfilled: false
created: 2021-12-09T09:05:34-05:00
currency: USD
total: 75.55
customer:
id: 1616616
email: jane@gmail.com
address1: 567 Main St
city: Stafford
state: VA
country: USA
zip: "22554"
```

Module 1: The Modern API - Design: Lab 3 (4/17)



Step 1: Complete the Shopify Experience API Specification

4. Go to the Setup located at the top right corner.

5. In the drop down menu select **Import from Exchange**

The screenshot shows the MuleSoft Design Center interface. The top navigation bar includes 'Design Center', a user icon, and 'wo'. Below the bar, the title 'Shopify Experience API/master' is displayed. On the left, there's a sidebar with 'Files' and a 'shopify-exp-api.raml' file selected. The main area shows the RAML code for the Shopify Experience API. On the right, a context menu is open over the project settings, with the 'Import from Exchange' option highlighted by a red box and labeled '5.'.

```
1  #%RAML 1.0
2  title: Shopify Exp API
3  mediaType:
4    - application/json
5  version: 2.0.1
6  protocols:
7    - HTTPS
8  /order:
9    get:
10      responses:
11        "200":
12          body:
13            example:
14              strict: true
```

Module 1: The Modern API - Design: Lab 3 (5/17)



Step 1: Complete the Shopify Experience API Specification

6. Look for **Shopify Experience API** and Select it.

7. Press **Import asset**

Import Asset from Exchange

Name	Version	Type	Language	Date modified
Cloud Information Model	0.1.0 Stable	API Fragment	RAML	Apr 28, 2021
Shopify Experience API	1.0.0 Stable	REST API	RAML 1.0	Apr 13, 2022
Product API	1.0.0 Stable	REST API	RAML 1.0	Apr 13, 2022
Omni Channel Experience API	1.0.0 Stable	REST API	RAML 1.0	Apr 13, 2022
Social Network API for Facebook	1.0.0 Stable	REST API	RAML 1.0	Apr 13, 2022
Order API	1.0.0 Stable	REST API	RAML 1.0	Apr 13, 2022

4. Shopify Experience API

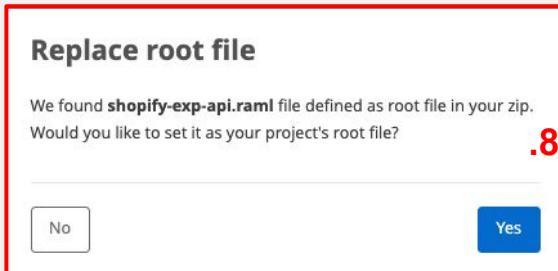
.7

Module 1: The Modern API - Design: Lab 3 (6/17)



Step 1: Complete the Shopify Experience API Specification

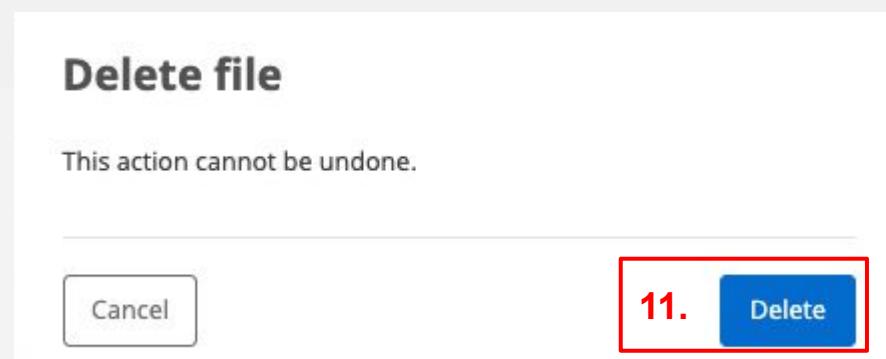
8. Choose **Yes** from the Window Replace root file.



9. On the left pane locate the <you name or initials>-shopify-experience-api and click on the three dots.

10. Select last option on the list, **Delete...**

11. A **Delete File** Window will appear, Click **Delete**.



A screenshot of a RAML editor interface. On the left, there is a file list with files: 'exchange.json', 'shopify-exp-api.raml' (which is highlighted with a red box and labeled '9.'), and 'oaa-shopify-experience-api.raml' (also highlighted with a red box and labeled '10.'). To the right of the file list is a context menu with options: 'Set as root file', 'Open in new editor tab', 'Rename', 'Copy name', 'Copy path', 'Duplicate', 'Duplicate as...', 'Download', and 'Download as...'. The 'Set as root file' option has a tooltip: 'Sets the context for parsing your specification.' A red number '12' is located at the bottom right corner of the slide.

Module 1: The Modern API - Design: Lab 3 (7/17)



Step 1: Complete the Shopify Experience API Specification

12. You should see the following complete API specification with only the `shopify-experience-api.raml` in the left pane:

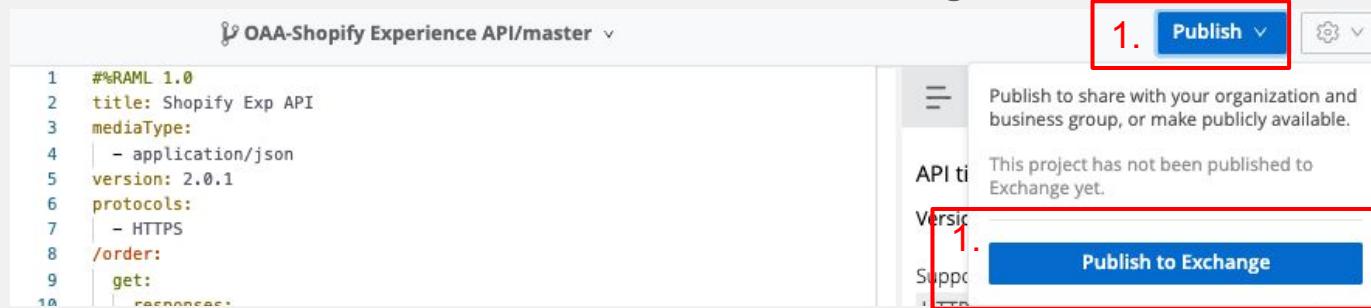
A screenshot of the MuleSoft Design Center interface. The left pane shows a file tree with a red box highlighting the file 'shopify-exp-api.raml'. The number '12.' is overlaid on this box. The right pane displays the API documentation for 'Shopify Exp API' version 2.0.1, listing endpoints for '/order', '/products', '/products/{id}', '/inventory', and '/inventory/{location}/{id}/{quantity}' with their respective HTTP methods (GET, PATCH, POST).

Module 1: The Modern API - Design: Lab 3 (8/17)



Step 2: Publish Your API to Anypoint Exchange

1. Click on the **Publish** button and then on the **Publish to Exchange** button:



2. A popup window to capture the attributes that you want to publish to Exchange will appear. Most of these fields will be pre-populated and the RAML will be verified to ensure that there are no errors. Complete the **Asset version** field with "1.0.0."

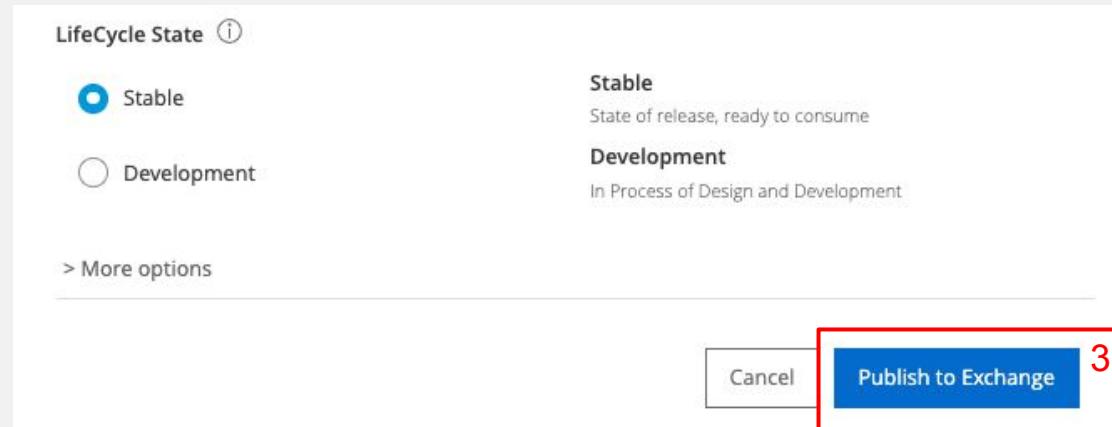
A screenshot of the 'Publish to Exchange' dialog box. It has two main sections: 'Asset versioning' and 'Additional help'. In the 'Asset versioning' section, there is a field labeled 'Asset version (required)' containing '1.0.0', which is highlighted with a red box. Below it is another field labeled 'API version (required)' containing '2.0.1'. To the right of these fields is a note about SemVer and examples like '1.0.0 or 4.3.1'. In the 'Additional help' section, there are two bullet points: 'Changing a project's main/root file' and 'What is an API version?'. At the bottom of the dialog, there are sections for 'LifeCycle State' (with 'Stable' selected) and 'Stable' (described as 'State of release, ready to consume') and 'Development' (described as 'In Process of Design and Development'). There is also a link to 'More options'.

Module 1: The Modern API - Design: Lab 3 (9/17)



Step 2: Publish Your API to Anypoint Exchange

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



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

- RAML Spec: This is the RAML Specification. Everybody on 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.

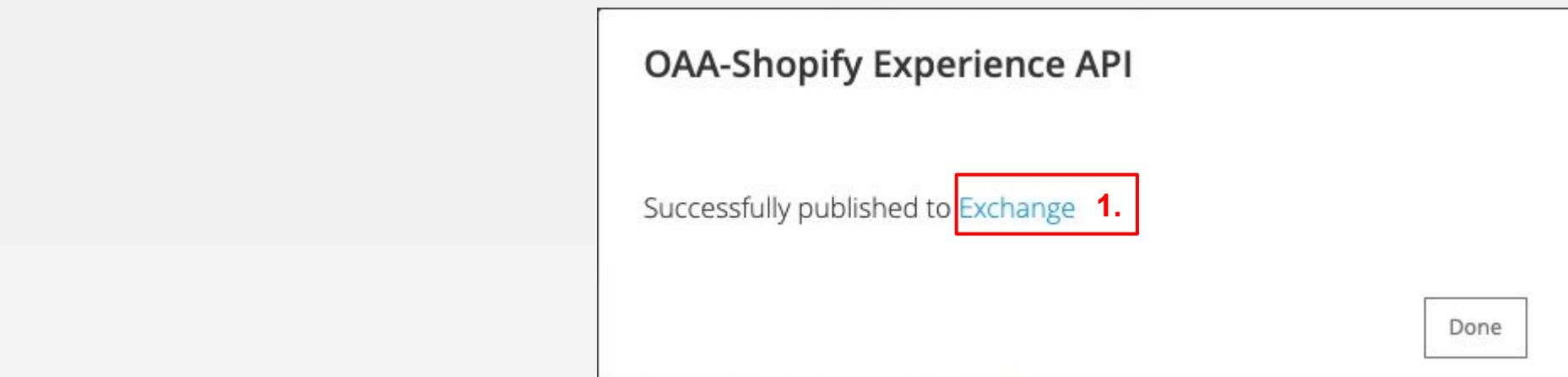
Module 1: The Modern API - Design: Lab 3 (10/17)



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.

- 1a. After you click the **Publish to Exchange** the **Shopify Experience API** Specification will start loading into **Exchange**. Once loaded there will be a pop window letting you know that your **Shopify Experience API** is in **Exchange**. To go to the API specification you can simply click on the “**Exchange**” link in the window.

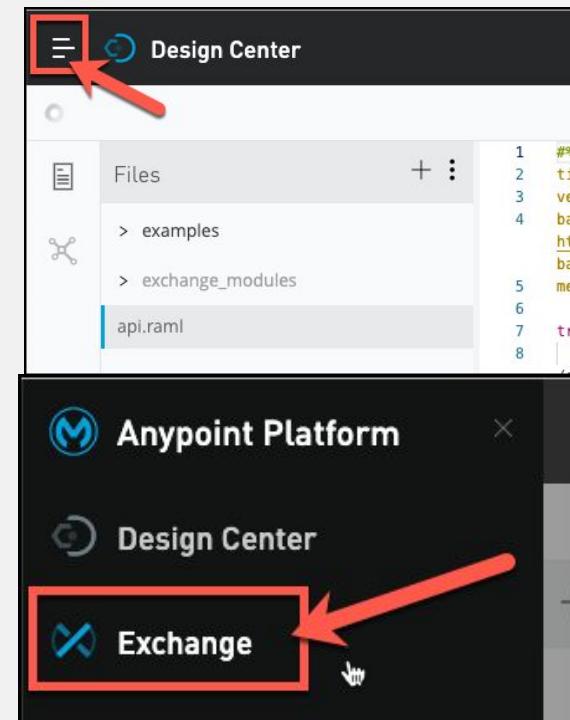


Module 1: The Modern API - Design: Lab 3 (11/17)



Step 3: Augment the Documentation for Your API

1b. Alternatively, if you had already closed out of the confirmation window you can Navigate to Exchange from the "hamburger menu" at the top left corner of **Design Center**.



You can then quickly search for the “<your name or initial>-Shopify Experience API” in **Exchange** as we did **Lab 1** of this **Module**

Module 1: The Modern API - Design: Lab 3 (12/17)



Step 3: Augment the Documentation for Your API

- Once you are in the <your name or initial>-Shopify Experience API you should see something that looks like this:

A screenshot of the Shopify Experience API documentation interface. At the top, there's a navigation bar with 'Exchange' and 'DTC Workshop DS' tabs. Below the navigation, the API name 'OAA-Shopify Experience API' is displayed along with its type ('REST API'), owner ('DTC Workshop DS'), and last update ('Updated 17 minutes ago'). On the right side, there are buttons for 'Share', 'Download', 'View code', and 'Add version'. Version information shows '2.0.1 Private' and '1.0.x'. A 'Manage versions' dropdown is also present. In the center, there's a sidebar with 'Assets list' and 'PAGES' sections. Under 'PAGES', 'Home' is selected, and a sub-section 'SPECIFICATION' is shown with 'Summary', 'Endpoints', '/order', and '/products' items. To the right of the sidebar, there's a large 'Edit documentation' button with a red box around it and the number '3.' above it. A decorative illustration of hot air balloons and mountains is at the bottom. A status bar at the bottom right indicates 'Latest 1.0.0' and 'Stable'.

- Let's start adding some content to the API by click on the **Edit documentation** icon

Module 1: The Modern API - Design: Lab 3 (13/17)



Step 3: Augment the Documentation for Your API

4a. 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:

Welcome to the **NTO developer portal**. This API represents the Direct to Consumer Shopify Shopping Cart Application - that includes:

-order
-product
-inventory tracking

The **Shopify API is an Experience API** that is a part of NTO's API led Architecture . This same architecture allows NTO to leverage a **headless commerce** platform that is **highly composable** and **future proof!**

If you have problems copying the code you can find example [here](#)

Module 1: The Modern API - Design: Lab 3 (14/17)



Step 3: Augment the Documentation for Your API

4b. **Alternatively**, if you are more comfortable with WYSIWYG editing you can click on the Visual button and copy an paste the following into the editor body:

Welcome to the **NTO developer portal**. This API represents the Direct to Consumer Shopify Shopping Cart Application - that includes:

- Order
- Product
- inventory tracking

The **Shopify API is an Experience API** that is a part of NTO's API led Architecture .
This same architecture allows NTO to leverage a **headless commerce** platform
that is **highly composable** and **future proof!**

Module 1: The Modern API - Design: Lab 3 (15/17)



Step 3: Augment the Documentation for Your API

5. Click on the "Save" button at bottom of screen. 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.

The screenshot shows the MuleSoft Exchange interface for creating a new draft of the OAA-Shopify Experience API. The top navigation bar includes 'Exchange', 'DTC Workshop DS', and 'WO'. The main header displays the API name 'OAA-Shopify Experience API', its type 'REST API', version '2.0.1 Private', and status '1.0.x'. A red box highlights the 'Save' button in the top right corner of the main content area, which contains a WYSIWYG editor and a summary of the API's purpose and features.

Creating a new draft

OAA-Shopify Experience API

REST API | DTC Workshop DS | Updated 47 minutes ago

Workshop Owner published 47 minutes ago

Manage versions | 2.0.1 Private | 1.0.x

Add tags

Assets list

PAGES

Home

+ Add new page

SPECIFICATION

① Summary

OTHER DETAILS

+ Add terms and conditions

All contents © Mule

Welcome to the **NTO developer portal**. This API represents the Direct to Consumer Shopify Shopping Cart Application - that includes:
-order
-product
-inventory tracking

The **Shopify API is an Experience API** that is a part of NTO's API led Architecture . This same architecture allows NTO to leverage a **headless commerce** platform that is **highly composable** and **future proof!**

Cancel Save Publish

Module 1: The Modern API - Design: Lab 3 (16/17)



Step 3: Augment the Documentation for Your API

6. You can press Publish, Discard , Edit or View published.

- Discard - Throw away changes
- View Published - View published documentation
- Edit - Lets you edit the content
- Publish - Publish documentation changes to exchange

The screenshot shows the MuleSoft Exchange interface for the "OAA-Shopify Experience API". The top navigation bar includes "Exchange", "DTC Workshop DS", and user status. Below the header, the API name "OAA-Shopify Experience API" is displayed along with its type ("REST API"), dataset ("DTC Workshop DS"), and last update ("Updated 53 minutes ago"). On the right side, there are buttons for "Share", "Download", "View code", and "Add version". Version information shows "2.0.1 Private" as the latest. A red box highlights the bottom row of buttons: "Discard", "View published", "Edit", and "Publish", with the number "6." next to "Publish". The footer contains links to "Assets list", "PAGES", and "All cd...". A welcome message at the bottom states: "Welcome to the NTO developer portal. This API represents the Direct to Consumer Shopify Shopping Cart Application - that includes: -order -product".

Module 1: The Modern API - Design: Lab 3 (17/17)



Step 3: Augment the Documentation for Your API

7. Press the **Publish** button. Your documentation changes are now visible on Exchange!

A screenshot of a web browser showing the 'OAA-Shopify Experience API' page on the NTO developer portal. The page title is 'OAA-Shopify Experience API'. Below it are status indicators: 'REST API', 'DTC Workshop DS', and 'Updated 53 minutes ago'. To the right are buttons for 'Share', 'Download', 'View code', and 'Add version'. A 'Manage versions' dropdown shows '2.0.1 Private' and '1.0.x'. Below that, a 'Latest' dropdown shows '1.0.0' and a 'Stable' dropdown is set to 'Stable'. At the bottom, there are buttons for 'Discard', 'View published', 'Edit', and 'Publish'. The 'Publish' button is highlighted with a red box and the number '7.' is written next to it. Other UI elements include '+ Add tags', 'Viewing draft', and a welcome message: 'Welcome to the NTO developer portal. This API represents the Direct to Consumer Shopify Shopping Cart Application - that includes:'.

Exchange documentation can be comprised of 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.

Congratulations! You have completed all of the labs in Module 1!

Proceed to Module 2

Module 2: Ubiquitous Connectivity

Module 2: Ubiquitous Connectivity: Overview (1/2)



Overview

You have completed the design of the Shopify API and now you are ready to implement it.

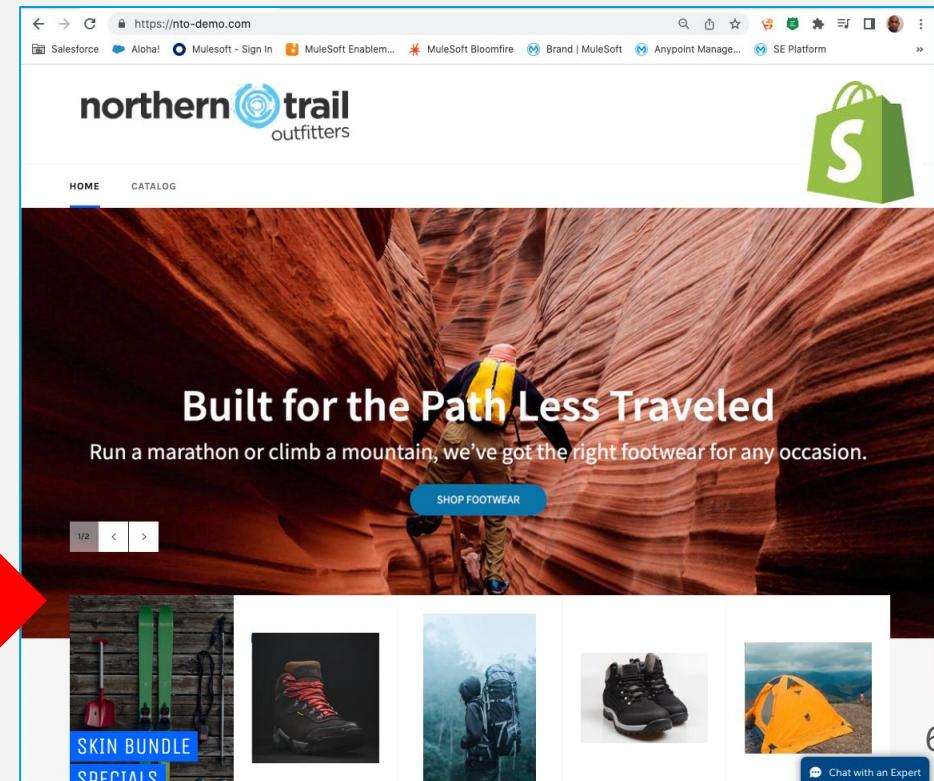
In this module, we will create the implementation of our API that will process the REST requests.

The Module is separated into three different labs.

- [Lab 1: Create Shopify Project from API Specification](#)
- [Lab 2: Implement the Shopify API in Studio](#)
- [Lab 3: Deploy the Shopify API Implementation](#)

To test the connectivity to the NTO Shopify shopping cart we will be testing a single use case of retrieving all Shopify orders. Since sourcing consumer orders from Shopify is critical for NTO's DTC goals we will make sure that works.

We will leverage the <https://nto-demo.com/> Shopify site to do just this.

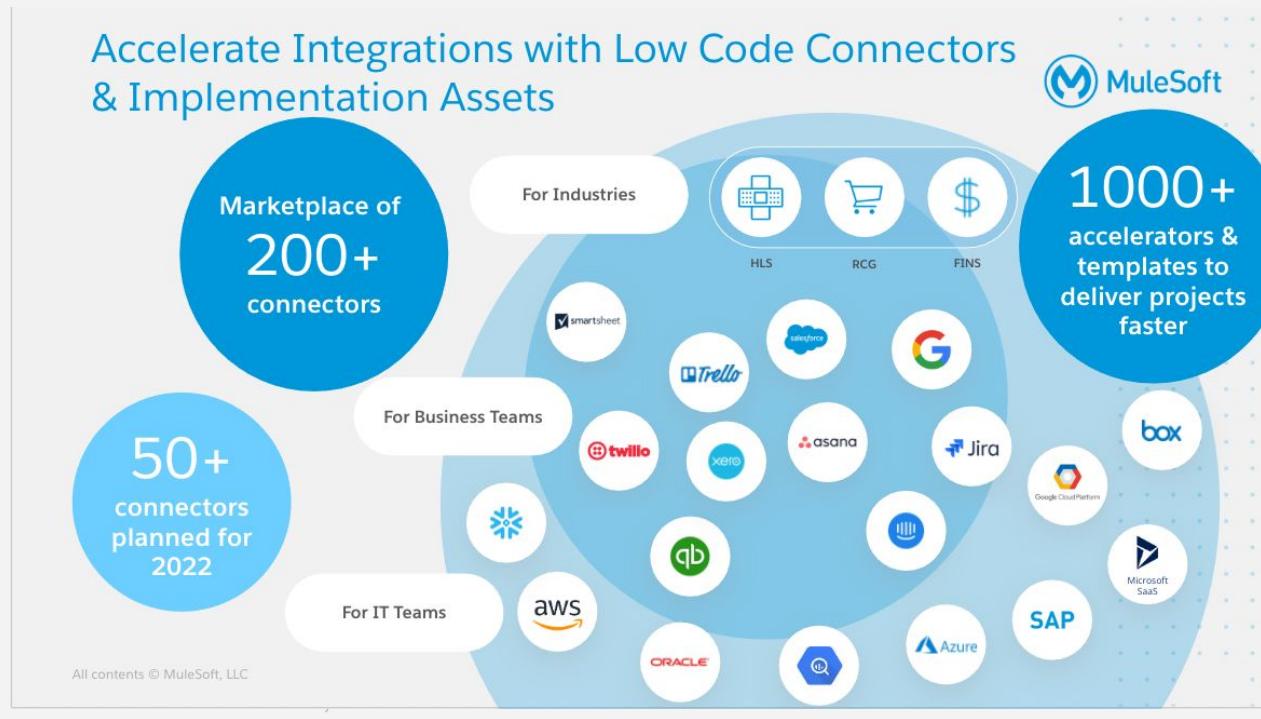


Module 2: Ubiquitous Connectivity: Overview (2/2)



Overview

This module will feature low code connectivity via what we call “**connectors**”. Today Anypoint Studio has over 200 system specific connectors (e.g. Salesforce, Shopify, SAP, etc.) that makes connecting to some of our main source systems easy. Since NTO is using Shopify, we will be getting very familiar with the **Shopify Connector**.



Proceed to Lab 1



Module 2: Lab 1 Create Shopify Project from API Specification

Module 2: Ubiquitous Connectivity: Lab 1 (1/16)



Overview

In this first lab, you will use Anypoint Studio to create a Mule application where there will be one flow for each method of each resource (i.e. GET). Additionally you will use APIKit (as part of your Mule application) to process each REST requests, transform them to messages to be handled and processed by each flow.

The implementation will consist of a few steps

1. Create a new Mule Project in Anypoint Studio from the Shopify API RAML
2. Review the Project with the Shopify API Scaffolding in place

Then we will test this new API using the API Console.

Module 2: Ubiquitous Connectivity: Lab 1 (2/16)



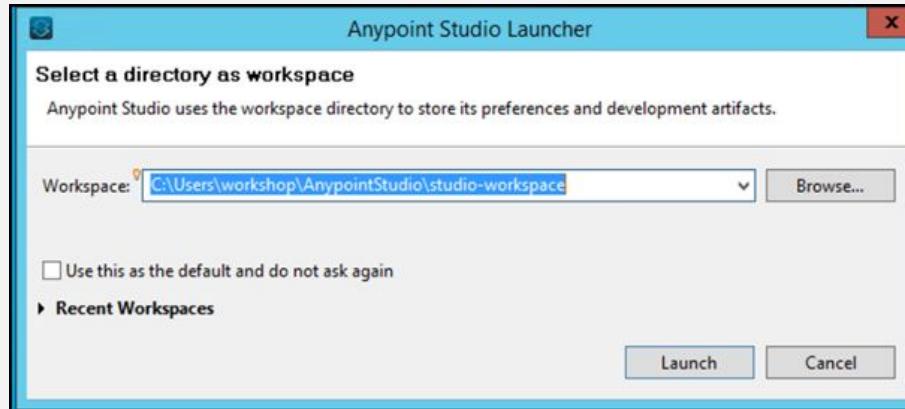
Step 1: Create a new Mule Project from the RAML

In this step we will create a new Mule application in Anypoint Studio from the Omni Channel API RAML Definition. This will be the implementation of our REST API.

1. If you are using Remote Desktop look for the Start **Anypoint Studio** from the desktop icon.



2. When you open the Studio for the first time, you are going to be asked for the workspace where the projects are going to be saved. Select the one that comes by default.



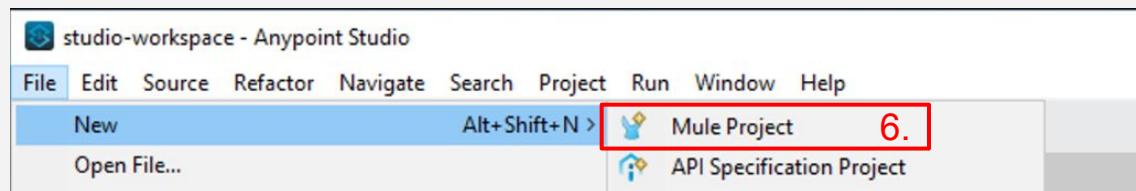
Module 2: Ubiquitous Connectivity: Lab 1 (3/16)



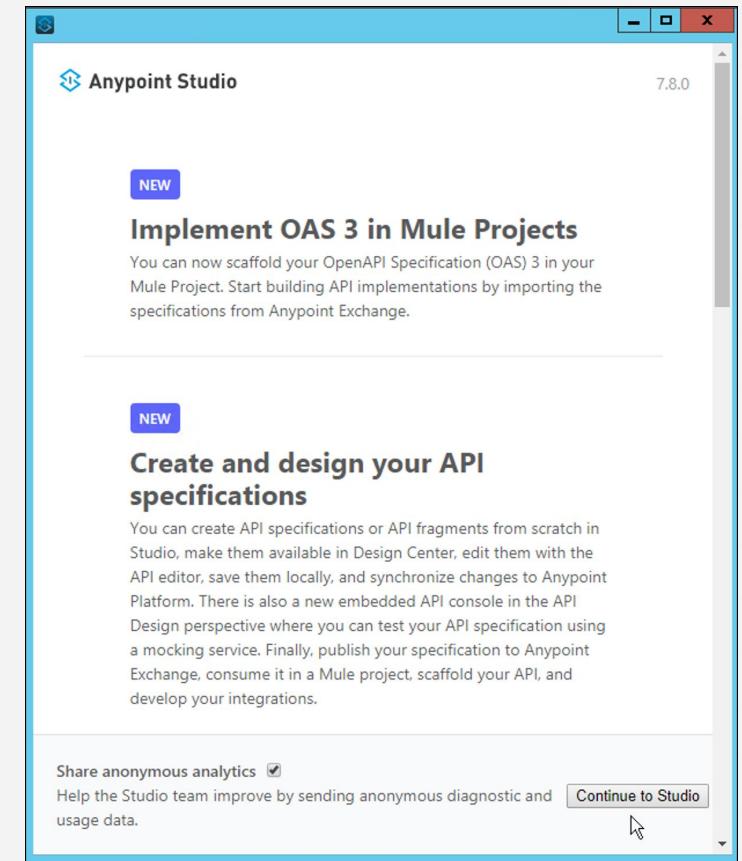
Step 1: Create a new Mule Project from the RAML

3. You are presented with the Anypoint Studio Welcome Screen.
4. Scroll to the bottom if you want to know more on Studio. After that, press **Continue to Studio**.
5. In the Workspace Launcher, use the default workspace directory location.
6. 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.



7. Give the project the name **shopify-exp-api-v1**
8. Select the **Mule Server 4.4.0 EE** (or latest version if present).



Module 2: Ubiquitous Connectivity: Lab 1 (4/16)



Step 1: Create a new Mule Project from the RAML

9. Under **Import a Published API** we are going to import the RAML Spec from Exchange. Click on **green +** this will present two options from Exchange or from Maven
10. Select from **Exchange**.

The screenshot shows the 'New Mule Project' dialog box. The 'Project Name' field contains 'shopify-exp-api-v1'. The 'Runtime' dropdown is set to 'Mule Server 4.4.0 EE'. In the 'API Implementation' section, the 'Import a published API' tab is selected. Below it, there is a note: 'Start building API implementations by importing the specification here.' followed by a 'Learn more' link. A red box highlights the green '+' icon next to the text 'from Exchange' and the number '10.'.

New Mule Project

Project Settings

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

Project Name: shopify-exp-api-v1

Runtime

Mule Server 4.4.0 EE

Install Runtimes

API Implementation

Add an API implementation to your project to automatically set up an APIkit router and create placeholder flows for each resource method

Import a published API Import RAML from local file Download RAML from Design Center

① Start building API implementations by importing the specification here. [Learn more](#)

9. **+** 10. from Exchange from Maven

Please select or add a dependency to see more information.

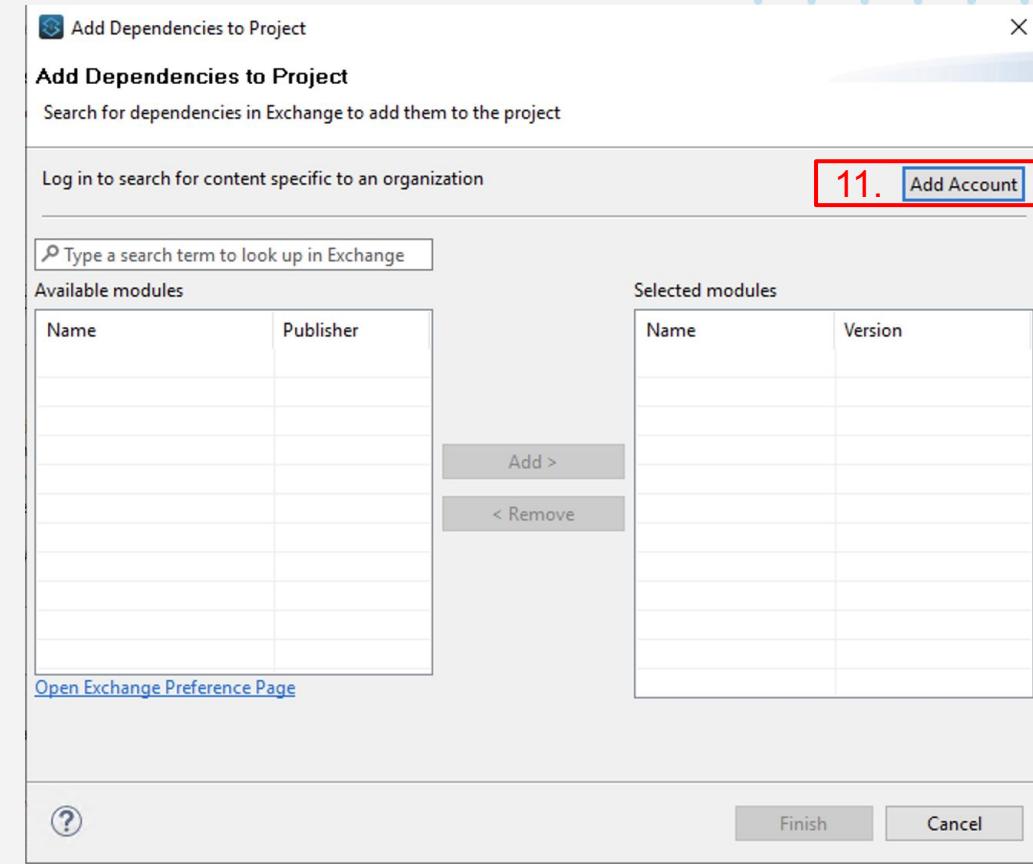
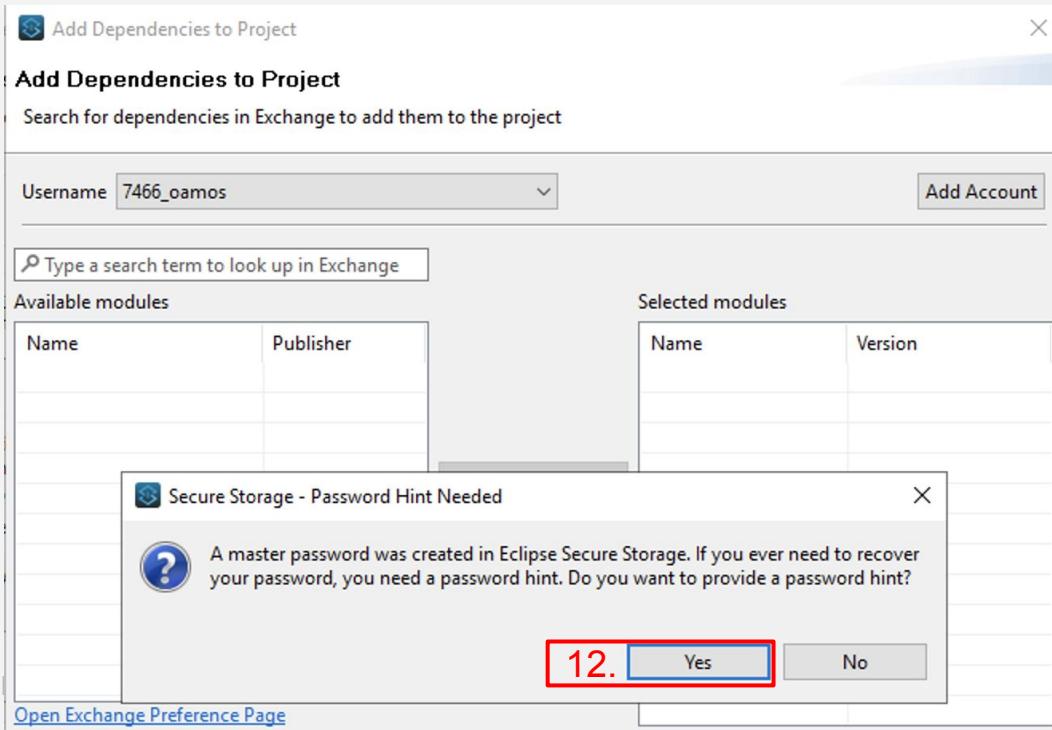
Module 2: Ubiquitous Connectivity: Lab 1 (5/16)



Step 1: Create a new Mule Project from the RAML

11. Once you clicked there, you will see a new window. You need to login to the Anypoint Platform. So click on the **Add Account** button.

12. A **Secure Storage** prompt will appear click **Yes** to proceed.



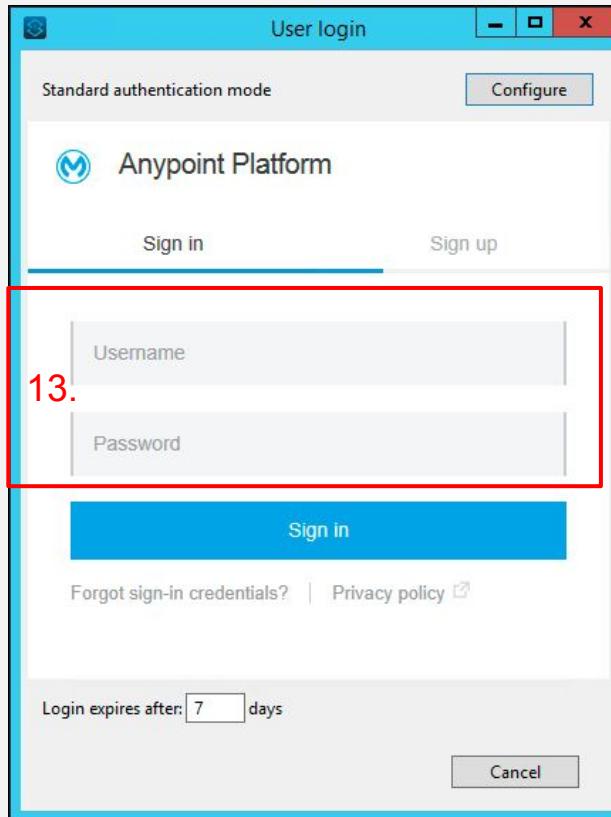
Module 2: Ubiquitous Connectivity: Lab 1 (6/16)



Step 1: Create a new Mule Project from the RAML

13. A new Login window will appear. Insert the credentials that were sent to your email earlier.

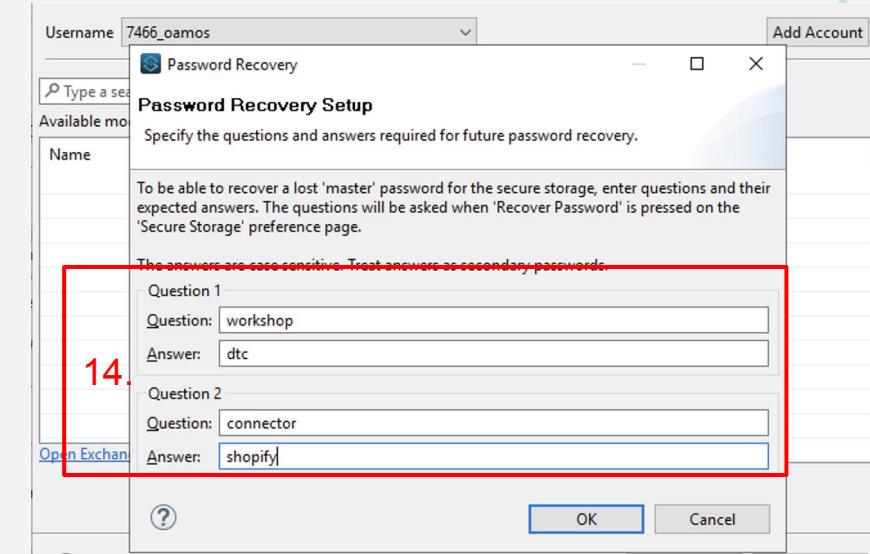
-



13.

14. You will be taken to a Password Recovery Setup window. From there add the following details:

- Question 1: Question: **workshop** Answer: **dtc**
- Question 2: Question: **connector** Answer: **shopify**



14.

Module 2: Ubiquitous Connectivity: Lab 1 (7/16)



Step 1: Create a new Mule Project from the RAML

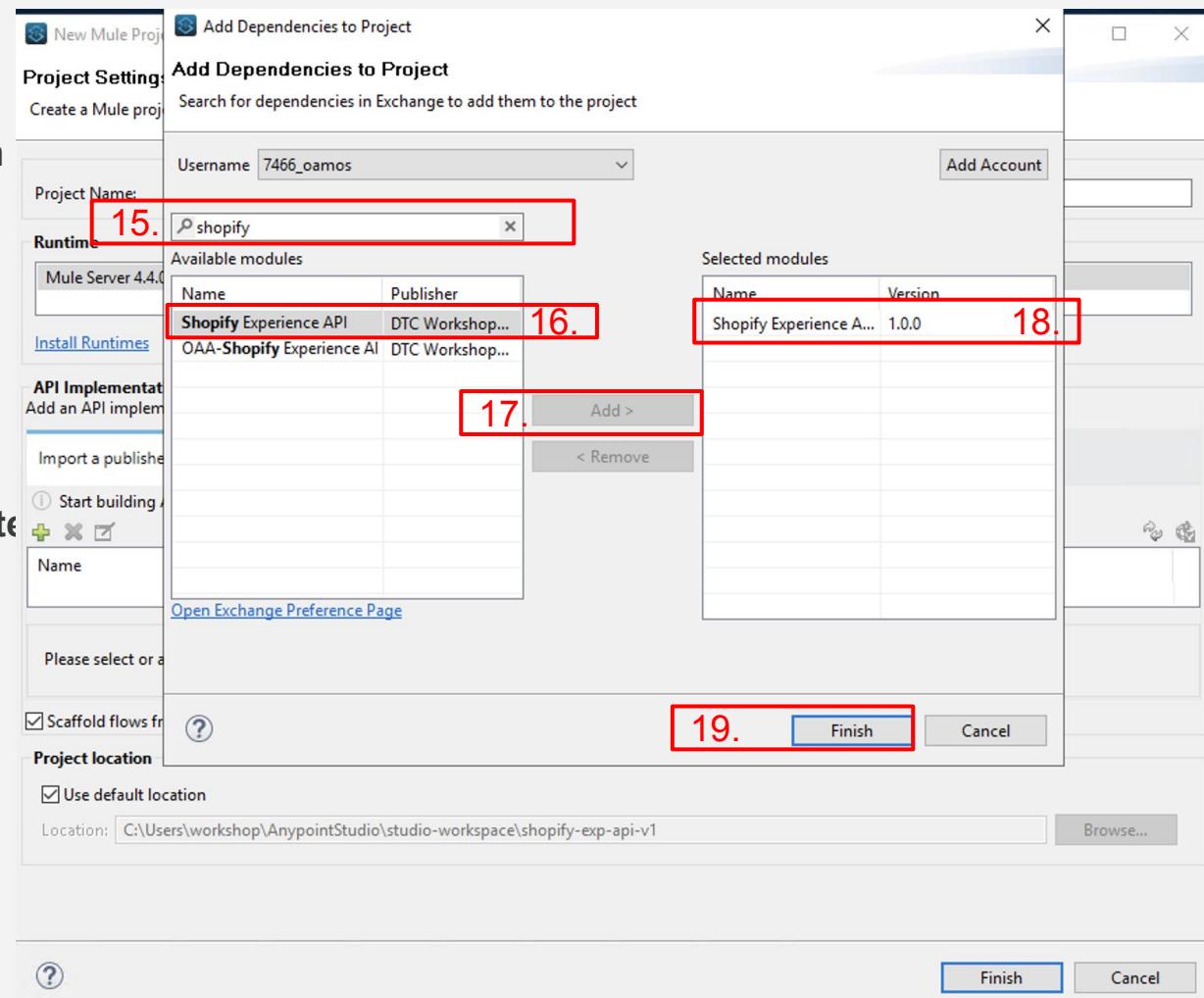
15. You will be taken to the Add Dependencies to Project window. From there type “**shopify**” in the search area.

16. You might see several options, select the **Shopify Experience API** that doesn’t have any names or initials in front of it.

17. Click the **Add** button.

18. The **Shopify Experience API** should be on the right pane of **Selected modules**. Make sure that is the case before proceeding.

19. Click **Finish**.

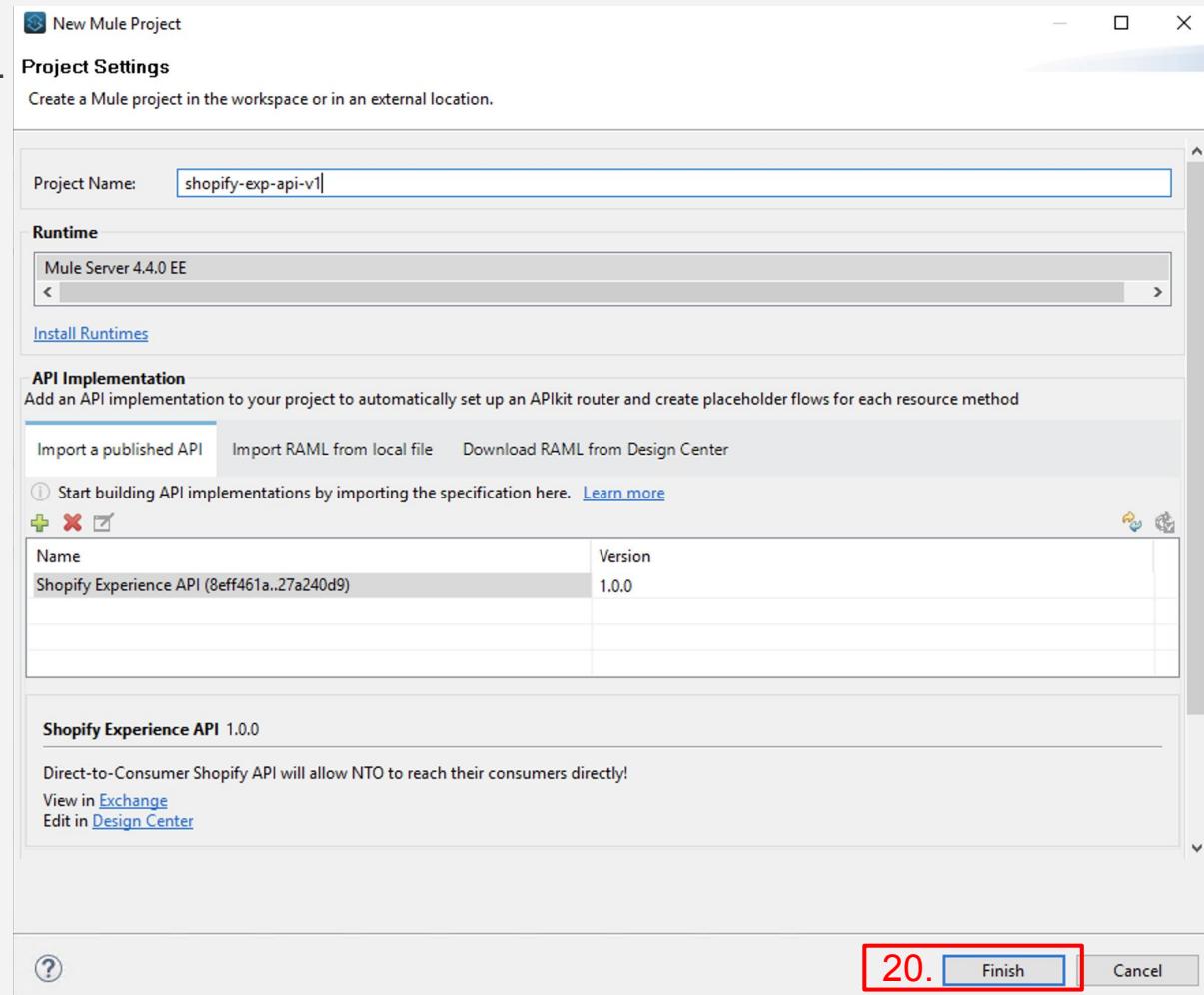


Module 2: Ubiquitous Connectivity: Lab 1 (8/16)



Step 1: Create a new Mule Project from the RAML

20. You should be back at the **Project Settings** Window.
Click **Finish**.



Module 2: Ubiquitous Connectivity: Lab 1 (9/16)



Step 2: Review Scaffolded Flows in the Project

Done! Now let's explore what was automatically generated for our project.

The screenshot shows the Anypoint Studio interface with the following components:

- Package Explorer:** Shows the project structure: `shopify-exp-api-v1` containing `src/main/mule` (with `shopify-exp-api.xml`) and other source/resource directories.
- Middle Panel:** Displays the `shopify-exp-api-main` flow. It starts with a **Listener** (represented by a globe icon) connected to an **APIKit Router** (represented by a square icon with multiple output ports). Below the router are two **Error handling** sections:
 - On Error Propagate type: APIKIT:BAD_REQUEST**: Contains a **Transform Message** component (represented by a blue circle with a white icon).
 - On Error Propagate type: APIKIT:NOT_FOUND**: Contains a **Transform Message** component.
 - On Error Propagate type: APIKIT:METHOD_NOT_ALLOWED**: Contains a **Transform Message** component.
- Mule Palette:** Located on the right, it lists various Mule components categorized under **Favorites**: **Logger (Core)**, **Listener (HTTP)**, **Request (HTTP)**, **Transform Message (Core)**, **Set Payload (Core)**, **Flow Reference (Core)**, and **Choice (Core)**.
- Outline:** Shows the `Mule Configuration` section with flows like `shopify-exp-api-main`, `shopify-exp-api-console`, and `patch:order:application/json:shopify-exp-api-config`.

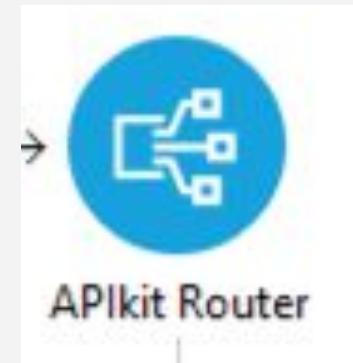
Module 2: Ubiquitous Connectivity: Lab 1 (10/16)



Step 2: Review Scaffolded Flows in the Project

If you click your mouse cursor to the middle pane under the tab called **shopify-exp-api**, you will see our first “flow” called **shopify-exp-api-main**. When you scroll your mouse down you will start seeing other flows (e.g. **shopify-exp-api-console**, and so on).

All the flows are defined by your API Design in the RAML file and have created a scaffolding for the Shopify API project. Typically, there will be flows that look like this **get:\resource**, **post:\resource**, **put:\resource**, etc. Note that the name of the flow is very important for the **APIkit router** to be able to route the request to the appropriate flow - you don’t want to change these.



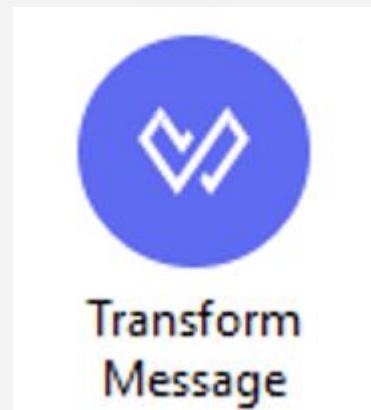
Module 2: Ubiquitous Connectivity: Lab 1 (11/16)



Step 2: Review Scaffolded Flows in the Project

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

The static response returned by the auto-generated **Dataweave Transform Message** allows you to test the API as a stub immediately after generation. The placeholder Transform Message will have sample payload outputs like the one you supplied for your version of the Shopify API in Module 1 Lab 2. Obviously these flows can be enhanced to evolve them into full API implementation as we will see in the next lab.



```
4@ {  
5@   id: 413413890297800,  
6@   confirmed: true,  
7@   fulfilled: false,  
8@   created: "2021-12-09T09:05:34-05:00",  
9@   currency: "USD",  
0@   total: 75.55,  
1@   customer: {  
2@     id: 1616616,  
3@     email: "jane@gmail.com",  
4@     address1: "567 Main St",  
5@     city: "Stafford",  
6@     state: "VA",  
7@     country: "USA",  
8@     zip: "22554",  
9@     phone: "555-710-9325",  
0@     line_items: [  
1@       {  
2@         id: 10417352802515,  
3@         sku: "841483107332",  
4@         name: "Waist Cincher - Soft Nude / L",  
5@         quantity: 1,  
6@         gift_card: false,  
7@         price: 75.55,  
8@         tax_code: "PC040100"  
9@       }  
0@     ]  
1@   }  
2@ }
```

Module 2: Ubiquitous Connectivity: Lab 1 (12/16)



Step 2: Review Scaffolded Flows in the Project

Let's review a few of the flows in our Mule project.

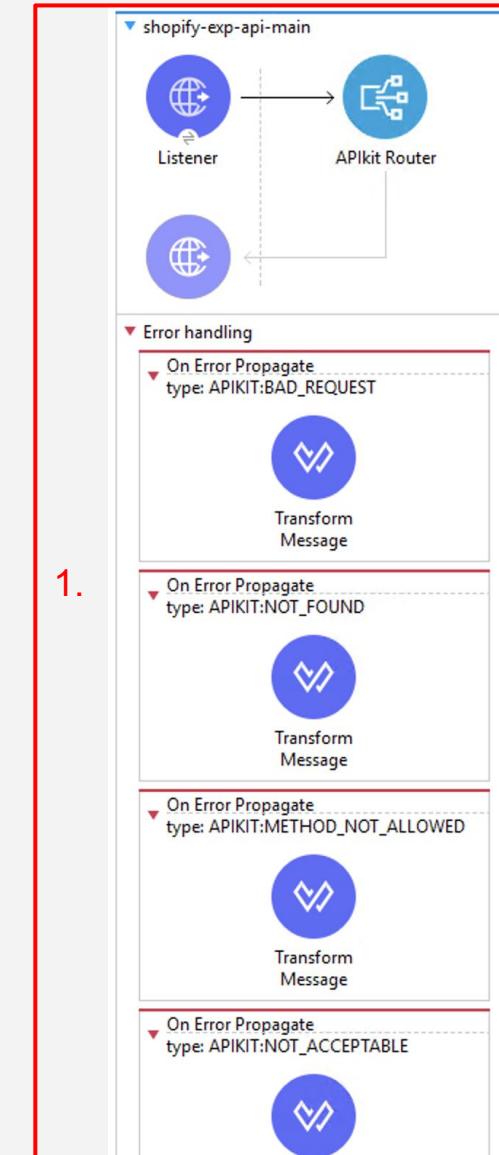
1. shopify-exp-api-main

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.

By selecting the http listener one can take a look at the HTTP configuration, there you will see the Protocol, Host and Port attributes, all that information means that the listener is in fact listening for requests on <http://localhost:8081/api>.

Inside the api-main flow also an [Error-Handling](#) block is included. In this block multiple **On Error Propagate** definitions are autogenerated for common API error responses such as **METHOD_NOT_ALLOWED**, **BAD_REQUEST** etc.



Module 2: Ubiquitous Connectivity: Lab 1 (13/16)



Step 2: Review Scaffolded Flows in the Project

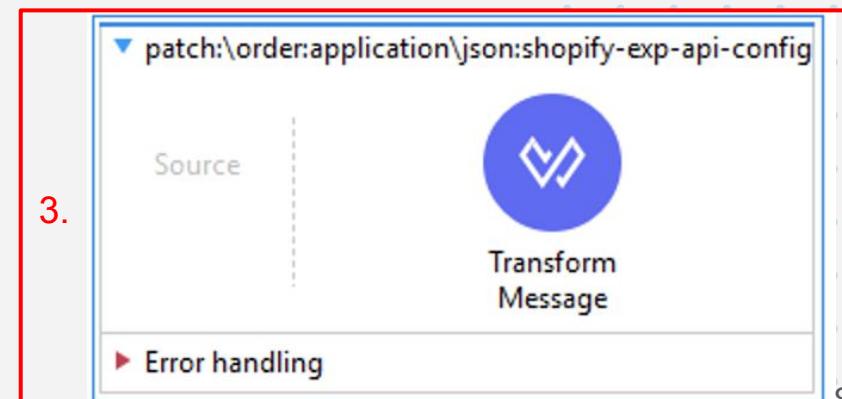
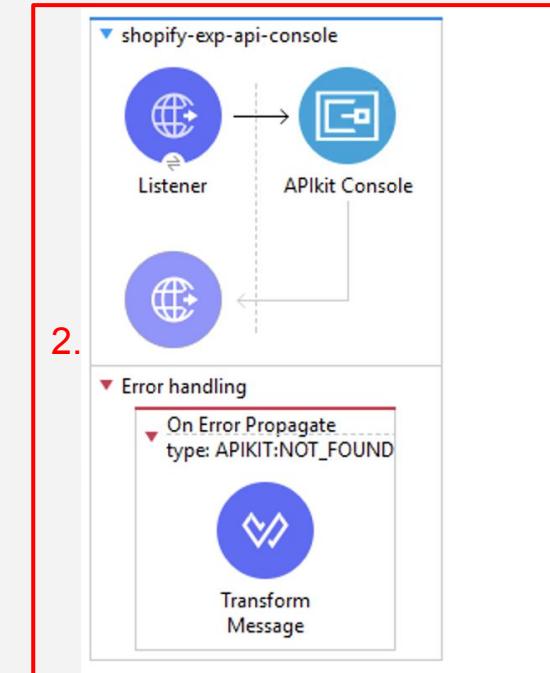
Let's review a few of the flows in our Mule project.

2. shopify-exp-api-console

The console flow is not mandatory. It provides an auto generated interactive HTML console for interacting with your api for testing etc. It is auto generated from your RAMI/api spec. Whereas to test an API you will be able to test from the similar UI as when using Mocking Services previously in Module 1.

3. patch:\order\application\json:shopify-exp-api-config

Is our first flow that represents a resource endpoint. This flow is the framework for the PATCH operator in our order resource of our Shopify API. Patch command will add new consumer orders in NTO's Shopify and can update existing orders from the same call.



Module 2: Ubiquitous Connectivity: Lab 1 (14/16)



Step 2: Review Scaffolded Flows in the Project

Let's review a few of the flows in our Mule project.

4. get\inventory:shopify-exp-api-config

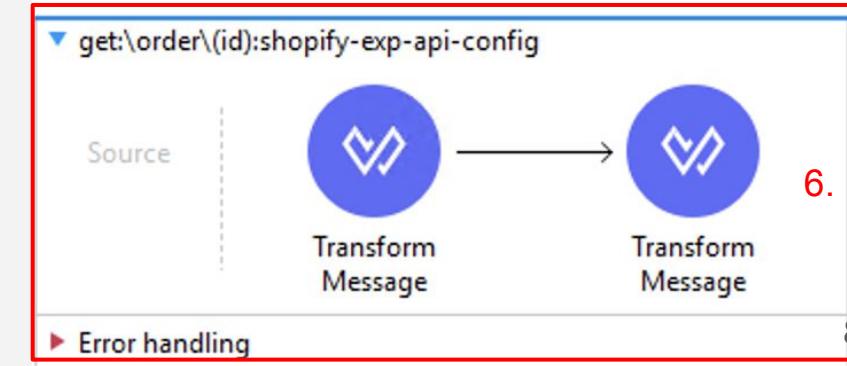
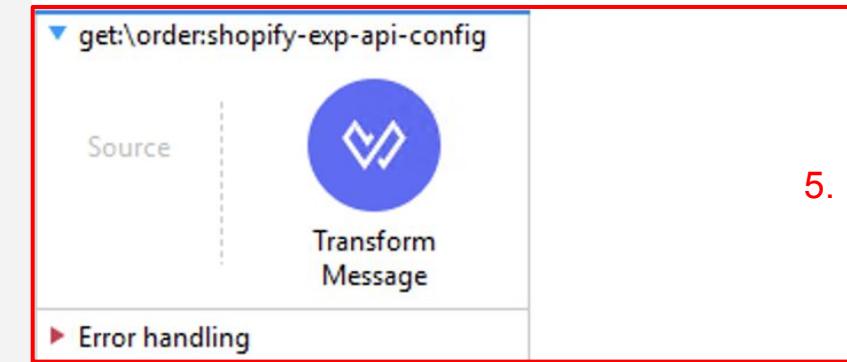
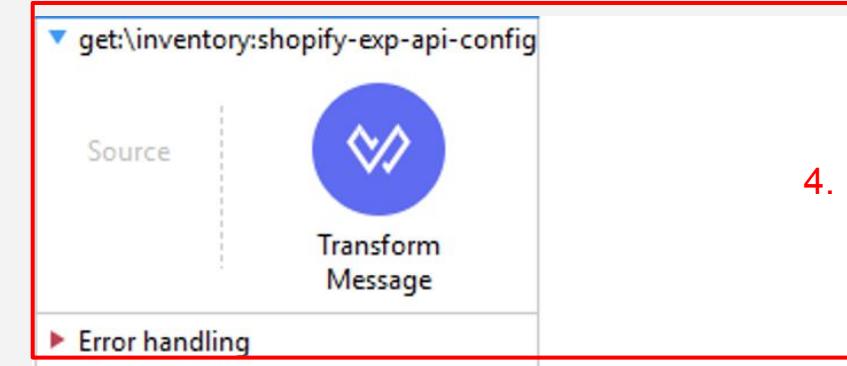
This flow will be the end point to allow the full list retrieval of items and inventory within NTO's Shopify application.

5. get\order:shopify-exp-api-config

This flow will get the all consumer orders that reside in NTO's Shopify site. **This will be the flow we will be working on in our next lab, Lab 2.**

6. get\order\id:shopify-exp-api-config

This flow is getting an instance or a single order by ID.



Module 2: Ubiquitous Connectivity: Lab 1 (15/16)



Step 2: Review Scaffolded Flows in the Project

Let's review a few of the flows in our Mule project.

7. get\products:shopify-exp-api-config

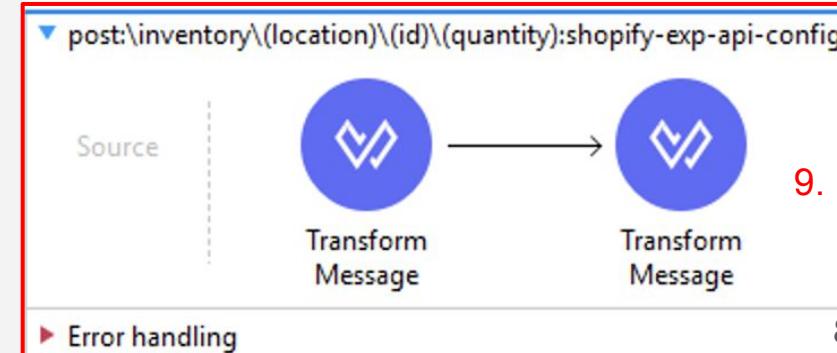
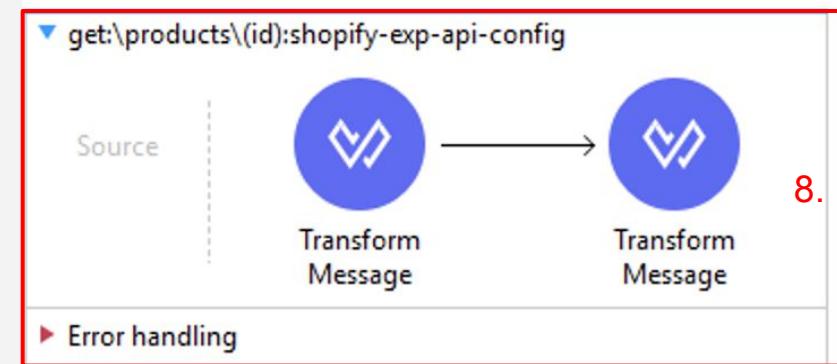
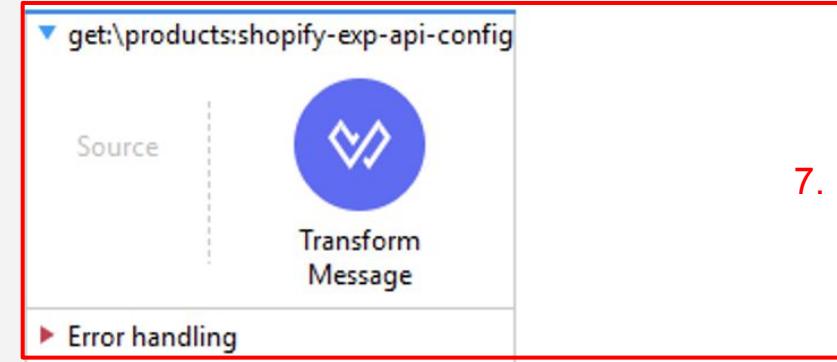
This flow will retrieve full list retrieval of products and product data within NTO's Shopify Site.

8. get\products\:(id):shopify-exp-api-config

This flow will get product information from a single product.

9. post:\inventory\:(location)\:(id)\:(quantity):shopify-exp-api-config

This flow will allow to update inventory levels within NTO's Shopify Site while passing three needed attributes (Product ID, location ID, Updated Quantity). This way of executing the end point offers a lightweight POST command that doesn't require a BODY in the POST request.



Module 2: Ubiquitous Connectivity: Lab 1 (16/16)



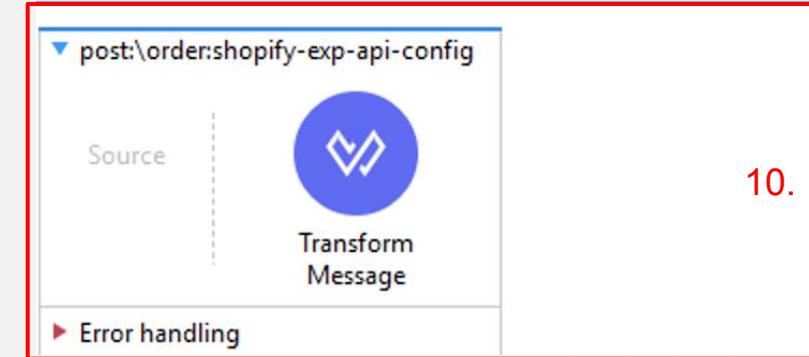
Step 2: Review Scaffolded Flows in the Project

Let's review a few of the flows in our Mule project.

10. post\order:shopify-exp-api-config

Lastly, we have a POST orders to allow for consumer orders that were not created in Shopify to be in the Shopify order details as away of allowing for consistent consumer experience across channels.

Now that we have familiarized ourselves with the scaffolding of our Shopify API we can now proceed to building the connectivity to the NTO Shopify Site.





Module 2: Lab 2 Implement the Shopify API In Studio

Module 2: Ubiquitous Connectivity: Lab 2 (1/23)



Overview

Since you understand how to start a project by importing the Shopify RAML Specification. You have also seen how the project scaffolding works with all the required resource endpoints. Now we need to add connectivity to Shopify and we will be using a Shopify Connector that will allow us to quickly create connectivity.

The implementation will consist of a few steps

1. Find the Shopify Connector via Anypoint Studio (through Exchange)
2. Create Configuration Properties YAML to externalize needed Shopify credentials
3. Add our Shopify Connector operator in the flow we will be working on
4. Update the Transform Message in the flow we are working on
5. Test the implementation by running the Shopify project

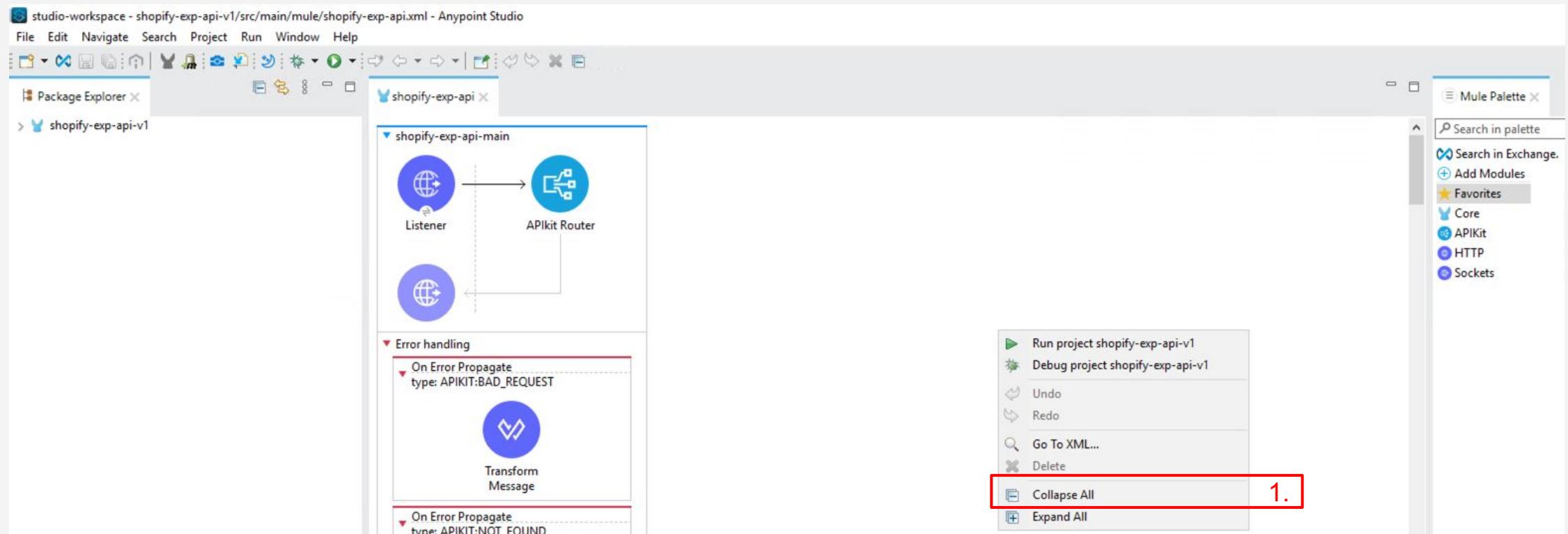
Module 2: Ubiquitous Connectivity: Lab 2 (2/23)



Step 1: Setup up working on your flow

We will start to add implementation connectivity of Shopify and will test out on flow endpoint resource

get\order:shopify-exp-api-config. To make working on this endpoint a bit easier let's collapse all of the flows in the Project. We do so by right clicking in the body (any blank area will do) and right-mouse click to see the **Collapse All** option and select it.

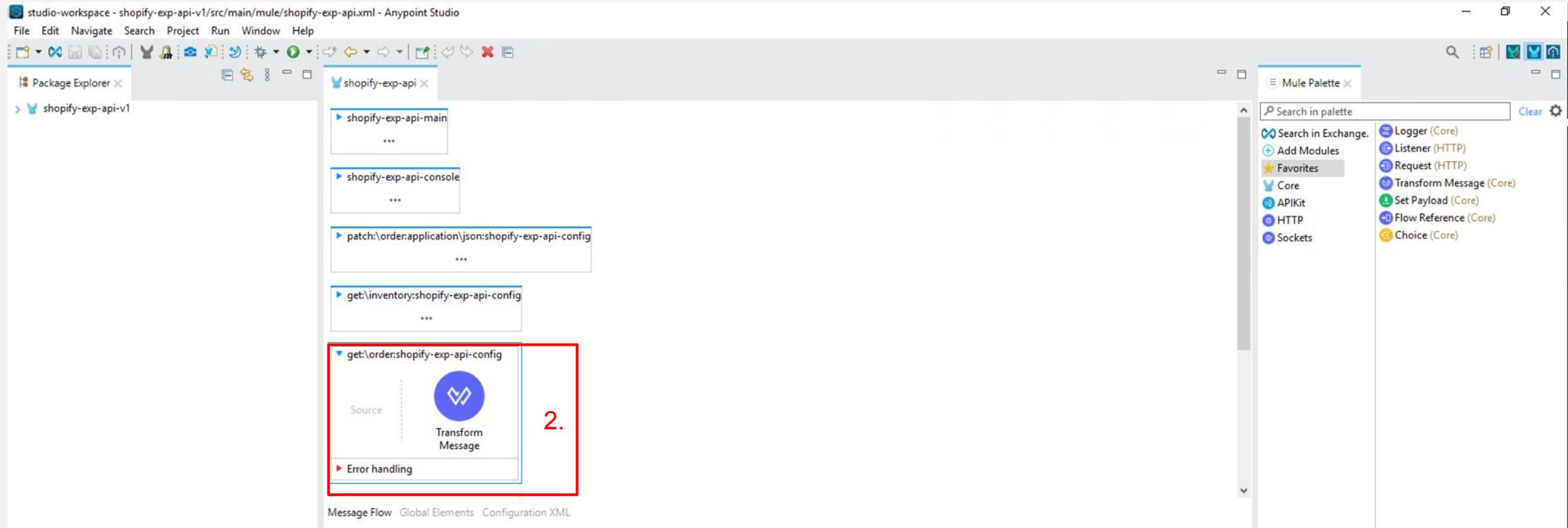


Module 2: Ubiquitous Connectivity: Lab 2 (3/23)



Step 1: Setup up working on your flow

2. Now find the get\order:shopify-exp-api-config flow and click on the blue arrow in the top left to expand the flow like this:

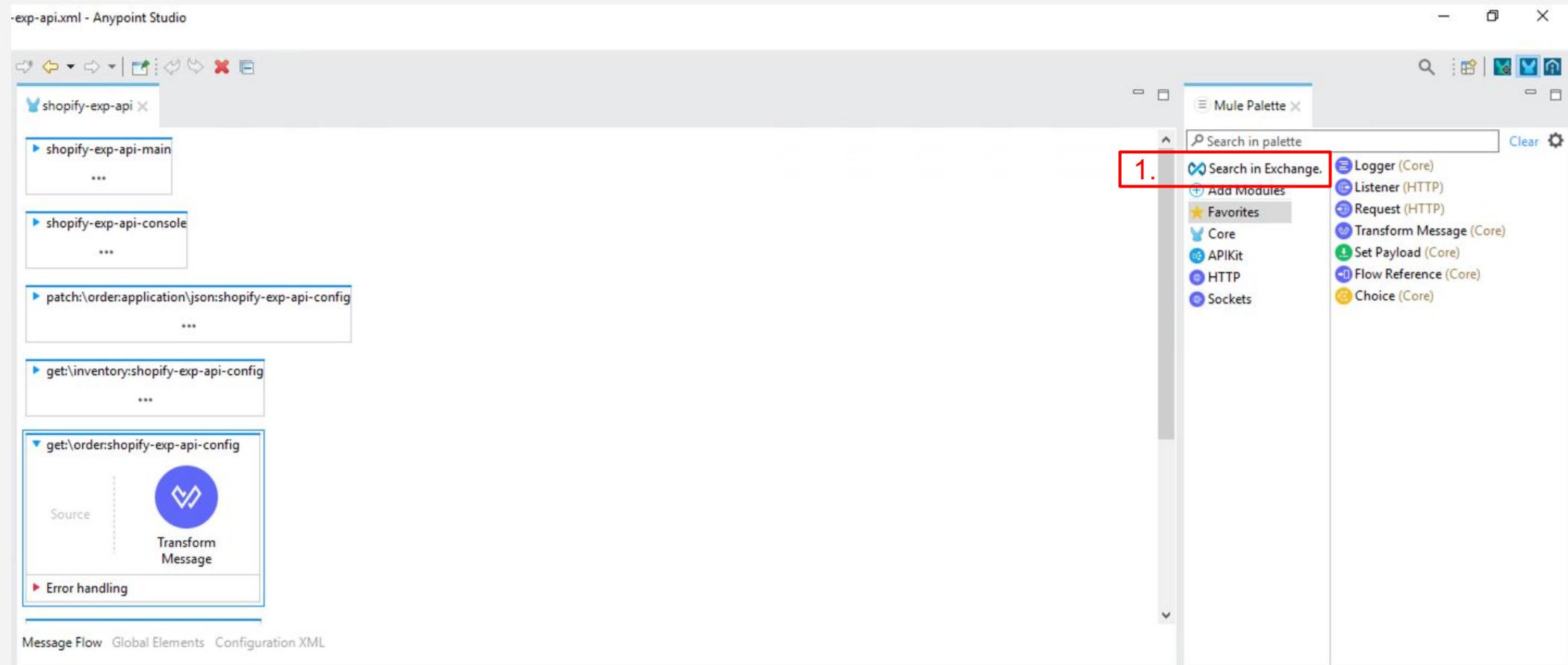


Module 2: Ubiquitous Connectivity: Lab 2 (4/23)



Step 2: Find and load the Shopify Connector

1. Put your cursor to the far right pane called Mule Palette. And click on the **Search in Exchange** option.



Module 2: Ubiquitous Connectivity: Lab 2 (5/23)



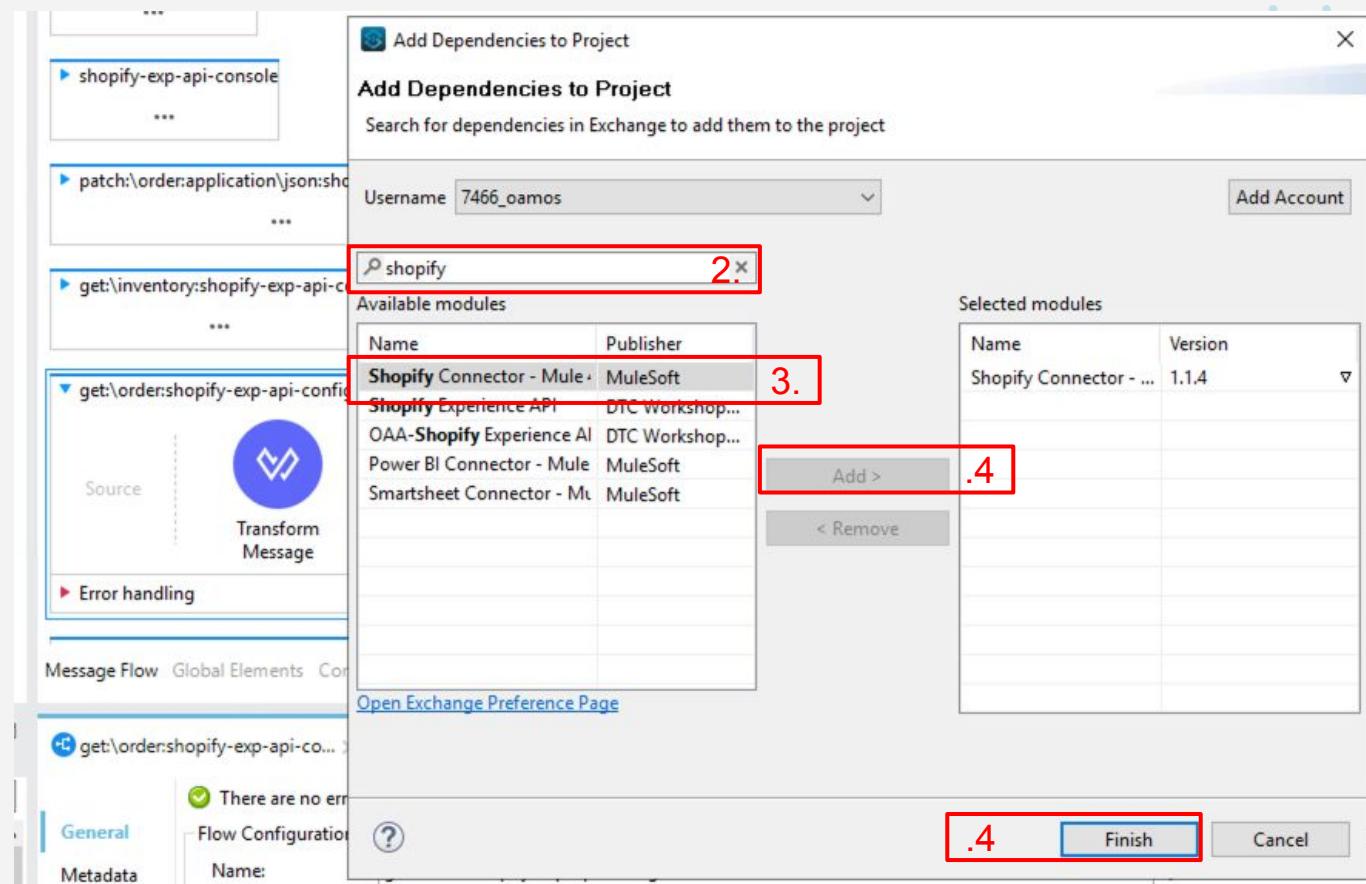
Step 2: Find and load the Shopify Connector

2. The **Add Dependencies to Project** Window will appear. In the search area type “**shopify**”. The **Shopify Connector - MuleSoft** should be the first option to appear.

3. Select the connector to highlight it.

3. Select the connector and click the **Add** button.

4. Then Select **Finish**.

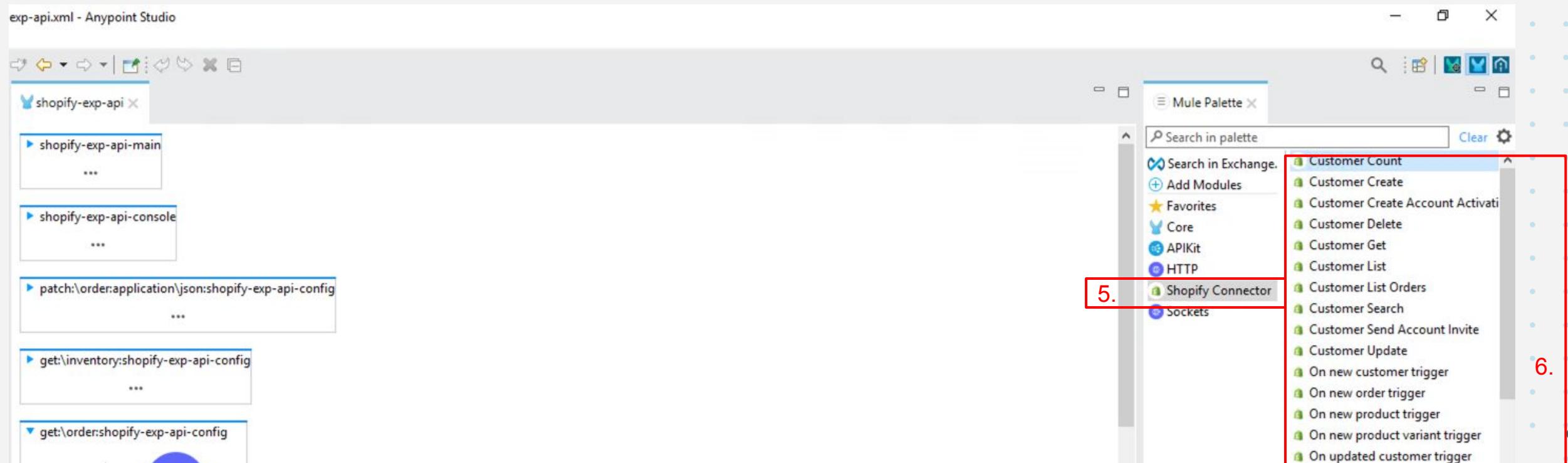


Module 2: Ubiquitous Connectivity: Lab 2 (6/23)



Step 2: Find and load the Shopify Connector

5. The Mule Palette in the right pane should have updated with the Addition of the **Shopify Connector**.
6. On the furthest right sub pane there will be an extensive list of Shopify Operators.

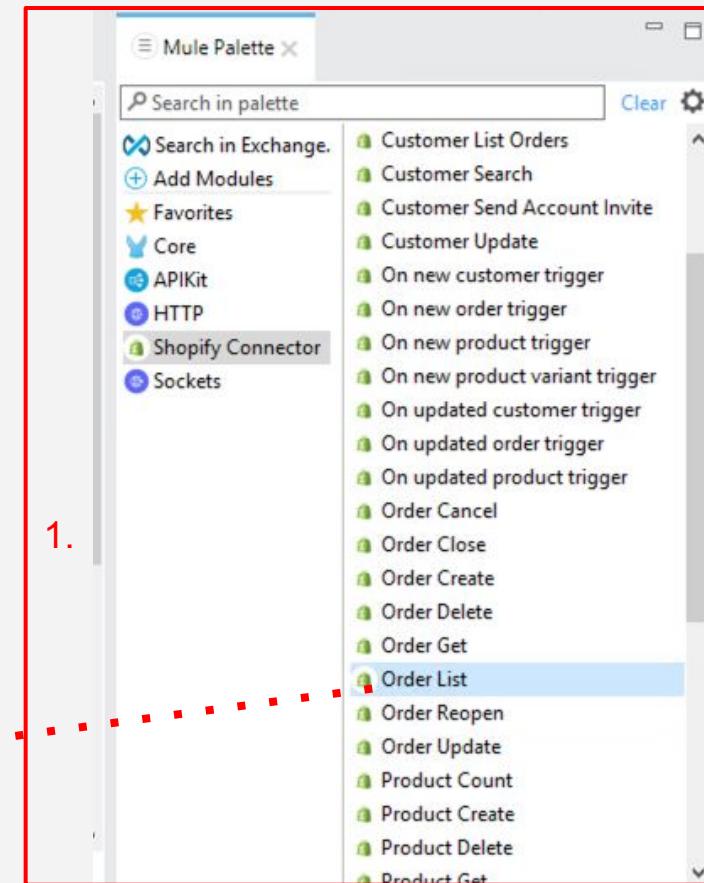


Module 2: Ubiquitous Connectivity: Lab 2 (7/23)



Step 3: Configure and Use Connector Operation

1. In the operator sub palette Find and Select **Order List** This operator is perfect for what we are looking to do, which is retrieve all orders from within the **NTO Shopify Site**.
2. Highlight and drag **Order List** to the **get:\order:shopify-exp-api-config** flow. Try to drag **Order List** next to the **Transform Message** like so:

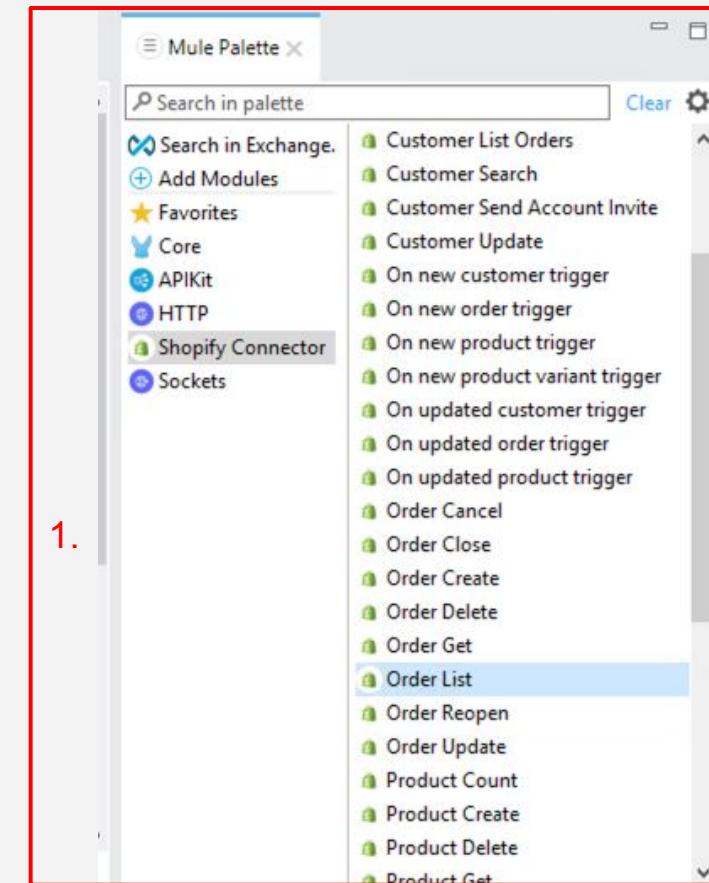
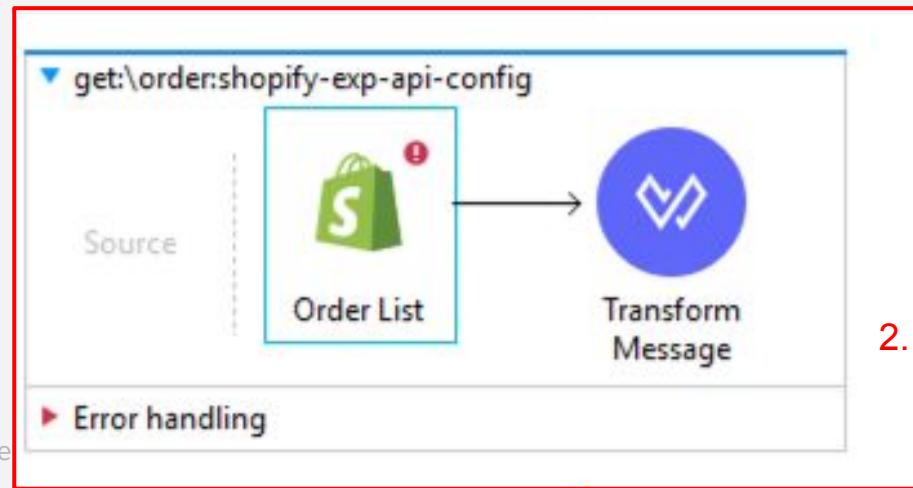


Module 2: Ubiquitous Connectivity: Lab 2 (7/23)



Step 3: Configure and Use Connector Operation

1. In the operator sub palette Find and Select **Order List** This operator is perfect for what we are looking to do, which is retrieve all orders from within the **NTO Shopify Site**.
2. Highlight and drag **Order List** to the **get:\order:shopify-exp-api-config** flow. Try to drag **Order List** next to the **Transform Message** like so:



2.

Module 2: Ubiquitous Connectivity: Lab 2 (8/23)



Step 3: Configure and Use Connector Operation

- Double click on the **Order List** in the `get:\order:shopify-exp-api-config` flow. Look at the lower pane just below the flow and you will see configuration settings with an **Attribute 'config-ref' is required** Warning. We need to configure the connector.

The screenshot shows the MuleSoft Anypoint Studio interface. A red box highlights the 'Order List' configuration pane. The pane displays a warning message: 'Attribute 'config-ref' is required'. The configuration fields include:

- Display Name: Order List
- Connector configuration: (empty field)
- General settings:
 - ids:
 - limit:
 - Since id:
 - Created at min:

On the right side of the interface, there is a sidebar titled 'Shopify Connector' with various trigger options listed. At the bottom right, there is a 'Input' and 'Output' pane showing message structures.

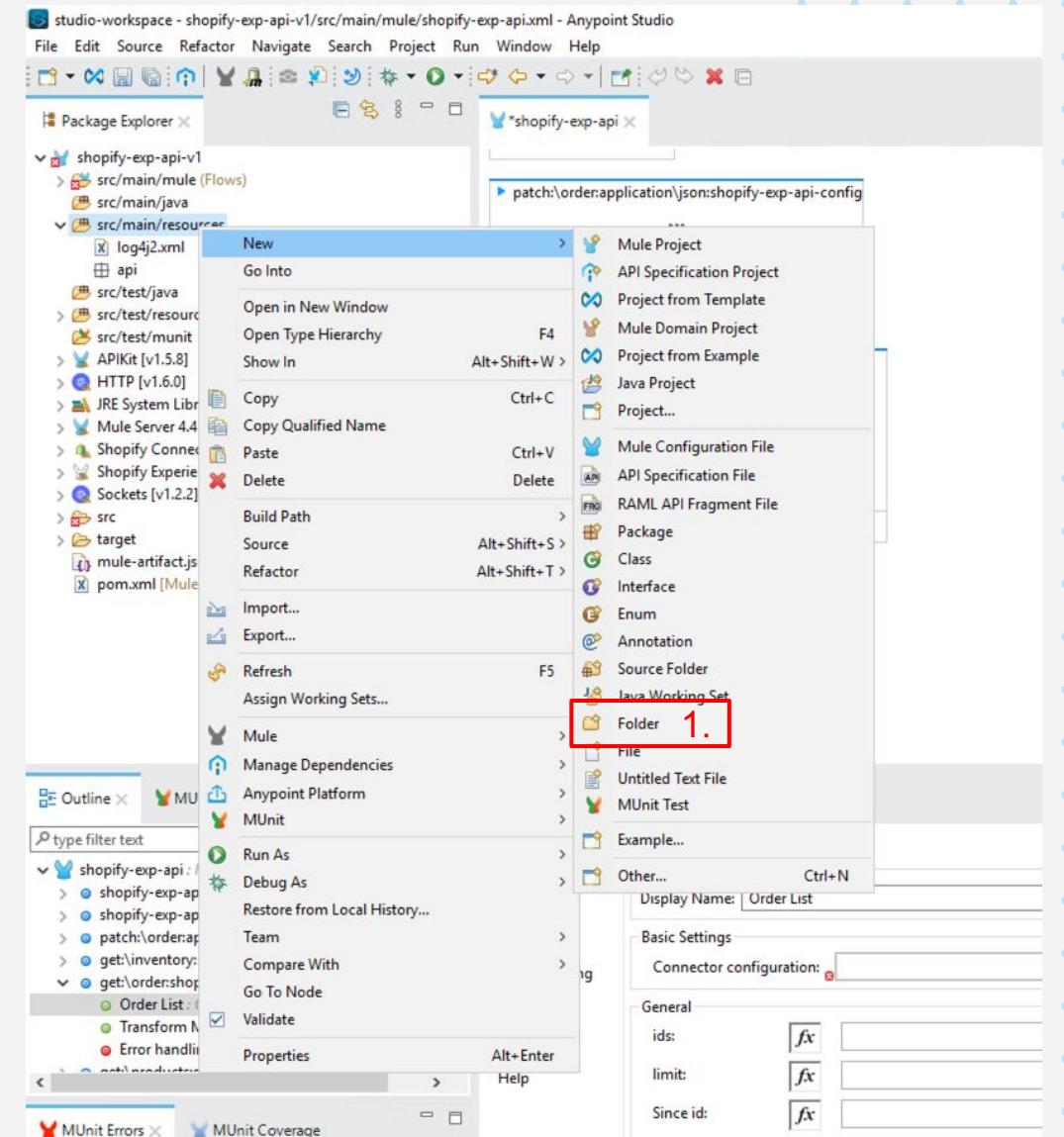
Module 2: Ubiquitous Connectivity: Lab 2 (9/23)



Step 4: Create a Property File

We could enter our Shopify configuration details directly in the below area. But it's a good practice to create a configuration file to hold all the parameterized properties in the project. We are going to create one and configure the connection credentials for our Shopify API implementation. First, we are going to create a folder and then create the config file.

1. In the left pane labeled **Package Explore**, Click on **src/main/resources** in the **Package Explorer** view, right click and create a **New → Folder**, name it **configuration** and click **Finish**

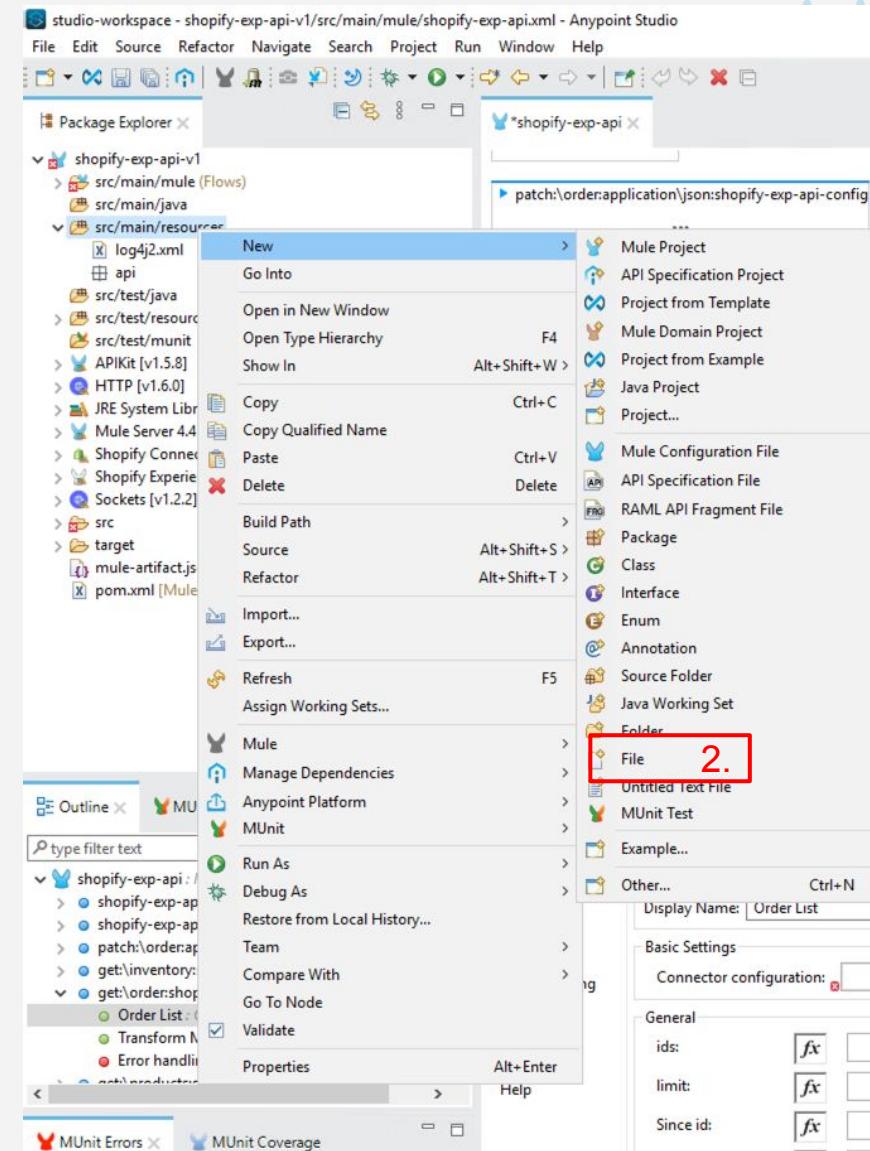
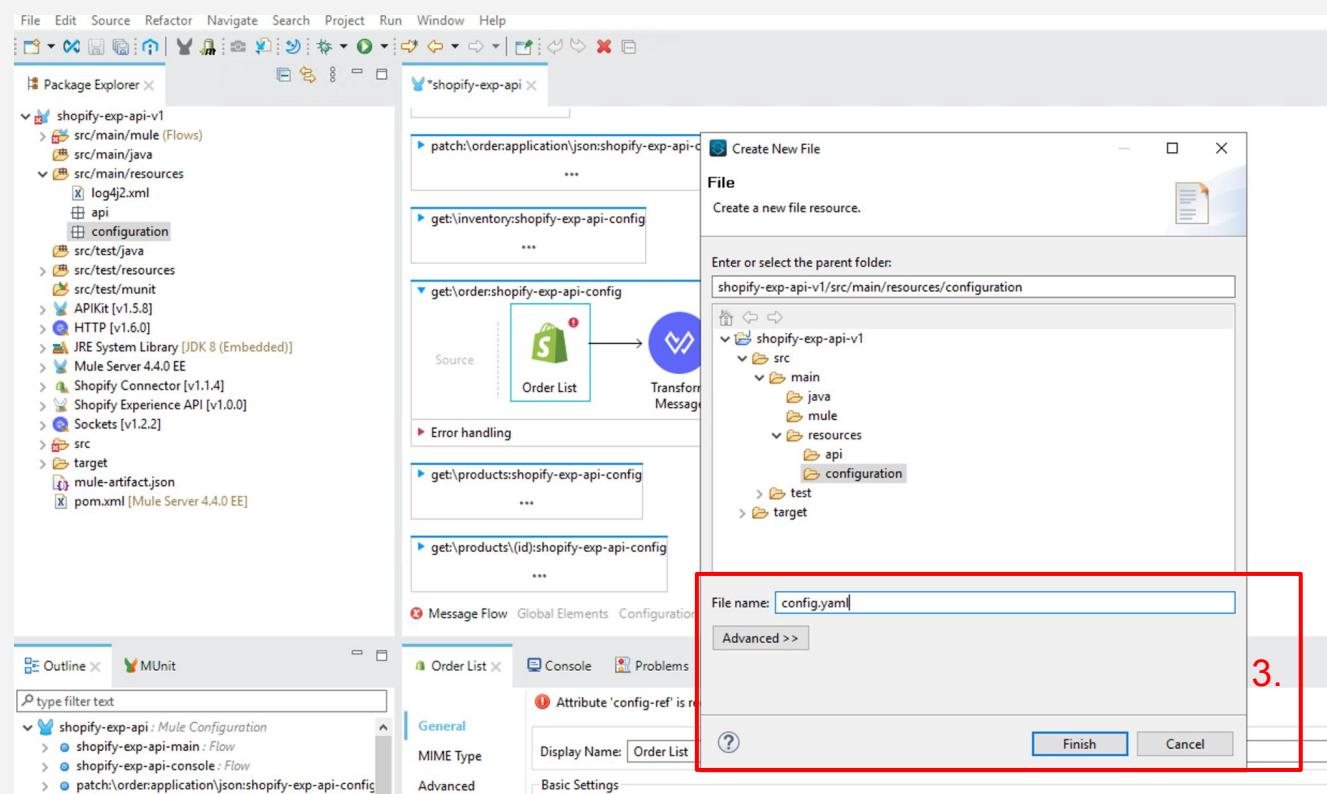


Module 2: Ubiquitous Connectivity: Lab 2 (10/23)



Step 4: Create a Property File

2. Click on **src/main/resources/configuration** in the Package Explorer view, right click and create a **New → File**.
3. Name it **config.yaml** and click **Finish**.



Module 2: Ubiquitous Connectivity: Lab 2 (11/23)

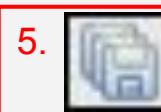


Step 4: Create a Property File

4. Copy the following contents into **config.yaml**. These are the connection credentials your API will use for the Shopify connector.

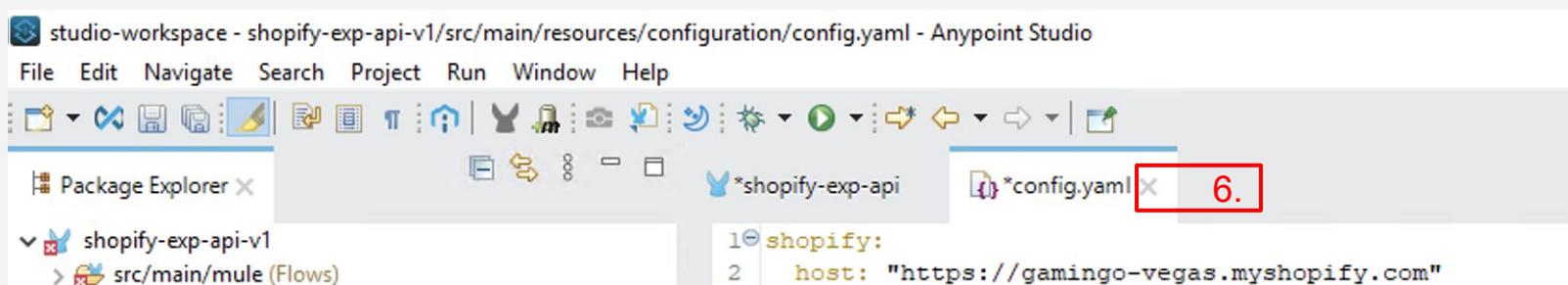
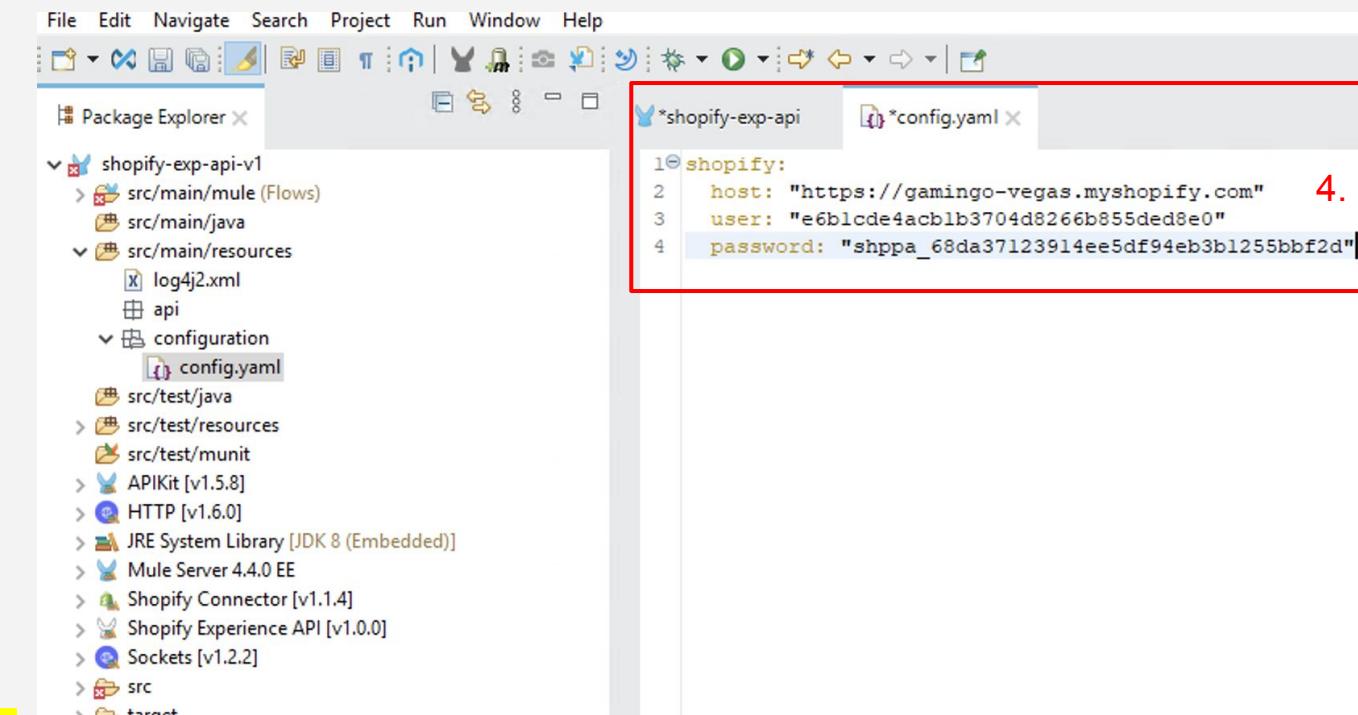
```
shopify:  
  host: "https://gamingo-vegas.myshopify.com"  
  user: "e6b1cde4acb1b3704d8266b855ded8e0"  
  password: "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"
```

5. Click the **Save All** icon on the top Navigation Menu.



6. Close **config.yaml** file tab.

If you have problems copying the code you can find example [here](#)



Module 2: Ubiquitous Connectivity: Lab 2 (12/23)



Step 4: Create a Property File

7. To finish the configuration of the properties file for your API, click on the **shopify-exp-api** view and click on the **Global Elements** tab at the bottom of the pane.

8. Click the **Create** Option.

9. Go to **Global Configurations> Configuration Properties**.

10. Click **OK**.

The screenshot shows the Mule Studio interface with the following steps highlighted:

- Step 7: A red box highlights the "Global Elements" tab in the bottom navigation bar of the main window.
- Step 8: A red box highlights the "Create" button in the context menu of the Global Configuration Elements list.
- Step 9: A red box highlights the "Configuration properties" option in the "Choose Global Type" dialog.
- Step 10: A red box highlights the "OK" button in the "Choose Global Type" dialog.

The main window displays the "Global Configuration Elements" list with three items:

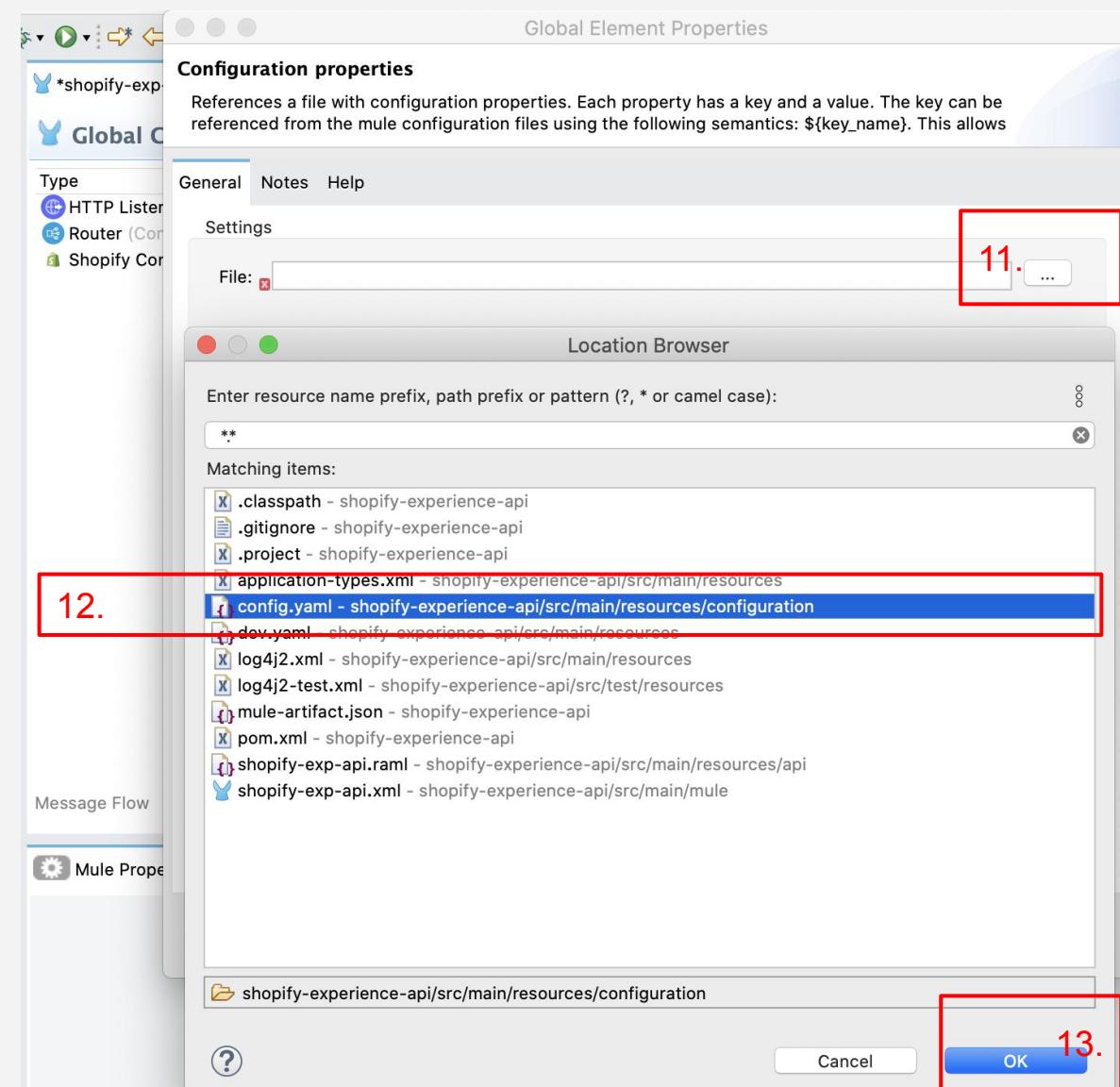
Type	Name	Description
HTTP Listener config	shopify-exp-api-httpListen	
Router (Configuration)	shopify-exp-api-config	
Shopify Connector Con	Shopify_Connector_Config	

Module 2: Ubiquitous Connectivity: Lab 2 (13/23)



Step 4: Create a Property File

11. By **File** Click the ...
12. Select the **config.yaml** file you just created
13. Click **OK**.



Module 2: Ubiquitous Connectivity: Lab 2 (13/23)



Step 4: Create a Property File

14. Navigate down to the lower configuration panel and click on the green “+” by **Connector configuration: Shopify_Connector_Config**

Order List X Console Problems

General

MIME Type

Advanced

Error Mapping

Display Name: Order List

Basic Settings

Connector configuration: **Shopify_Connector_Config**

15. Enter the following in the **Shopify_Connector_Config** Window:

- Username: \${shopify.user}
- Password: \${shopify.password}
- Base Uri: \${shopify.host}

Global Element Properties

Shopify Connector Config

Type: Shopify Connector Config

Name:

Connection: Basic Auth Connection Provider

General Advanced Notes Help

General Proxy Security Advanced

Username: \${shopify.user}

Password: Show password

Base Uri: \${shopify.host}

Message Flow

Mule Properties

Test Connection... Cancel **16. OK**

16. Click **OK**.

Module 2: Ubiquitous Connectivity: Lab 2 (14/23)



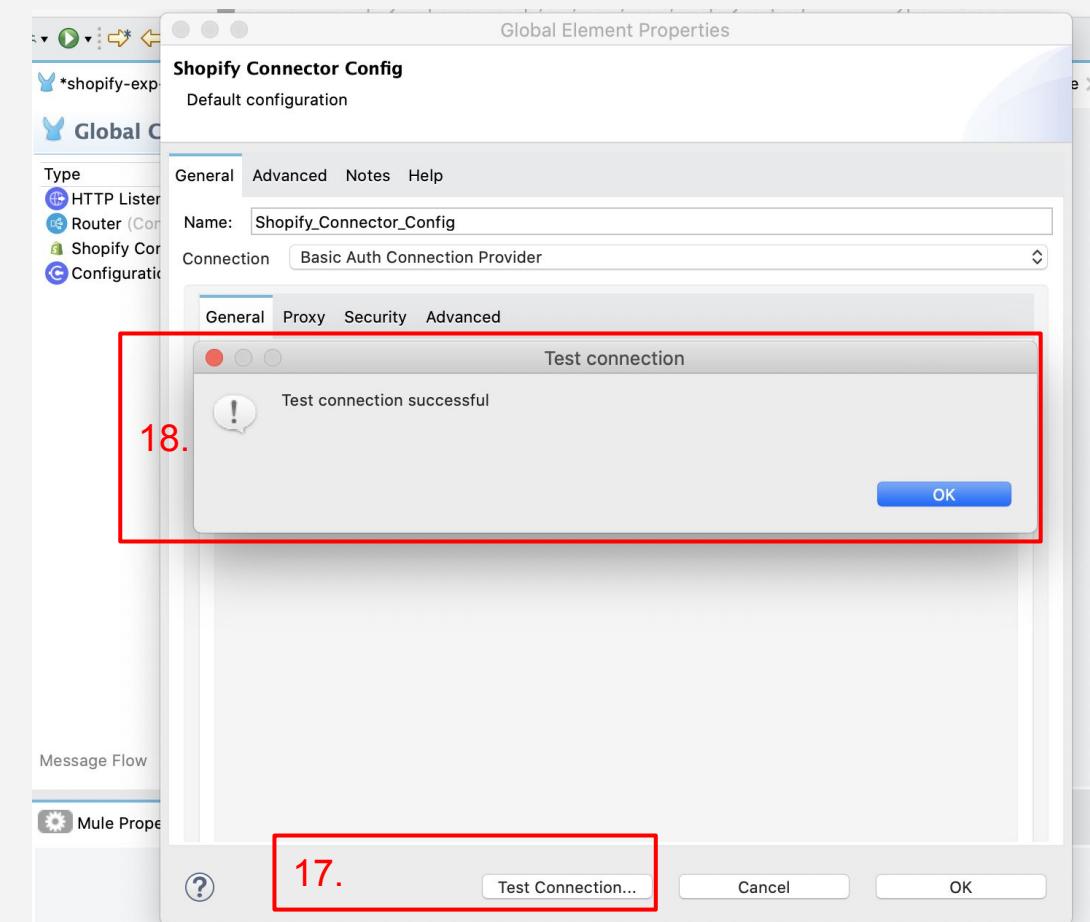
Step 4: Create a Property File

17. Click **Test Connection...** Anypoint Studio will now attempt connecting to the NTO Shopify. **There are several different ways to authenticate to Shopify, but for the sake of the workshop we are using this Basic Authentication method.**

18. You should receive a “**Test connection successful**” Message. Press **OK** to Close the message.

19. Click **OK** again in the **Shopify_Connector_Config** Window.

Now our Shopify API is connected to NTO’s Shopify Account!



Module 2: Ubiquitous Connectivity: Lab 2 (15/23)

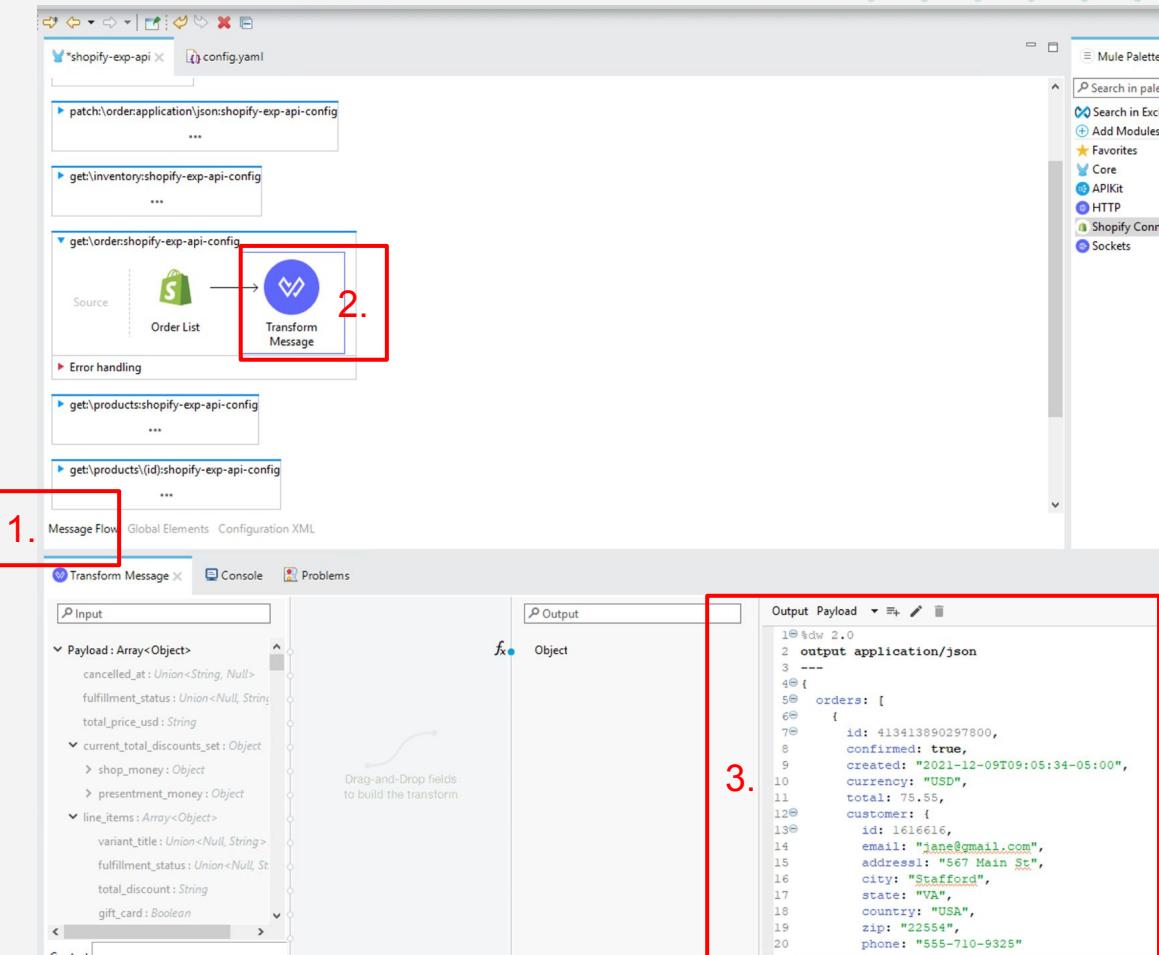


Step 5: Transform Message

1. Click **Message Flow** tab to switch out of the Global Element Pane back to the Message Flow pane where we need to update our `get:\order:shopify-exp-api-config` flow.

2. Move your mouse to the Transform Message in the `get:\order:shopify-exp-api-config` flow and double click it.

3. In the below **Transform Message** pane you will see some code in the right pane called **Output Payload**. This is example response body that we created when we designed the **Shopify API** in **Module 1**.



Module 2: Ubiquitous Connectivity: Lab 2 (16/23)



Step 5: Transform Message

1. Delete the sample code under the Output Payload pane and replace with the following:

```
%dw 2.0
output application/json
---
payload
```

2. Click the **Save icon** at the top right corner of the **Transform Message Pane**. To save you changes.

We have now made the necessary changes to the `get:\order:shopify-exp-api-config` flow and we are now ready to test it!

A screenshot of the MuleSoft Anypoint Studio interface. On the left, the 'Input' pane shows a complex JSON schema for an order object, including fields like 'cancelled_at', 'fulfillment_status', 'total_price_usd', and arrays for 'current_total_discounts_set' and 'line_items'. In the center, a flow diagram shows an 'Object' node connected from the input to an 'Output' node. On the right, the 'Output Payload' pane contains the following DW script:

```
1 %dw 2.0
2 output application/json
3 ---
4 payload
```

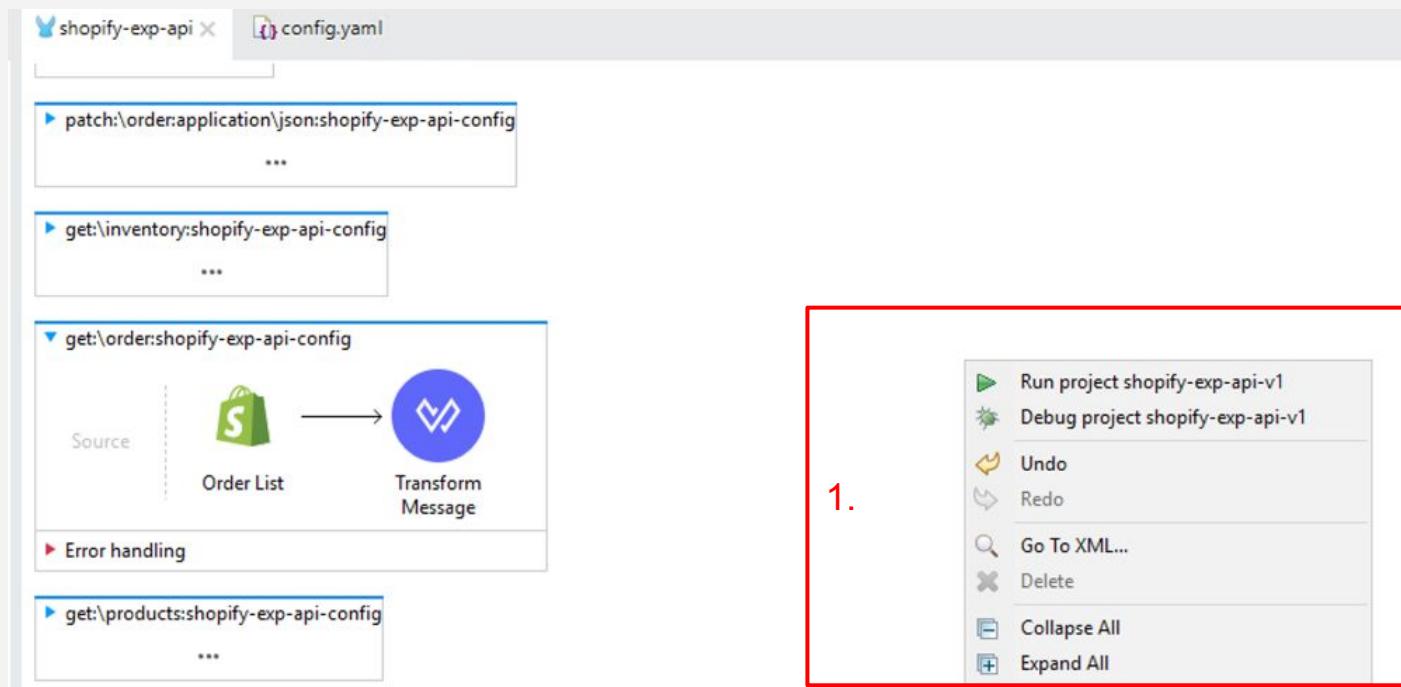
Module 2: Ubiquitous Connectivity: Lab 2 (17/23)



Step 6: Testing the Shopify API Connectivity to NTO's Shopify

In order to test the API we will need to run our project on our local machine.

1. To run the project all you need to do is Right Click in any blank area in the **shopify-exp-api** Message Flow and Select **Run project shopify-experience-api**.



Module 2: Ubiquitous Connectivity: Lab 2 (18/23)



Step 6: Testing the Shopify API Connectivity to NTO's Shopify

While we wait for the project to run let's add our own Shopify order date in the **NTO Shopify Site**!

1. Go to <https://nto-demo.com/>

2. Let's buy the “**Hiking boots**”, by selecting that item.

A screenshot of a web browser displaying the NTO Shopify homepage. The URL bar at the top contains the address "https://nto-demo.com/" with a red box highlighting it. The main content area features a large image of a person hiking through a narrow slot canyon. Overlaid on the image is the text "Built for the Path Less Traveled" and "Run a marathon or climb a mountain, we've got the right footwear for any occasion." Below this is a blue "SHOP FOOTWEAR" button. At the bottom of the page, there is a grid of product thumbnails. One specific thumbnail for "Hiking boots" is highlighted with a red box. Other visible products include a "SKIN BUNDLE SPECIALS" offer, "Special Edition! Boot", "Avery Long Hike Backpack", and "Rugged Tent". A navigation bar at the bottom includes links for "Salesforce", "Aloha!", "Mulesoft - Sign In", "MuleSoft Enablement", "MuleSoft Bloomfire", "Brand | MuleSoft", "Anypoint Manager", "SE Platform", and "New Hire Glossary...". A "Chat with an Expert" button is located in the bottom right corner.

Module 2: Ubiquitous Connectivity: Lab 2 (18/23)



Step 7: Testing the Shopify API Create Order

1. Click **Buy It Now**

2. Add some fake contact details like so and Click **Continue to Shipping**.

Keep record of the fake email you used it will come in handy in subsequent steps.

NTO Demo
Information > Shipping > Payment

Express checkout

PayPal **Google Pay**

OR

Contact information

Email or mobile phone number: muleyoscar@nto.com

Email me with news and offers

Shipping address

First name (optional): Muley Last name: Oscar

Address: 1313 Mockingbird Lane

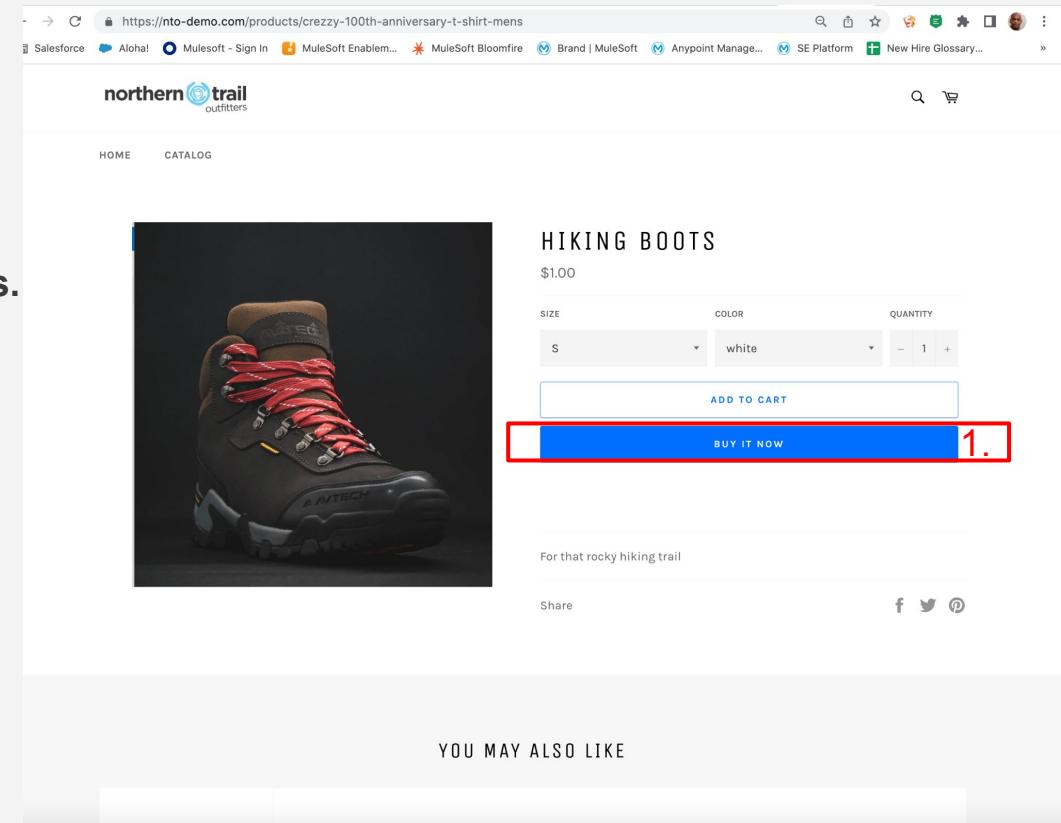
Apartment, suite, etc. (optional):

City: South Pasadena

Country/Region: United States State: California ZIP code: 91030

Continue to shipping

The contact information section is highlighted with a red border, and the "Continue to shipping" button is labeled with a red "2".



Module 2: Ubiquitous Connectivity: Lab 2 (19/23)



Step 7: Testing the Shopify API Create Order

4. Select the **No Charge** Radio Button.
5. Click **Pay Now**.
6. You should see an Order Confirmation. Close out of <https://nto-demo.com/>.

NTO Demo
Confirmation #WG7RPH92
Thank you Muley!

Your order is confirmed
You'll receive a confirmation email with your order number shortly.
[Download Shop to track package](#)

Customer information
Contact information: muleyoscar@nto.com
Payment method: No charge - \$5.90
Shipping address: Mule Oscar, 1313 Mockingbird Lane, South Pasadena CA 91030, United States
Billing address: Mule Oscar, 1313 Mockingbird Lane, South Pasadena CA 91030, United States
Shipping method: Standard

6.

Information > Shipping > Payment

Contact: muleyoscar@nto.com | Change
Ship to: 1313 Mockingbird Lane, South Pasadena CA 91030, United States | Change
Method: Standard - \$4.90 | Change

Payment
All transactions are secure and encrypted.

Credit card
 PayPal

4. No charge

no charge

Billing address
Select the address that matches your card or payment method.

Same as shipping address
 Use a different billing address

5. **Pay now** [Return to shipping](#)

Module 2: Ubiquitous Connectivity: Lab 2 (20/23)



Step 7: Testing the Shopify API Create Order

7. Switch back to **Anypoint Studio** and by now your project should be in a **DEPLOYED** status

8. Click on **Open console** in the left lower pane labeled

API Consoles

The screenshot shows the Anypoint Studio interface with the following details:

- Package Explorer:** Shows the project structure for "shopify-experience-api" with various files like "shopify-exp-api.xml", "application-types.xml", and "config.yaml".
- Mule Palette:** On the right, it lists components such as "Logger (Core)", "Listener (HTTP)", "Request (HTTP)", "Core", "APIKit", "HTTP", "Shopify Connector", and "Sockets".
- Message Flow Editor:** Displays three flows:
 - "get:order:shopify-exp-api-config" which takes an "Order List" from a "Source" and transforms it into a "Transform Message".
 - "get:products:shopify-exp-api-config" which takes a "Source" and transforms it into a "Transform Message".
 - "get:products\{id\}:shopify-exp-api-config" which takes a "Source" and transforms it into a "Transform Message".
- Console:** Shows deployment logs:

```
INFO 2022-04-16 21:35:57,060 [WrapperListener_start_runner] org.eclipse.jetty.server.AbstractConnector: Started ServerConnector@3e8a9
*****
* - + DOMAIN + - * - + STATUS + - - *
*****
* default * DEPLOYED *
*****
*****
```
- MUnit Errors:** Shows no errors.
- Annotations:** A red box highlights the "Open console" button in the API Consoles pane, and another red box highlights the deployment log output.
- Page Number:** A red number "7." is in the bottom right corner.

Module 2: Ubiquitous Connectivity: Lab 2 (21/23)



Step 7: Testing the Shopify API Create Order

9. A Web Browser tab will get launch to show the **API Console** Web UI. Click **/order GET** button to test our flow out.

The screenshot shows a web browser window displaying the API Console for the Shopify Exp API. The URL is `http://localhost:8081/console/`. The main title is "Shopify Exp API". On the left, there's a sidebar with endpoints: "/order", "/products", "/{id}", "/inventory", and "/{location}/{id}/{quantity}". The "/order" endpoint has three methods listed: "GET" (highlighted in green), "PATCH", and "POST". Below these, under "/products", there is a single "GET" method. The right side of the interface displays detailed information for the "/order" endpoint, including the API title ("Shopify Exp API"), version ("2.0.1"), supported protocols ("HTTPS"), and the base URL ("`http://localhost:8081/api`").

Module 2: Ubiquitous Connectivity: Lab 2 (22/23)



Step 7: Testing the Shopify API Create Order

10. You will be taken to a view for the **GET /order** resource endpoint. Click the send button **Send**.

The screenshot shows the API console interface for the Shopify Exp API. The left sidebar lists endpoints: /order (selected), /products, /products/{id}, /inventory, and /location/{id}/{quantity}. The main area displays the details for the /order endpoint. The method is set to GET, and the URL is http://localhost:8081/api/order. The response status is 200. The body media type is application/json, and the example shows a JSON object with an 'orders' key containing an array of order details. The 'Send' button at the bottom right is highlighted with a red box and contains the number 10.

Request URL
http://localhost:8081/api/order

Query parameters
 Show optional parameters
[Add](#)

Headers
 Text editor
[COPY](#)
Add a header to the HTTP request.
[Add](#)

Send **10.**

Module 2: Ubiquitous Connectivity: Lab 2 (23/23)



Step 7: Testing the Shopify API Create Order

11. Response will show in the lower right pane with a large payload of an array of orders.

12. Do a **MAC: Command + F, PC: Ctrl +F**, and search for the fake email you used for your order.

The Shopify response is a very comprehensive payload take a look at the structure. In future Lab updates we will show you how to make this payload tailored and more readable.

Great work! We successfully tested our Implementation now lets deploy it.

Proceed to Lab 3

The screenshot shows the MuleSoft API console interface. At the top, there are several tabs: Salesforce, Aloha!, Mulesoft - Sign In, MuleSoft Enablement, MuleSoft Bloomfire, Brand, and one with the user's email address, muleyoscar@nto.com. The main area has a title "Shopify Exp API". On the left, there's a sidebar with "API console" and "Shopify Exp API" sections, and a tree view of endpoints: /order (with GET, PATCH, POST methods), /products, /{id}, /inventory, and /{location}/{id}/{quantity}. The main content area shows a "Responses" section for a "GET http://localhost:8081/api/order" request. It displays a "Body" with a JSON example and a "Responses" table with a single row for status code 200. The response body is a large JSON object representing an array of orders. A red box highlights the user's email address in the top navigation bar. Another red box highlights the number 12 in the top right corner. A third red box highlights the user's email address again in the response body. A fourth red box highlights the number 11 in the bottom right corner of the response body. A fifth red box highlights the number 12 in the bottom right corner of the response body.

```
{
  "orders": [
    {
      "id": 413413890297800,
      "confirmed": true,
      "created": "2021-12-09T09:05:34-05:00",
      "currency": "USD",
      "total": 75.55,
      "customer": {
        "id": 1616616,
        "email": "jane@gmail.com",
        "address1": "567 Main St",
        "city": "Stafford",
        "state": "VA",
        "country": "USA",
        "zip": "22554",
        "phone": "555-710-9325"
      },
      "line_items": [
        {
          "id": 10417352802515,
          "sku": "841483107332",
          "name": "Waist Cincher - Soft Nude / L",
          "quantity": 1,
          "gift_card": false,
          "tax_code": "PC040100"
        }
      ],
      "note": null
    },
    {
      "id": 41341389029771,
      "confirmed": true,
      "created": "2021-12-09T09:05:32-05:00",
      "currency": "USD",
      "total": 65.55,
      "customer": {
        "id": 1616616,
        "email": "john@gmail.com",
        "address1": "124 Main St",
        "city": "Chicago",
        "state": "IL",
        "country": "USA",
        "closed_at": null,
        "confirmed": true,
        "contact_email": "muleyoscar@nto.com",
        "created_at": "2022-04-16T21:19:05-07:00",
        "currency": "USD",
        "current_subtotal_price": "1.00",
        "current_subtotal_price_set": true
      }
    }
  ]
}
```



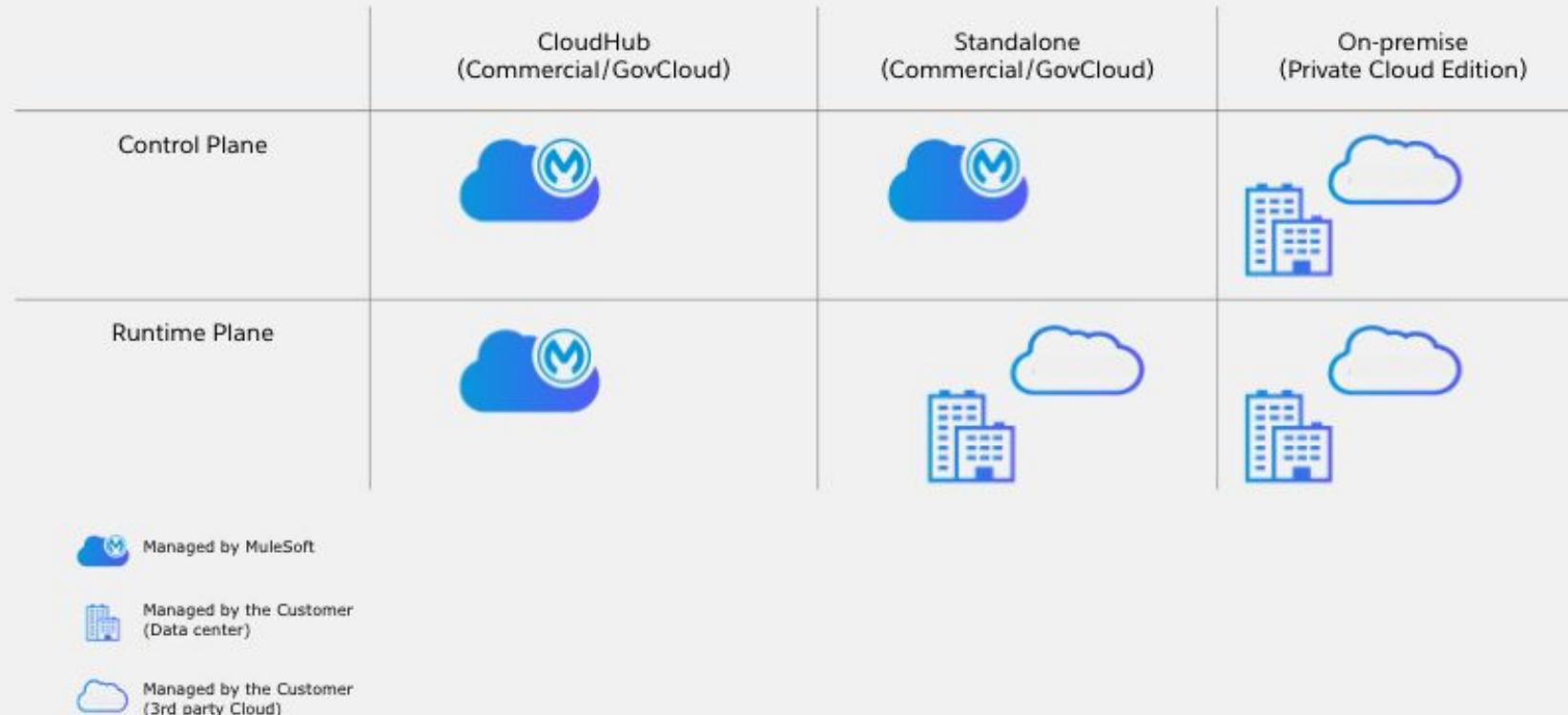
Module 2: Lab 3 Deploy the Shopify API Implementation

Module 2: API Deployment : Overview



Overview

Anypoint projects can be deployed in a number of ways (via our Cloud, On Premise Data Center, 3rd Party (AWS, Google,Azure), or a mix of all. This workshop will show how deployment can be done in our Cloud managed offering, **CloudHub**.



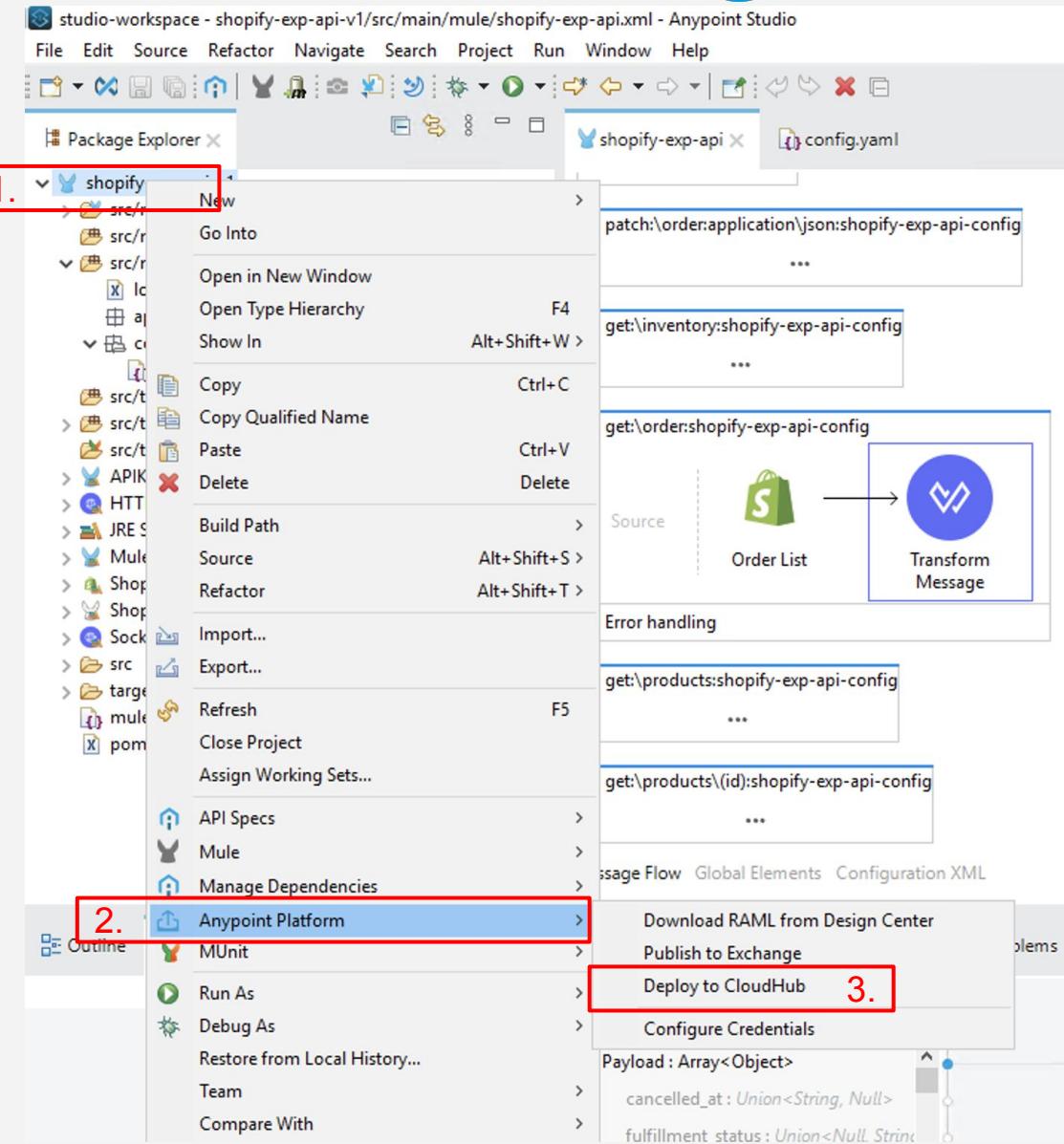
- **Footnote:** In some of our more advanced workshops we also show complete CI/CD process of deployment. Reach out to your Account Executive if you like to learn more about MuleSoft's CI/CD capabilities.

Module 2: Ubiquitous Connectivity: Lab 3 (2/10)



Step 1: Deploy Shopify Project from Anypoint Studio

1. With your mouse select the **shopify-experience-api** under Package Explorer in the left pane and **Right Mouse Click**.
2. Menu option should appear scroll down and select **Anypoint Platform**.
3. Sub menu will appear to the right. Select **Deploy to Cloud**.



Module 2: Ubiquitous Connectivity: Lab 3 (3/10)



Step 1: Deploy Shopify Project from Anypoint Studio

4. Anypoint login window will appear. Enter your Anypoint credentials and click

Sign In.

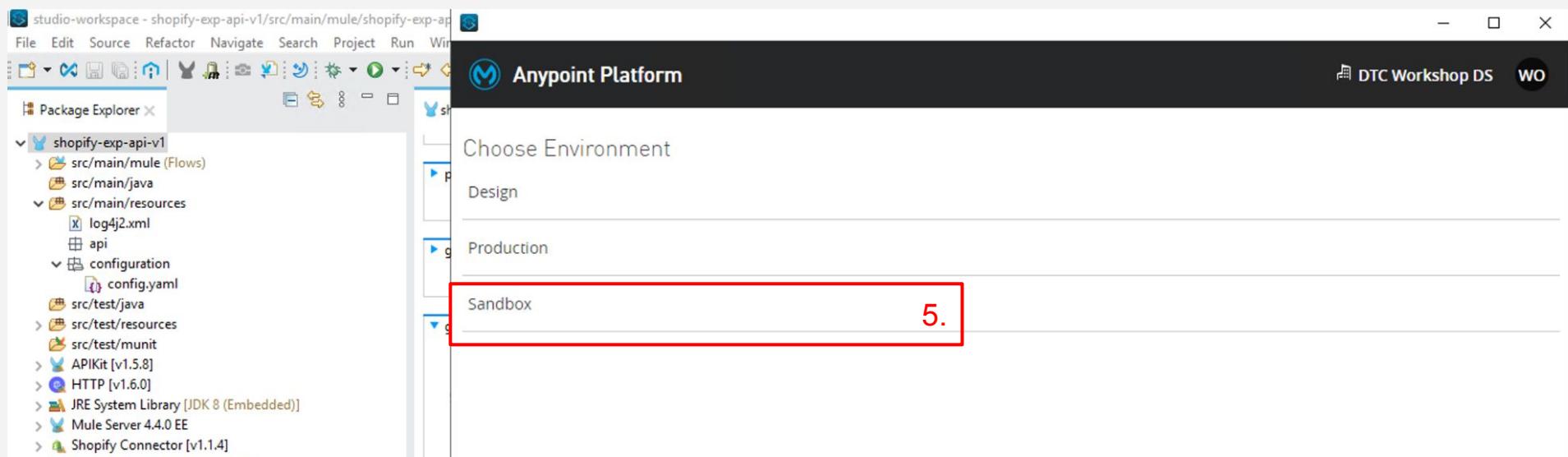
A screenshot of the Anypoint Studio interface. On the left, the 'Package Explorer' shows a project named 'shopify-exp-api-v1' with various source code and configuration files listed. On the right, a browser window titled 'Anypoint Platform' displays the login page at 'anypoint.mulesoft.com/login/signin?'. The login form has a red box around it, highlighting the 'Sign in to Anypoint Platform' header, the username field containing '7466_oamos', and the password field containing '.....'. A red number '4.' is placed to the left of the password field. Below the form are links for 'Forgot your credentials?' and 'Use custom domain'. At the bottom of the browser window, there are links for 'Don't have an account?' and 'Sign up'. The status bar at the bottom of the studio shows 'Outline', 'APIkit Consoles X', and 'MUnit'.

Module 2: Ubiquitous Connectivity: Lab 3 (4/10)



Step 1: Deploy Shopify Project from Anypoint Studio

5. Under **Choose Environment** Select **Sandbox**.



Module 2: Ubiquitous Connectivity: Lab 3 (5/10)



Step 1: Deploy Shopify Project from Anypoint Studio

6. Under Deployment Application name you deployment “**<first name or initials>-shopify-exp-api-v1**”.

7. Click **Deploy Application**.

The screenshot shows the Anypoint Platform interface with the 'Deploying Application' dialog open. The 'Deployment Target' field contains the value 'oaa-shopify-exp-api-v1', which is highlighted with a red box and labeled '6.'. At the bottom right of the dialog, there is another red box around the 'Deploy Application' button, labeled '7.'.

Anypoint Platform - Deploying Application

Deployment Target: oaa-shopify-exp-api-v1 (6.)

Runtime: 4.4.0

Properties

Runtime version: 4.4.0

Worker size: 0.1 vCores

Workers: 1

Insight

To use Monitoring and Visualizer with this version, you may need to enable the agent after deploying. [Learn how](#)

Logging

Static IPs

Automatically restart application when not responding

Persistent queues | Encrypt persistent queues

Use Object Store v2

Cancel (7.) Deploy Application (7.) 116

Anypoint Studio - Package Explorer

Shopify project structure:

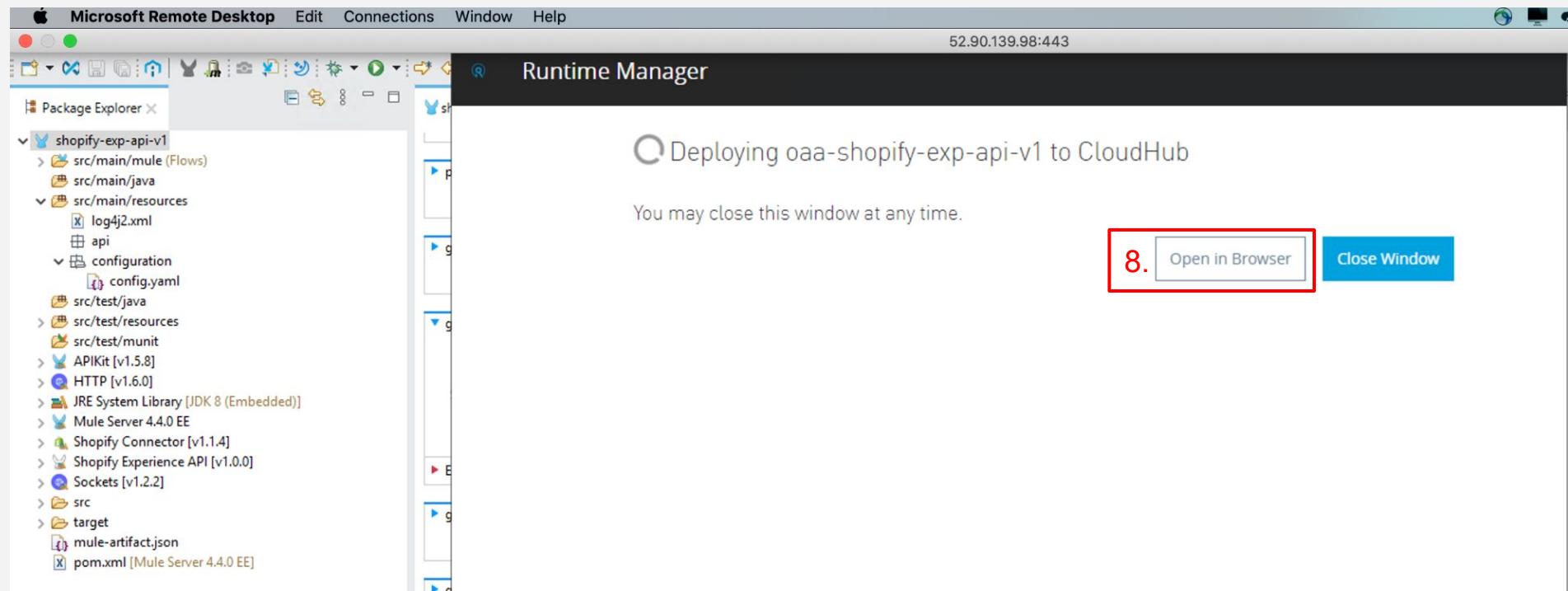
- src/main/mule (Flows)
- src/main/java
- src/main/resources
 - log4j2.xml
 - api
 - configuration
 - config.yaml
- src/test/java
- src/test/resources
- src/test/munit
- APIKit [v1.5.8]
- HTTP [v1.6.0]
- JRE System Library [JDK 8 (Embedded)]
- Mule Server 4.4.0 EE
- Shopify Connector [v1.1.4]
- Shopify Experience API [v1.0.0]
- Sockets [v1.2.2]
- src
- target
 - mule-artifact.json
 - pom.xml [Mule Server 4.4.0 EE]

Module 2: Ubiquitous Connectivity: Lab 3 (6/10)



Step 1: Deploy Shopify Project from Anypoint Studio

8. Screen will show that your application is **Deploying**. Click **Open in Browser**.



Module 2: Ubiquitous Connectivity: Lab 3 (7/10)



Step 1: Deploy Shopify Project from Anypoint Studio

9. Screen will show that your application is **Deploying**.

The screenshot shows the Anypoint Runtime Manager interface. On the left, there is a sidebar with a tree view of project files for 'shopify-exp-api-v1'. The main area is titled 'Runtime Manager' with tabs for 'Sandbox' (selected), 'Applications', 'Servers', 'Alerts', 'VPCs', and 'Runtime Fabrics'. A large button labeled 'Deploy application' is visible. Below it, a search bar says 'Search Applications' and a link 'Switch back to classic applications list'. The main table lists 'All Applications (1)'. The table has columns: Name, Server, Status, Runtime Version, and Date Modified. One row is shown: 'oaa-shopify-exp-api-v1' is running on 'CloudHub' with a status of 'Deploying', runtime version '4.4.0', and last modified '2022-04-17 16:21:26'. A red box highlights the entire interface, and a red number '9.' is in the bottom right corner.

Name	Server	Status	Runtime Version	Date Modified
oaa-shopify-exp-api-v1	CloudHub	Deploying	4.4.0	2022-04-17 16:21:26

Module 2: Ubiquitous Connectivity: Lab 3 (8/10)



Step 1: Deploy Shopify Project from Anypoint Studio

10. Wait until you see that status has changed to **Started**. Click on the <first name or initials>shopify-exp-api-v1.

The screenshot shows the 'Runtime Manager' interface in Anypoint Studio. On the left, there is a sidebar with a tree view of the project structure for 'shopify-exp-api-v1'. The tree includes nodes for 'src/main/mule (Flows)', 'src/main/java', 'src/main/resources' (containing 'log4j2.xml' and 'api'), 'configuration' (containing 'config.yaml'), 'src/test/java', 'src/test/resources', 'src/test/munit', 'APIKit [v1.5.8]', 'HTTP [v1.6.0]', 'JRE System Library [JDK 8 (Embedded)]', 'Mule Server 4.4.0 EE', 'Shopify Connector [v1.1.4]', 'Shopify Experience API [v1.0.0]', and 'Sockets [v1.2.2]'. The main area is titled 'Runtime Manager' with a 'Sandbox' tab selected. It features a 'Deploy application' button, a search bar, and a table titled 'All Applications (1)'. The table has columns for 'Name', 'Server', 'Status', 'Runtime Version', and 'Date Modified'. A single row is listed: '10. oaa-shopify-exp-api-v1' (with '10.' highlighted by a red box), running on 'CloudHub' with a green 'Started' status, runtime version '4.4.0', and last modified on '2022-04-17 16:23:13'.

Name	Server	Status	Runtime Version	Date Modified
10. oaa-shopify-exp-api-v1	CloudHub	Started	4.4.0	2022-04-17 16:23:13

Module 2: Ubiquitous Connectivity: Lab 3 (9/10)



Step 1: Deploy Shopify Project from Anypoint Studio

11. You will be taken to a Dashboard screen **Right Click** on the <first name or initials>shopify-exp-api-v1 and select **Open link in new Tab**.

The screenshot shows the Anypoint Management Center interface. On the left, a sidebar lists 'Sandbox', 'Applications' (with 'Dashboard' selected), 'Insight', 'Logs', 'Object Store', 'Queues', 'Schedules', and 'Settings'. The main area displays the 'Runtime Manager' for the application 'oaa-shopify-exp-api-v1'. It shows a table of Mule messages with columns for timestamp and duration. A context menu is open over the domain link 'oaa-shopify-exp-api-v1' in the 'Mule messages' section, with the 'Open link in new tab' option highlighted by a red box labeled '11.'.

Module 2: Ubiquitous Connectivity: Lab 3 (10/10)



Step 1: Deploy Shopify Project from Anypoint Studio

12. Replace browser URL with: <http://<your first name or initial>-shopify-exp-api-v1.us-e2.cloudbu.io/api/order> and hit Enter/Return.

You should get a payload response like so.

Now we have successfully deployed our Shopify API!

Congratulations! You have completed Module 2!

A screenshot of a web browser window titled 'Anypoint Management Center'. The address bar shows 'oaa-shopify-exp-api-v1.us-e2.cloudbu.io/api/order' with a red box around it and the number '12.' next to it. The page content displays a JSON object representing an order payload.

```
{
  "id": 4724450984187,
  "admin_graphql_api_id": "gid://shopify/Order/4724450984187",
  "app_id": 580111,
  "browser_ip": "13.110.54.38",
  "buyer_accepts_marketing": false,
  "cancel_reason": null,
  "cancelled_at": null,
  "cart_token": null,
  "checkout_id": 32664397873403,
  "checkout_token": "a0836e877f17da8955c7851170fc9232",
  "client_details": {
    "accept_language": "en",
    "browser_height": null,
    "browser_ip": "13.110.54.38",
    "browser_width": null,
    "session_hash": null,
    "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.127 Safari/53"
  },
  "closed_at": null,
  "confirmed": true,
  "contact_email": "muleyoscar@nto.com",
  "created_at": "2022-04-16T21:19:05-07:00",
  "currency": "USD",
  "current_subtotal_price": "1.00",
  "current_subtotal_price_set": {
    "shop_money": {
      "amount": "1.00",
      "currency_code": "USD"
    },
    "presentment_money": {
      "amount": "1.00",
      "currency_code": "USD"
    }
  },
  "current_total_discounts": "0.00",
  "current_total_discounts_set": {
    "shop_money": {
      "amount": "0.00",
      "currency_code": "USD"
    },
    "presentment_money": {
      "amount": "0.00",
      "currency_code": "USD"
    }
  }
}
```

Proceed to Module 3



Module 3: Overview

Module 3: Apply Policy to Shopify API: Overview



Overview

In these labs you will **configure policies and SLAs** that will allow you to manage security, quality of service to the **Shopify API**. In addition, you will **monitor your API through API analytics**.

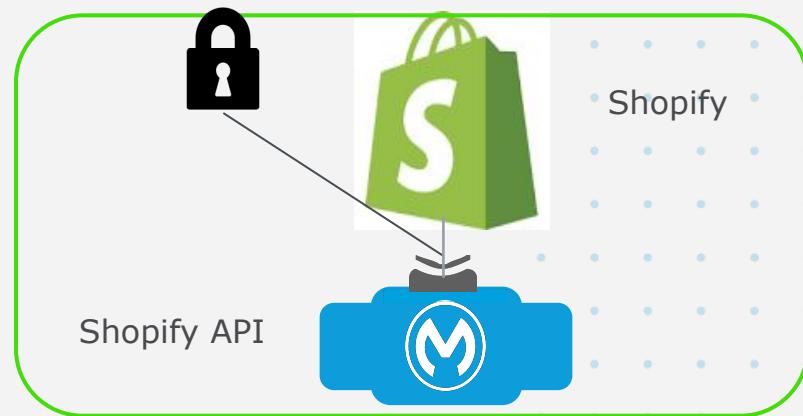
We will proxy an existing API in Lab 1 and You can:

- Manage [security](#) and enforce [policies](#)
- Manage [contracts](#) associated with that API
- Monitor the [API](#) with analytics and KPI metrics in a dashboard view

This makes it easy for the **API manager** to understand how the API is performing, how it's being used and by whom, and for the API Administrator to identify potential issues before they arise.

The 1 lab in this unit is:

- [Lab 1: Proxy an existing Shopify API & Apply Policies to Shopify API](#)





Module 3: Lab 1

Module 3: Apply Policy to Shopify API: Lab 1 (1/23)



Overview

API management is essential to an API-Led architecture as it provides a governance framework to your APIs in all three layers.

For API Management to take place, we need to be able to host our APIs, both new and existing, on an API Gateway that will be used for enforcing policies and collecting data for analytics.

MuleSoft can apply governance directly to a MuleSoft implemented API or through a proxy gateway for these and other external APIs.

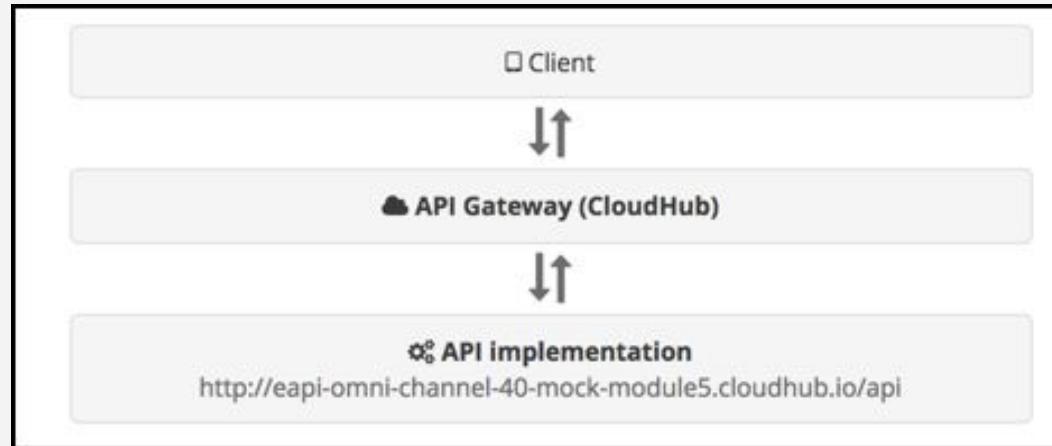
The APIs we will use are two versions of the Shopify API. You will create a proxy gateway to an already deployed mock implementation in this Lab 1.

In this lab, we will define an API that will act as a proxy for NTO's Shopify API. The proxy will be deployed to Anypoint Platform's API Gateway, which is powered by the Mule runtime.

Module 3: Apply Policy to Shopify API: Lab 1 (2/23)



Clients will access the API through the API Gateway which will then forward the requests to the actual Omni Channel API Mock Implementation. Having the proxy deployed on the API gateway allows Anypoint Platform to manage, control access and monitor the usage of the API, which we will look at in the succeeding labs.



Module 3: Apply Policy to Shopify API: Lab 1 (3/23)



Step 1: Go to API Manager

1. In Anypoint Platform (<https://anypoint.mulesoft.com>), click the **API Manager** icon to start creating your API.

The screenshot shows the Anypoint Platform dashboard with several sections:

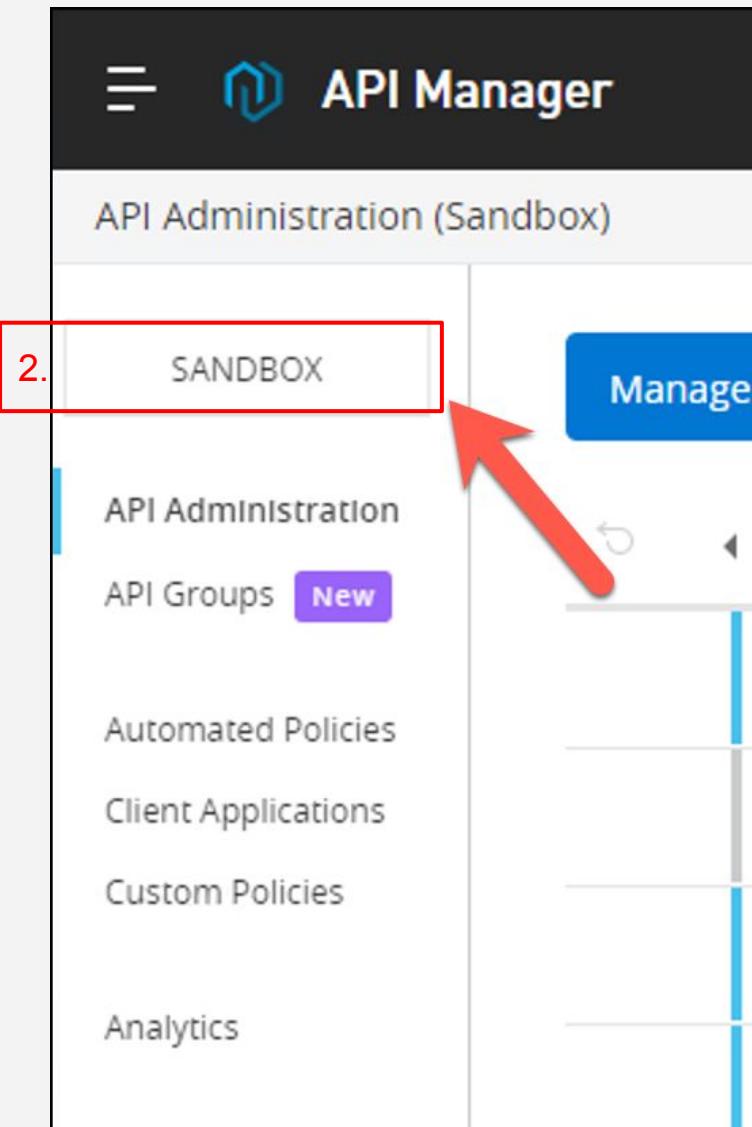
- Design Center**: Get started creating Mule applications and APIs. Includes a "Start designing" button.
- DataGraph**: Unify many APIs into a single graph of data, and consume exactly the data you need in a single request. Includes a "Unify & consume" button.
- Tutorials**: Getting started with MuleSoft - Hello Mule →. First time using MuleSoft? Build an API in under 10 minutes and deploy it to CloudHub.
- Exchange**: Discover and share reusable APIs, connectors, and templates. Includes a "Discover & share" button.
- Management Center**
- Access Management**: Manage users, business groups, and audit logs.
- API Manager**: Manage clients, policies, SLAs, traffic, and alerts. This item is highlighted with a red box and a numbered callout "1.".
- Runtime Manager**: Deploy, manage & monitor deployed applications.
- Data Gateway**: Access data with Salesforce Connect.
- MQ**: Send messages using queues & pub-sub services.

Module 3: Apply Policy to Shopify API: Lab 1 (4/23)



Step 1: Go to API Manager

2. Select **SANDBOX** as the environment



Module 3: Apply Policy to Shopify API: Lab 1 (5/23)



Step 2: Configure an API Proxy

For this lab, we are going to configure the API as a proxy to NTO's Shopify API Implementation. The API is available as an HTTP Restful API accessible through the base URL <http://shopify-exp-api-v1.us-e2.cloudhub.io/api>.

Let's take a look to the products resource of this API that returns product information when an HTTP GET is issued.

To see the all the resources of this API, you can view the API definition and Summary in Exchange

A screenshot of the MuleSoft Exchange interface. The main page displays the 'Shopify Experience API' listing. The listing includes a brief description: 'Direct-to-Consumer Shopify API will allow NTO to reach their consumers directly.', the REST API endpoint 'http://shopify-exp-api-v1.us-e2.cloudhub.io/api', the owner 'Oscar Amos', and the last update time 'Updated 3 hours ago'. It also shows the version '2.0.1 Private' and '1.0.x' status. On the left, there's a sidebar with navigation links like 'Assets list', 'PAGES', 'Home', 'SPECIFICATION', 'Summary', 'Endpoints', 'Order', '/products', '/(id)', '/Inventory', '/(location)/(id)/(quantity)', 'OTHER DETAILS', and 'API Instances'. The right side of the screen features a large diagram illustrating the architecture, showing 'Shopify' at the top, connected to 'Shopify API' which then connects to various components like 'Order Fulfillment API', 'Notifications', 'Orders', 'Products', 'Customers', 'Cart Sync', 'Shipping Rates', and external services like 'MySQL DB', 'NET WS', 'Salesforce', and 'Google'. Below the diagram, there are two mobile device screenshots showing the app interface, and a 'Orders' screen from the NTO developer portal.

Module 3: Apply Policy to Shopify API: Lab 1 (6/23)



Step 2: Configure an API Proxy

To create the proxy we are going to get the API Definition from **Exchange**.

1. Now let's configure an API proxy gateway for this API. Navigate to the API Manager page, click on **Manage API** and select **Manage API from Exchange**.

The screenshot shows the MuleSoft API Manager interface. The top navigation bar includes the MuleSoft logo, a search bar, and various icons. The main content area is titled "API Administration (Sandbox)". On the left, there is a sidebar with links: "Sandbox" (selected), "API Administration" (highlighted in blue), "API Groups" (with a "New" button), "Automated Policies", "Client Applications", "Custom Policies", and "Analytics". The main panel has a "Manage API" button, a search bar, and a "Filter by" dropdown. Below these are three options: "Manage API from Exchange" (which is highlighted with a red box and a red number "1." to its right), "Create new API", and "Import API from zip file". To the right, there is a message "No APIs found in this environment" next to a small icon. The bottom right corner of the slide shows the number "130".

Module 3: Apply Policy to Shopify API: Lab 1 (7/23)



Step 2: Configure an API Proxy

Configure the API with the following information:

2. Under **API name**: Type “**Shopify Experience API**”. Notice when you start to write the name, the field is auto completed:

- a. **Asset type**: Select from the drop down list RAML/OAS.
- b. **API version**: Select from the drop down list 1.0.0.
- c. **Asset version**: Select from the drop down list 1.0.1.
- d. **Managing type**: Select Endpoint with proxy.
- e. **Proxy Deployment Target**: Choose Cloudhub
- f. **Mule Version**: Check the Mule 4 option.

The screenshot shows the MuleSoft API Manager interface. On the left, there's a sidebar with a 'Sandbox' button and a back arrow labeled 'API Administration'. The main area has a title 'Manage API from Exchange' and a sub-section 'API Configurations'. A search bar contains the text '2. Shopify Experience API', which is highlighted with a red rectangle. Below the search bar are fields for 'Asset type' (set to 'RAML/OAS'), 'API version' (set to '2.0.1'), and 'Asset version' (set to '1.0.1'). Under 'Managing type', the radio button for 'Basic Endpoint' is selected. In the 'Application type' section, the radio button for 'Mule application' is selected, with a note below it stating 'Running on Hybrid, CloudHub or Runtime Fabric'. At the bottom, the 'Mule version' dropdown is set to 'Mule 4 Recommended'. The top right corner of the interface shows a navigation bar with 'DTC Workshop DS', a help icon, and a user icon.

Module 3: Apply Policy to Shopify API: Lab 1 (8/23)



Step 2: Configure an API Proxy

Configure the API with the following information:

3. Type: <http://shopify-exp-api-v1.us-e2.cloudhub.io/api> in the Section labeled **Implementation URI**:

4. Click **Advanced options** to expand to as follows.

5. Add <first name or initials>-proxy-shopify in the field, **API Instance label**.

6. And Click **Save**.

Mule version: Mule 4 Recommended Mule 3 or below

Implementation URI: `http://shopify-exp-api-v1.us-e2.cloudhub.io/api` 3.

TLS Context for outbound traffic: [Add TLS Context](#) Used to secure outbound traffic

Path: /

Advanced options 4.

Proxy version: 3.1.1

For more information on different proxy versions please visit this [page](#)

Enable Console: Enable the Console Flow in the autogenerated proxy.

Validations: Validate the inbound requests against the provided specification.

Strict validations (optional) Query params

Scheme: HTTP HTTPS

API instance label: (Optional) `<oaa>-proxy-shopify` 5.

Recommended if you have multiple managed instances of the same API

Module 3: Apply Policy to Shopify API: Lab 1 (9/23)



Step 2: Configure an API Proxy

Configure the API with the following information:

7. Set the field Runtime version to **4.4.0**.

8. In Proxy application field type:

<first name or initials>-proxy-shopify

9. Then click **Deploy**.

The screenshot shows the MuleSoft API Manager interface. On the left, a sidebar menu includes options like SANDBOX, API Administration, Alerts, Contracts, Policies, SLA Tiers, and Settings. The main content area displays the "Shopify Experience API 2.0.1" settings. Key details shown include:

- API Status: Unregistered
- Asset Version: 1.0.1
- Type: RAML/OAS
- Implementation URL: <http://shopify-exp-api-v1.us-e2.cloudhub.io/api>
- Add consumer endpoint
- API Instance: ID 17769864, Label: <oaa>-proxy-shopify
- Autodiscovery: API ID 17769864

Below this, there are tabs for Requests, Top Apps, and Latency, with a "Download CSV" button. A time range selector shows "1h" selected. The Requests tab displays "No data available".

At the bottom, there are sections for API Configuration and Deployment Configuration. The Deployment Configuration section is highlighted with red boxes and numbered steps:

- Runtime version: 4.4.0 (Step 7)
- Proxy application name: oaa-proxy-shopify (Step 8)
- Deploy button (Step 9)

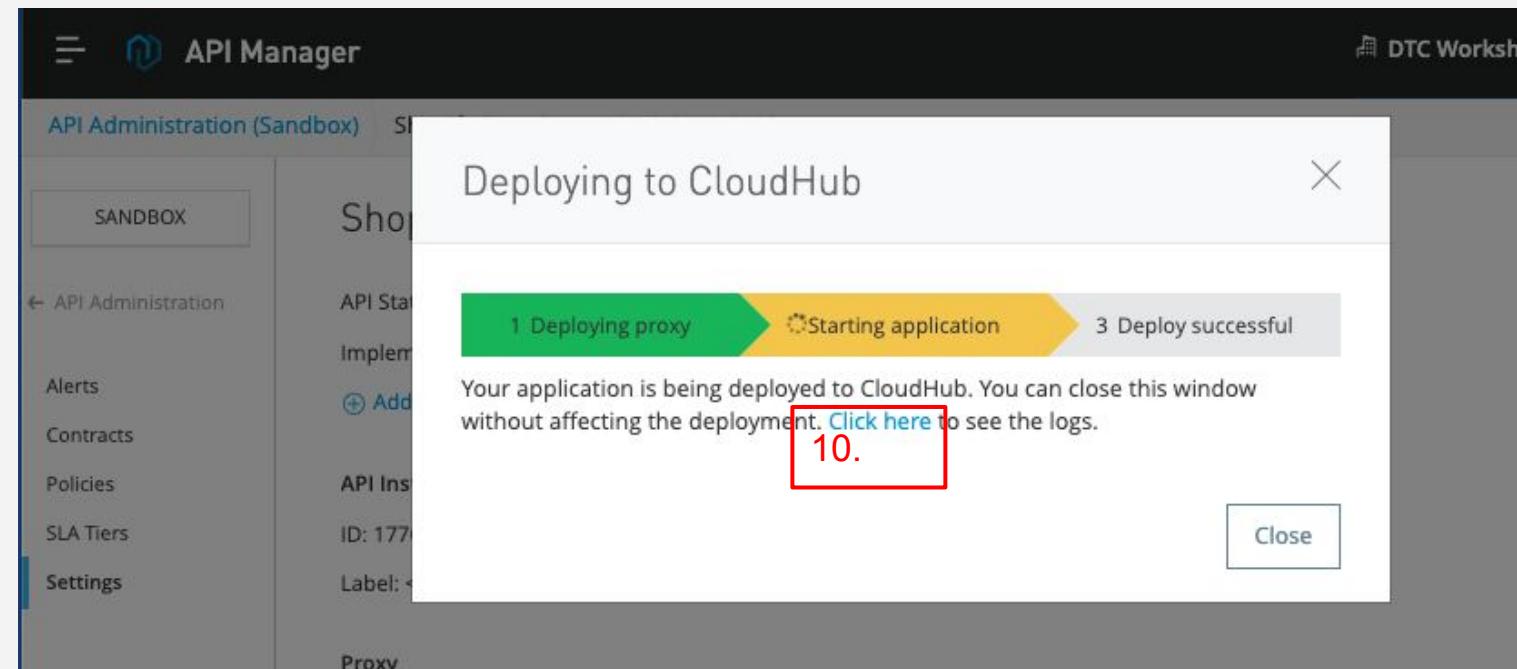
There is also a checkbox for "Update application if exists".

Module 3: Apply Policy to Shopify API: Lab 1 (10/23)



Step 2: Configure an API Proxy

10. Click on **Click Here** to see the log data and monitor the progress.



Module 3: Apply Policy to Shopify API: Lab 1 (11/23)



Step 2: Configure an API Proxy

11. A new browser tab will open that will show the CloudHub log for this application in **Runtime Manager**.

12. Wait until you see “**Your application is started**” and you can close this browser tab and go to back to the previous tab.

11.

12.

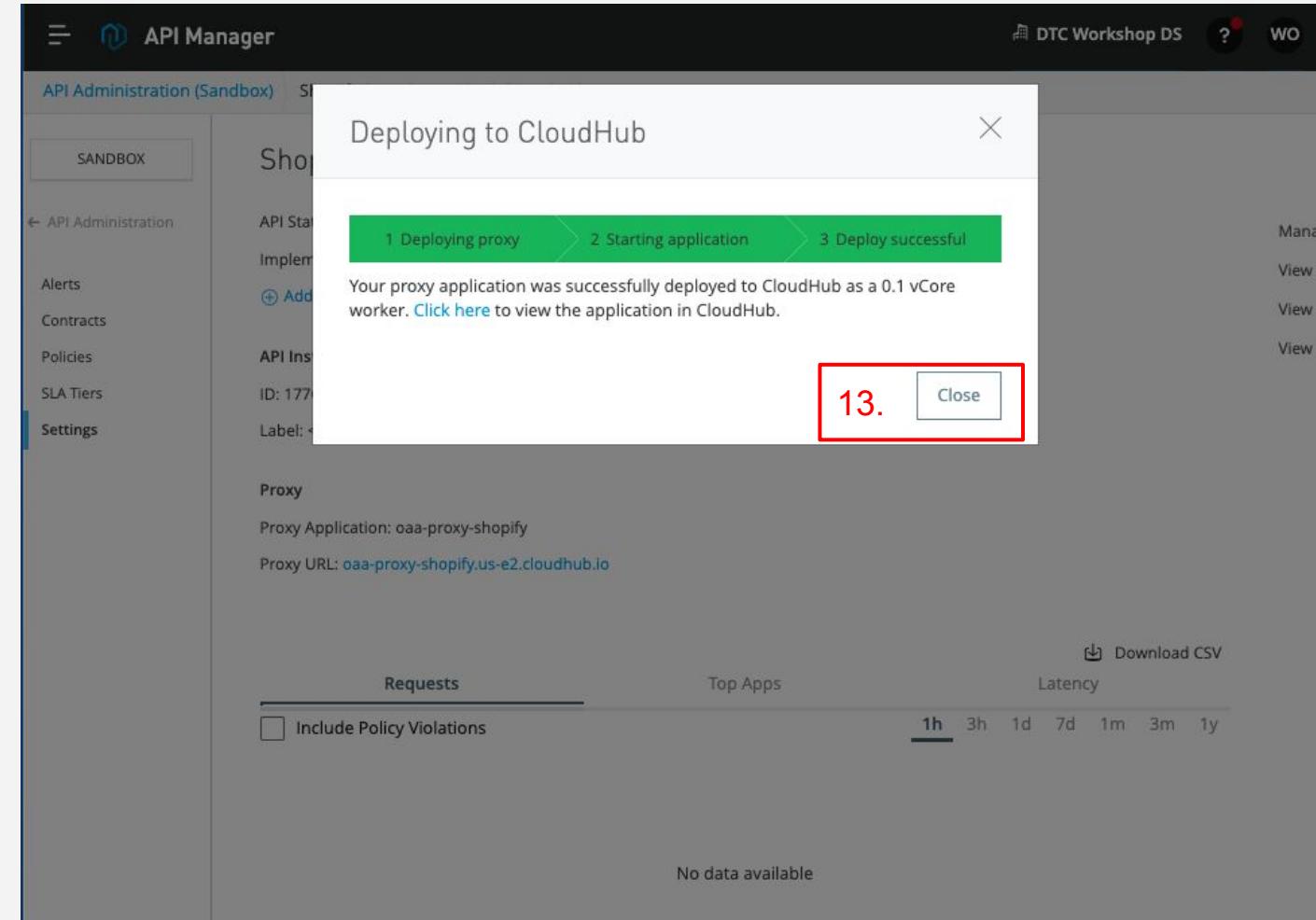
13:40:56.768 04/17/2022 Deployment system SYSTEM
Worker(18.222.21.200): Your application has started successfully.

Module 3: Apply Policy to Shopify API: Lab 1 (12/23)



Step 2: Configure an API Proxy

13. Go back to the previous browser tab where you were configuring the API. You should see the green Deploy Successful in the status bar. Click **Close**.



Module 3: Apply Policy to Shopify API: Lab 1 (13/23)



Step 2: Configure an API Proxy

14. Once it is deployed, at the top of the page you will see the status of the API. It should be **green** with a green dot next to it, as shown below. This indicates that your API was successfully deployed with its corresponding Proxy and is now being managed.

The screenshot shows the MuleSoft API Manager interface. On the left, there's a sidebar with options: SANDBOX, API Administration, Alerts, Contracts, Policies, SLA Tiers, and Settings. The Settings option is currently selected. The main content area displays the details for the "Shopify Experience API 2.0.1".
Key information shown:

- API Status:** Active (14.0.1) [highlighted with a red box]
- Asset Version:** 1.0.1
- Type:** RAML/OAS
- Implementation URL:** <http://shopify-exp-api-v1.us-e2.cloudhub.io/api>
- Mule runtime version:** 4.4.0-20220321
- API Instance:** ID: 17769864, Label: <oaa>-proxy-shopify
- Autodiscovery:** API ID: 17769864
- Proxy:** Application: oaa-proxy-shopify, URL: <http://oaa-proxy-shopify.us-e2.cloudhub.io>

At the bottom, there are tabs for Requests, Top Apps, and Latency, with the Requests tab selected. A checkbox for "Include Policy Violations" is present. The latency section shows timeframes from 1h to 1y, with 1h selected. A "Download CSV" button is also available. A note at the bottom says "No data available".

Module 3: Apply Policy to Shopify API: Lab 1 (14/23)



Step 3: Test the API Proxy

Your proxy API is now accessible via CloudHub.

1. At the Proxy URL, Right Click the URL and Select **Open Link in New Tab**. This will open a new Browser Tab.

The screenshot shows the MuleSoft API Administration interface. On the left, a sidebar has a 'Sandbox' button at the top, followed by links: API Administration, Alerts, Contracts, Policies, SLA Tiers, and Settings (which is highlighted with a blue bar). The main content area displays the following details for the 'Shopify Experience API 2.0.1':

- API Status: Active (green dot)
- Asset Version: 1.0.1 (Latest)
- Type: RAML/OAS
- Implementation URL: <http://shopify-exp-api-v1.us-e2.cloudhub.io/api>
- + Add consumer endpoint
- Mule runtime version: 4.4.0-20220321
- API Instance**: ID: 17769864, Autodiscovery: API ID: 17769864
- Label: <oaa>-proxy-shopify (with a pencil icon)
- Proxy**:
 - Proxy Application: oaa-proxy-shopify
 - Proxy URL: oaa-proxy-shopify.us-e2.cloudhub.io (boxed with a red border and labeled '1.')

Module 3: Apply Policy to Shopify API: Lab 1 (15/23)



Step 3: Test the API Proxy

Your proxy API is now accessible via CloudHub.

2. You will see a page with a error message stating:
“resource_not_found”

3. Add “**/order**” to the end of the proxy URL in the Browser and

4. **Refresh** the Page.

You should see the Shopify GET orders payload response. Good the proxy is working! Now to our next Step.

The image contains two screenshots of a web browser window. Both screenshots show a URL starting with "http://oaa-proxy-shopify.us-e2.cloudhub.io".

Screenshot 1: The URL is "http://oaa-proxy-shopify.us-e2.cloudhub.io". The response is a JSON object with a single key "error": "resource_not_found". A red box highlights this JSON object, and a red number "2." is placed to its right.

```
{"error": "resource_not_found"}
```

Screenshot 2: The URL is "http://oaa-proxy-shopify.us-e2.cloudhub.io/order". The response is a JSON object representing a Shopify Order. A red box highlights the URL, and a red number "3." is placed to its right. Another red box highlights the "4.C" button in the browser's address bar, and a red number "4." is placed to its left.

```
[{"id": 4724450984187, "admin_graphql_api_id": "gid://shopify/Order/4724450984187", "app_id": 580111, "browser_ip": "13.110.54.38", "buyer_accepts_marketing": false, "cancel_reason": null, "cancelled_at": null, "cart_token": null, "checkout_id": 32664397873403, "checkout_token": "a0836e877f17da8955c7851170fc9232", "client_details": { "accept_language": "en", "browser_height": null, "browser_ip": "13.110.54.38", "browser_width": null, "session_hash": null, }}
```

Module 3: Apply Policy to Shopify API: Lab 1 (16/23)



Step 4: Apply Policy To API

Your proxy API is now accessible via CloudHub.

1. Toggle to your previous Browser tab that has details on your Shopify API as seen:
2. Click **Policies** on the left menu pane.

The screenshot shows the API Manager interface for the Shopify Experience API (2.0.1). A red box surrounds the entire page. A red number '1.' is on the right side of the main content area, and a red number '2.' is inside the left sidebar near the 'Policies' link.

API Administration (Sandbox) Shopify Experience API (2.0.1) - Settings

SANDBOX

Shopify Experience API 2.0.1

API Status: Active Asset Version: 1.0.1 Latest Type: RAML/OAS

Implementation URL: <http://shopify-exp-api-v1.us-e2.cloudhub.io/api>

[+ Add consumer endpoint](#)

Mule runtime version: 4.4.0-20220321

API Instance Autodiscovery

ID: 17769864 API ID: 17769864

Label: <oaa>-proxy-shopify [edit](#)

Proxy

Proxy Application: oaa-proxy-shopify

Proxy URL: oaa-proxy-shopify.us-e2.cloudhub.io

Requests Top Apps

Include Policy Violations

1h 3h 1d

Module 3: Apply Policy to Shopify API: Lab 1 (17/23)



Step 4: Apply Policy To API

3. In the next Screen Click the **Apply New Policy** Button.

The screenshot shows the API Manager interface for the Shopify Experience API (2.0.1) in the Sandbox environment. The left sidebar has a 'Policies' item selected, indicated by a blue underline. The main content area displays the API details and policy sections.

API Administration (Sandbox) Shopify Experience API (2.0.1) - Policies

SANDBOX

← API Administration Alerts Contracts Policies SLA Tiers Settings

Shopify Experience API 2.0.1

API Status: Active Asset Version: 1.0.1 Latest Type: RAML/OAS
Implementation URL: <http://shopify-exp-api-v1.us-e2.cloudhub.io/api>
⊕ Add consumer endpoint
Mule runtime version: 4.4.0-20220321

Automated Policies

There are no automated policies applied for the selected runtime version.

API level policies

Apply New Policy 3.

There are no applied policies.

Module 3: Apply Policy to Shopify API: Lab 1 (18/23)



Step 4: Apply Policy To API

You will see a screen showing a list of different Policies. At the top **All Categories** you can filter the Policy listing by categories (e.g. Security, Quality of Service and so on).

Since we are trying to get NTO's Shopify Order data our requirement is to have some sort of system authentication before giving retrieving that data from Shopify.

4. Select **Basic Authentication - Simple**

5. Choose radio button **1.2.2**

6. Click **Configure Policy** Button Below.

Select Policy

All Categories All Mule Versions

Policies	Min Mule Version
> Client ID enforcement	
> Cross-Origin resource sharing	
> Detokenization	You need permission to apply this policy. Learn more
> OAuth 2.0 access token enforcement using Mule OAuth provider	
> Header Injection	
> Header Removal	
> Basic authentication - Simple	4.
<input checked="" type="radio"/> 1.2.2	5.
	4.1.0
	A 1.0

Selected: **Basic authentication - Simple 1.2.2**

Cancel **Configure Policy** 6.

Module 3: Apply Policy to Shopify API: Lab 1 (19/23)



Step 4: Apply Policy To API

7. In User Name and Password Fields type:
"workshop"

8. Click the **Apply** Button.

API Administration (Sandbox) Shopify Experience API (2.0.1) - Policies Apply Basic authentication - Simple policy

SANDBOX Policies

Apply Basic authentication - Simple policy

Enforces HTTP Basic authentication according to the details configured in the policy.

User Name *

workshop

User Password *

.....

Method & Resource conditions

Apply configurations to all API methods & resources

Apply configurations to specific methods & resources

Cancel Apply

Module 3: Apply Policy to Shopify API: Lab 1 (20/23)



Step 4: Apply Policy To API

9. You will see that the new Policy, **Basic authentication - Simple** is being enforced for our Shopify API.

It may take a several minutes before the policy takes effect, Great time to take a small break and come back.

The screenshot shows the MuleSoft API Manager interface. The left sidebar has tabs for 'Sandbox' (selected), 'API Administration', 'Alerts', 'Contracts', 'Policies' (selected), 'SLA Tiers', and 'Settings'. The main content area displays the 'Shopify Experience API 2.0.1' details: API Status: Active, Asset Version: 1.0.1 (Latest), Type: RAML/OAS, Implementation URL: <http://shopify-exp-api-v1.us-e2.cloudhub.io/api>, and Mule runtime version: 4.4.0-20220321. Below this, under 'Automated Policies', it says 'There are no automated policies applied for the selected runtime version.' Under 'API level policies', there is a blue button labeled 'Apply New Policy'. A table lists policies, with one row highlighted by a red box: 'Name' > Basic authentication - Simple (with an info icon), 'Category' Security, 'Fulfils' Requires authentication, and 'Requires' 9. The 'Requires' column is aligned with the step number from the previous text block.

Name	Category	Fulfils	Requires
> Basic authentication - Simple ⓘ	Security	Requires authentication	9.

Module 3: Apply Policy to Shopify API: Lab 1 (21/23)



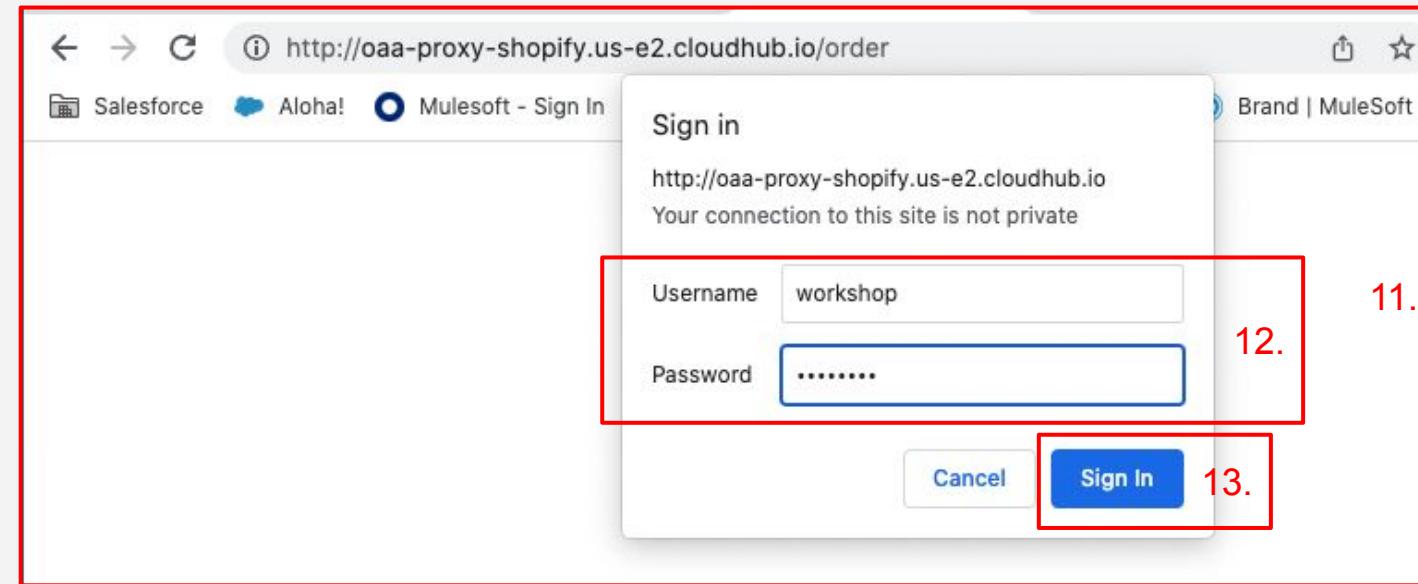
Step 4: Apply Policy To API

10. Toggle back to the Browser Tab you had for proxy URL:
`<your first name or initials>-shopify.us-e2.cloudhub.io/order`

11. Continue to **Refresh** the Browser until you see the following:

12. Enter “**workshop**” In the Username and Password Fields.

13. Click **Sign In**



Module 3: Apply Policy to Shopify API: Lab 1 (22/23)



Step 4: Apply Policy To API

14. You should see the Shopify Order Payload Response.

We did it! We just created a security policy for our Shopify API!

Plus, we have completed all Modules and Lab for this DTC Workshop! CONGRATS!!!

```
[{"id": 4724450984187, "admin_graphql_api_id": "gid://shopify/Order/4724450984187", "app_id": 580111, "browser_ip": "13.110.54.38", "buyer_accepts_marketing": false, "cancel_reason": null, "cancelled_at": null, "cart_token": null, "checkout_id": 32664397873403, "checkout_token": "a0836e877f17da8955c7851170fc9232", "client_details": { "accept_language": "en", "browser_height": null, "browser_ip": "13.110.54.38", "browser_width": null, "session_hash": null, "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.127 Safari/537.36" }, "closed_at": null, "confirmed": true, "contact_email": "muleyoscar@nto.com", "created_at": "2022-04-16T21:19:05-07:00", "currency": "USD", "current_subtotal_price": "1.00", "current_subtotal_price_set": { "shop_money": {}}
```

Module 3: Apply Policy to Shopify API: Lab 1 (23/23)



Summary

In this lab, we completed the following steps:

- Overview of Anypoint API Manager
- Configure the API proxy
- Test the API proxy using Postman



We saw how easy it is to proxy existing APIs and deploy it on a CloudHub-based API Gateway. This provides a **low friction** approach to manage existing APIs across your environment. You are able to leverage a **hybrid** API Gateway service offering which can be on-premise or on the cloud. In this lab, we saw how we can deploy our API Gateway to CloudHub which can significantly **speed up your deployment** without having to manage and maintain infrastructure.

For further reading on deploying an API gateway and proxying your API, please refer to the following documentation:

- [Proxying Your API](#)
- [Deployment Options](#)

Thank you!