



TRAILHEAD

Digital Commerce APIs

Exercise Guide

Version 1.0

TABLE OF CONTENTS

Preface	4
Overview	4
What You Will Learn	4
Prerequisites	5
Working with Industries Digital Commerce APIs	6
What are the Industries Digital Commerce APIs?	6
How do I get started?	6
The API Cache	6
Exercise 7-1: Prepare a Sales Catalog for Caching	9
What are sales catalogs?	9
Task 1: Set up the Catalog	11
Task 2: Check the Products' Status	13
Task 3: Create Catalog Product Relationships	14
Task 4: Initialize and run cache jobs	16
Exercise 7-2: Set up Postman and your Training Playground	19
What is Postman?	20
How does Postman connect to my training playground?	21
Task 1: Download and install Postman	22
Task 2: Download and install the Postman Environment	23
Task 3: Download and install the Postman Collection	25
Task 4: Configure the Connected App and Finish Configuring Postman	26
Task 5: Obtain an access token	30
Exercise 7-3: Explore the Digital Commerce APIs	34
The Phases of the Digital Commerce Experience	34
The Browse Phase APIs	34
Task 1: Execute Get Offers By Catalog	36
Task 2: Execute Get Offer Details	39
Configure Phase APIs	41
Task 3: Execute Post Offer Details with Configuration	41
Cart Phase APIs	45

Task 4: Execute Create Basket with BasketAction	46
Checkout Phase APIs	54
Task 5: Execute Create Cart From Basket	55
Exercise 7-4: Try Out Context Rules with the Digital Commerce APIs	60
Anonymous and Known User Browsing	60
Context Rules and Digital Commerce APIs	61
Task 1: Verify the context rule and make it cacheable	62
Task 2: Execute Get Offers by Catalog with the context parameter	64
Exercise 7-5: Explore Digital Commerce API Parameters	69
Digital Commerce Cacheable API Parameters	69
What are Context, Basket, and Cart Context Keys?	70
Task 1: Execute Get Offers by Catalog with the pagesize and offset parameters	72
Task 2: Execute Get Basket with the cartContextKey parameter	75
Task 3: Execute Get Basket with the includeAttachment parameter	78
Task 4: Execute Add to Basket with an invalid offer	80

Preface

These training exercises are based on the Winter '22 release of Salesforce Industries Communications, Media, and Energy & Utilities Clouds. For additional information about the topics covered in this module, see the documentation available in Salesforce Industries/Vlocity Success Community at <https://success.vlocity.com>.

Overview

This module provides practical knowledge to use the Digital Commerce cacheable APIs and caching architecture to develop digital commerce apps and sites. You will learn the basic functionalities of the APIs as well how to use their parameters. As you progress through this training, you complete practical, hands-on lab exercises designed for use with a training playground provided by Salesforce Industries.

What You Will Learn

When you complete this training, you will be able to:

- Test Digital Commerce APIs in context of the defined user experience phases
- Create a Connected App in your Training Playground
- Configure Postman to test Digital Commerce APIs
- Make Digital Commerce API calls with context rules and parameters to modify their default functionality

Prerequisites

The prerequisites for this training include a solid understanding of basic Salesforce concepts. You should also have a working knowledge of telecommunications, media, or energy and utilities industry business objectives. You should have a basic understanding of customer relationship management and configuration, pricing, and quoting. The prerequisite also includes completion of the following learning modules:

- Build CPQ Solutions for Industries
- Industries Digital Commerce Gateway Solution Overview
- Building Digital Commerce Catalogs
- Understanding Digital Commerce Application Architectures

Working with Industries Digital Commerce APIs

What are the Industries Digital Commerce APIs?

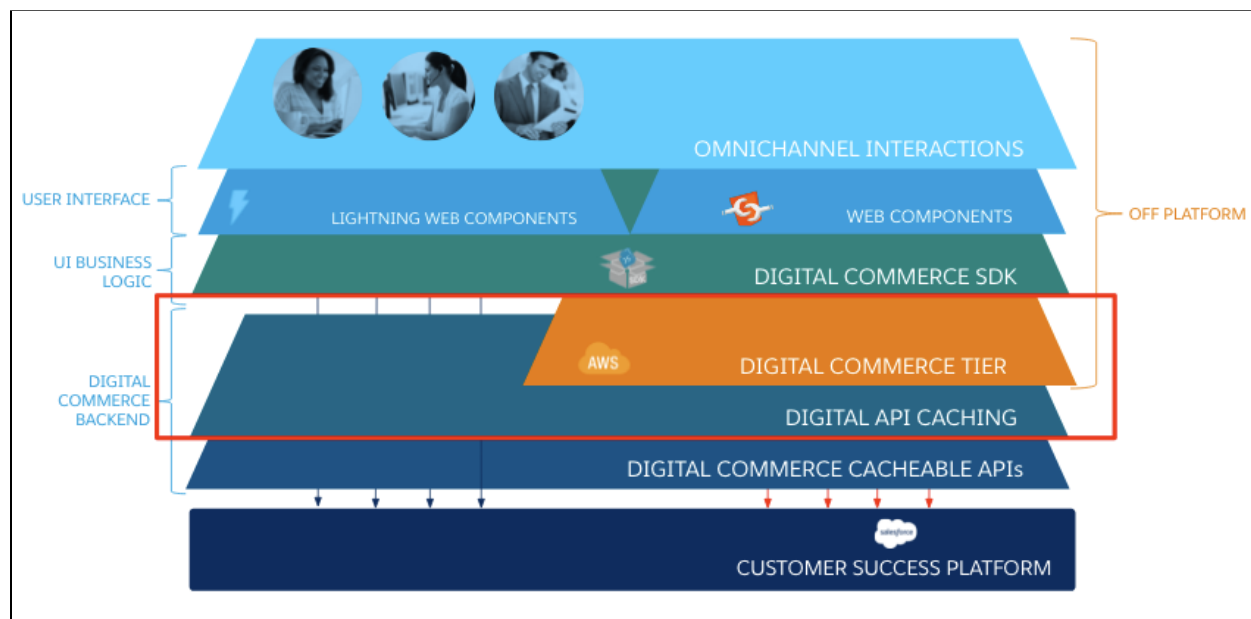
The Digital Commerce APIs extend catalog-driven product selection and configuration. These APIs support anonymous browsing and configuration. Using the Digital Commerce APIs, you can retrieve a list of Products and Promotions along with their details and configuration.

How do I get started?

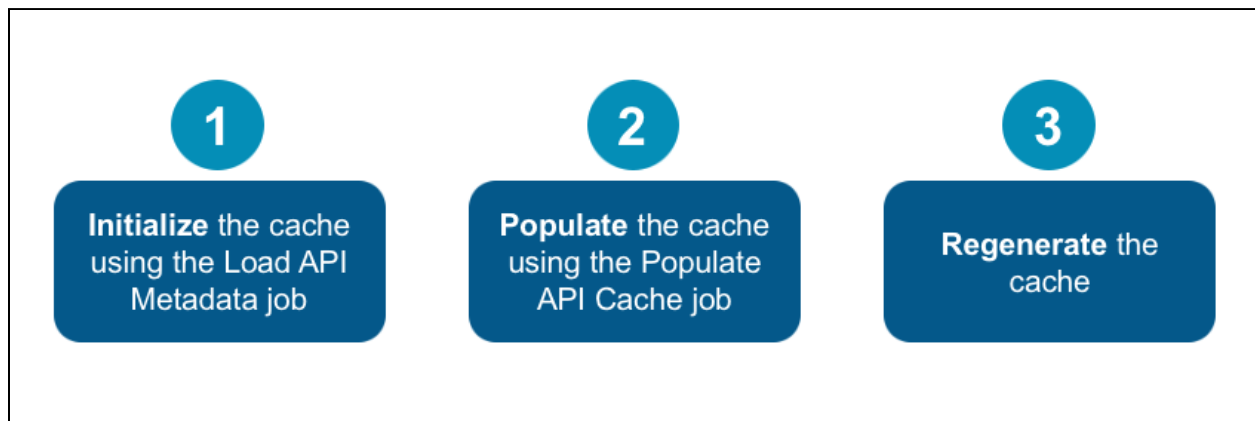
First you need to create product offerings within the product designer by specifying their name, structure, pricing, attributes, etc. Second, you should set up your sales catalog. Third, you should initiate and populate the API cache. Finally, you can use any API platform you like to test and explore the Digital Commerce APIs. We'll walk through this process together in this exercise.

The API Cache

In the digital commerce world, a large number of interactions are anonymous or nearly anonymous. The early stage of browsing catalogs and products lends itself to caching since many users receive the same responses. Therefore, intelligent caching strategies and cacheable APIs are critical in any enterprise-scale digital commerce solution.



Consequently, the Industries Digital Commerce solution includes a [caching mechanism](#). When requests are made by users, Digital Commerce first refers to the API Cache Response object before making a new request. This cache stores Digital Commerce API responses inside Salesforce for on-platform solutions. In addition, the optional Digital Commerce Gateway extends this [caching layer to AWS](#) for off-platform solutions. This caching design results in high-performance response times and supports high-volume anonymous and nearly anonymous browsing traffic that is common on consumer-facing websites.



Catalog administrators have a couple cache management tasks that they must do to initially populate the cache and then they can regenerate the cache when there are changes to the Product Catalog.



RESOURCE:

Find more information on Cache Management within the Understanding Digital Commerce APIs topic in the Digital Commerce APIs course. Please read the [documentation](#) as well. Additionally, please see the topic Working with the Cache Management APIs within the Working with Digital Commerce APIs course.

1. **Initialize the cache using the Load API MetaData job.** This job is run when first setting up Digital Commerce. It initializes the cacheable API Metadata sObject, which is a custom object that stores the metadata required by Digital Commerce APIs.

2. **Populate the cache using the Populate API Cache job.** This job populates the cache with products, product bundles, and promotions defined in the sales catalogs. In addition, context rules combinations, prices, offer details, and other data is also pre-populated and stored to ensure fast response times.
3. **Regenerate the cache.** When changes are made to any of the catalogs, products or promotions used in digital commerce, the cache must be regenerated.

Exercise 7-1: Prepare a Sales Catalog for Caching

Scenario

Eliza is ready to dive straight into the Digital Commerce APIs. Before Eliza can start exploring the APIs, she will need to have some cached data available for the APIs. Eliza decides to use a small sales catalog to test the APIs with.

Goal

- Validate an existing sales catalog is ready to be used with the Digital Commerce APIs
- Initialize and Populate the Cache for a sales catalog

Tasks

1. Set up the Catalog
2. Check the Products' Status
3. Create Catalog Product Relationships
4. Initialize and run cache jobs

Time: 20 mins



ALERT:

If you've just received your training playground, add your email address to the system administrator profile to ensure you receive all system notifications. In the upper-right, click on the **Avatar** and select **Settings**. Enter your email address in the **Email** field on the Personal Information page and click **Save**.


What are sales catalogs?

[Sales catalogs](#) are a subset of products/promotions from the Shared Catalog. Sales catalogs are ideal for Digital Commerce guided selling apps because they can be curated for specific product marketing strategies. In order to start building a guided selling experience, you will need to first create an ecommerce sales catalog.

Sales catalogs are made up of **Catalog Product Relationships**, which define products and promotions included in the shared catalog.

For this exercise, you will [create a catalog](#) and associate some products with it.

Task 1: Set up the Catalog

1. Using the Lightning App Launcher  , search for **Digital Commerce**, and then select it.
2. In the Lightning navigation bar, click **Catalogs**.
3. Click **New** in the top-right corner.
4. In the **New Catalog** dialog, enter the following information.

Field	Value	Notes
Catalog Name	Mobile Catalog	
Catalog Code	DC_MOBILE	Catalog codes are used by DC Web Components, SDK, and APIs in order to retrieve and display products to the UI. Salesforce Industries recommends using a DC_prefix to easily identify DC catalogs. This value is not required to be unique but we recommend it as a best practice.
Is Active	✓	This field must be set to active as it is enforced.
Start Date Time	[yesterday's date]	The Start Date Time is enforced. If you select today's date, it will default to 12 PM which is midday. For that reason, you will select yesterday's date to ensure the catalog will be active right now.
Description	Digital Commerce Catalog for Mobile Phones	
Default Price List	B2C Price List	All of the products in a catalog should be on the same price list. This price list will be used to look up the price list entry of the products that are included in the catalog. This field is mandatory.



NOTE:

The **Vendor**, **Is Catalog Root**, and **Source Type** fields are not currently used by the cacheable APIs and will not be reflected in your guided selling experience. The **Vendor** field can lookup an Account record for the sales catalog for reference purposes and has no impact on catalog products.

5. Click **Save**.

New Catalog

Information

* Catalog Name

Mobile Catalog

Is Catalog Root

☐

Source Type ⓘ

--None--

Catalog Code

DC_MOBILE

Global Key

Vendor

Search Accounts... 🔍

Is Default Catalog

☐

Is Active

☒

Start Date Time

Date

9/27/2019 📅

Time

12:00 PM ⌚

End Date Time

Date

Time

Description

Digital Commerce Catalog for Mobile Phones

Owner

Admin Vlocity VSO v104 DC

Default Price List

B2C Price List ✕

Cancel





Save & New

Save

Task 2: Check the Products' Status

Eliza's first priority is getting the new phones up on her new digital commerce guided selling experience. Before she adds them to her new digital commerce catalog, she needs to make sure that they are marked as active and have price entries in the shared catalog.

In this exercise, the products are already entered into the shared catalog for you.

1. In the Lightning navigation bar, click **Vlocity Product Console**.
2. In the Dashboard under Product Management, click the **search icon**  next to **Product**.
3. Click the **search icon** .
4. Ensure the following products and promotions are present and marked as **Active** in your training playground:
 - a. **Apple iPhone 13**
 - b. **Samsung Galaxy S10**
5. Click the **Dashboard** tab next to the **Search Product** tab.
6. In the Dashboard under Product Management, click the **search icon**  next to **Promotion**.
7. Click the **search icon** .
8. Ensure the following products and promotions are present and marked as **Active** in your training playground:
 - a. **iPhone 13 High Pri Customer Promo**



ALERT:

Products must be marked **active**, **orderable**, and have a valid **selling start/end date** otherwise they will not be returned in the API response and will not be cached. We recommend setting the **Product Specification Type** as **Offer** for all sellable products.

Click on each product/promotion and check the **Pricing** facet to ensure it has at least one price entry defined for the **B2C Price List**.

Task 3: Create Catalog Product Relationships

1. In the Lightning navigation bar, click **Catalogs**.
2. Click **Mobile Catalog**.
3. Click the **Related** tab header.


4. Next to Catalog Product Relationships (0), click the **New** button .

5. In the **New Catalog Product Relationship** dialog, enter the following information.


Field	Value	Notes
Catalog Product Relationship Name	Apple iPhone 13	
Catalog	Mobile Catalog	
Product	Apple iPhone 13	
Effective Date	[yesterday's date]	The Effective Date is enforced. You will select yesterday's date to ensure the catalog relationship will be active right now.
Is Active	✓	
Sequence	10	The Sequence Number field can be used to manipulate which offers show first in the offer list.


6. Click **Save**.
7. Repeat the previous two steps for Samsung Galaxy S10 and iPhone 13 High Pri Customer Promo except you will select iPhone 13 High Pri Customer Promo in the **Promotion** field instead of the **Product** field. Additionally, increment the **Sequence** number by 10 each time.

Eliza's first sales catalog is now created and has offers associated with it.

 **Catalog**
Mobile Catalog

Related Details

 **Catalog Relationships (0)** [New](#)

 **Catalog Product Relationships (3)** [New](#)

Catalog Product Relationship Name	Product	Sequence Number	
Apple iPhone 13	Apple iPhone 13	10	▼
Samsung Galaxy S10	Samsung Galaxy S10	20	▼
iPhone 13 High Pri Customer Promo		30	▼


[View All](#)



ALERT:

This catalog's products have not yet been cached and are therefore not yet available to the Digital Commerce cacheable APIs. You will learn how to populate the Digital Commerce cache in the next task.

Task 4: Initialize and run cache jobs

1. In the Lightning navigation bar, click **Vlocity CMT Administration**.
2. Below Admin Console, select **Cacheable API Jobs**.
3. Click the **Start** button  next to **LOAD API METADATA**. This is considered initializing the cache. You only need to complete this step after a first install of Industries Communications, Media, Energy, and Utilities, or after an upgrade if instructed. On the other hand, you will run the POPULATE API CACHE job to inorganically creates cache records for anonymous user browsing use cases, more specifically to generate cached responses for GetOffers and GetOfferDetails API calls.



RESOURCE:

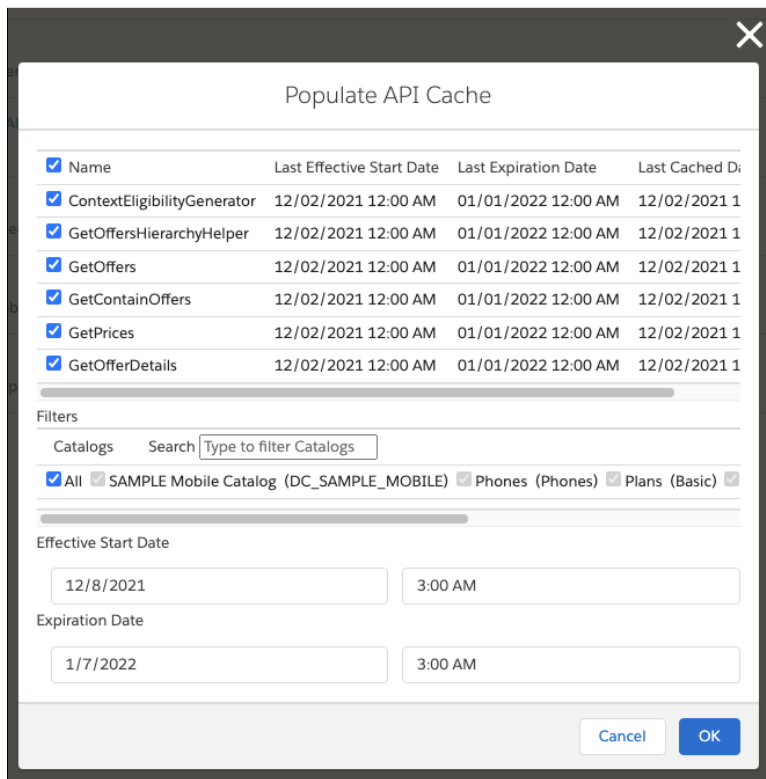
Read more about Cache Management in the [documentation](#).

4. Click the **Start** button  next to **POPULATE API CACHE**. This is considered populating the cache.

This process will populate the cache for all of the products in all of your active sales catalogs. It will create a “customer shape” or cached record for every valid result derived from any cached context rules that are applicable to your sales catalog products.

5. Select the **checkbox** at the top, which will then automatically select all of the items in the list.

6. Select the **checkbox** next to All so that the cache is populated for all of the sales catalogs in your training playground.



The dialog box titled "Populate API Cache" contains a table of API endpoints, a filters section, and date/time input fields. The table lists endpoints like ContextEligibilityGenerator, GetOffersHierarchyHelper, GetOffers, GetContainOffers, GetPrices, and GetOfferDetails, all with their last effective start and expiration dates. The filters section includes a search bar and checkboxes for "All", "SAMPLE Mobile Catalog (DC_SAMPLE_MOBILE)", "Phones (Phones)", and "Plans (Basic)". The Effective Start Date is set to 12/8/2021 3:00 AM, and the Expiration Date is set to 1/7/2022 3:00 AM. At the bottom are "Cancel" and "OK" buttons.

<input checked="" type="checkbox"/> Name	Last Effective Start Date	Last Expiration Date	Last Cached Date
<input checked="" type="checkbox"/> ContextEligibilityGenerator	12/02/2021 12:00 AM	01/01/2022 12:00 AM	12/02/2021 1
<input checked="" type="checkbox"/> GetOffersHierarchyHelper	12/02/2021 12:00 AM	01/01/2022 12:00 AM	12/02/2021 1
<input checked="" type="checkbox"/> GetOffers	12/02/2021 12:00 AM	01/01/2022 12:00 AM	12/02/2021 1
<input checked="" type="checkbox"/> GetContainOffers	12/02/2021 12:00 AM	01/01/2022 12:00 AM	12/02/2021 1
<input checked="" type="checkbox"/> GetPrices	12/02/2021 12:00 AM	01/01/2022 12:00 AM	12/02/2021 1
<input checked="" type="checkbox"/> GetOfferDetails	12/02/2021 12:00 AM	01/01/2022 12:00 AM	12/02/2021 1


Filters

Catalogs Search

☒ All ☐ SAMPLE Mobile Catalog (DC_SAMPLE_MOBILE) ☐ Phones (Phones) ☒ Plans (Basic) ☒

Effective Start Date

Expiration Date

7. Select the **OK** button . It will take a few minutes for all of the cache jobs to finish running.

POPULATE API CACHE

This will process existing data and populate the API cache.

Job Name	Status	Total Batches	Batches Processed	Created Date	Completion Date
GetOfferDetails	Completed	21	21	7/30/20 11:03 AM	7/30/20 11:03 AM
GetPrices	Completed	2	2	7/30/20 11:03 AM	7/30/20 11:03 AM
GetContainOffer	Completed	5	5	7/30/20 11:02 AM	7/30/20 11:03 AM
GetOffers	Completed	5	5	7/30/20 11:02 AM	7/30/20 11:02 AM
GetOffersItem	Completed	5	5	7/30/20 11:02 AM	7/30/20 11:02 AM
GetOffersHierarchy Helper	Completed	22	22	7/30/20 11:01 AM	7/30/20 11:02 AM
ContextEligibilityG enerator	Completed	5	5	7/30/20 11:01 AM	7/30/20 11:01 AM
RuleSetCombinati onGenerator	Completed	2	2	7/30/20 11:01 AM	7/30/20 11:01 AM
EcomDataProfileG enerator	Completed	5	5	7/30/20 11:01 AM	7/30/20 11:01 AM



ALERT:

Any time you make changes to your sales catalog products (e.g., modifying a price entry, adding a context rule to a product), you must regenerate the cache. You will regenerate the cache in Exercise 7-4.

Exercise 7-2: Set up Postman and your Training Playground

Scenario

Eliza is ready to explore the APIs but has one last step to complete, she needs a tool to test the APIs. Eliza has heard a lot of good things about Postman so she decides to try using it. Finally, she also needs an authorization token so that she can use Postman with her training playground.

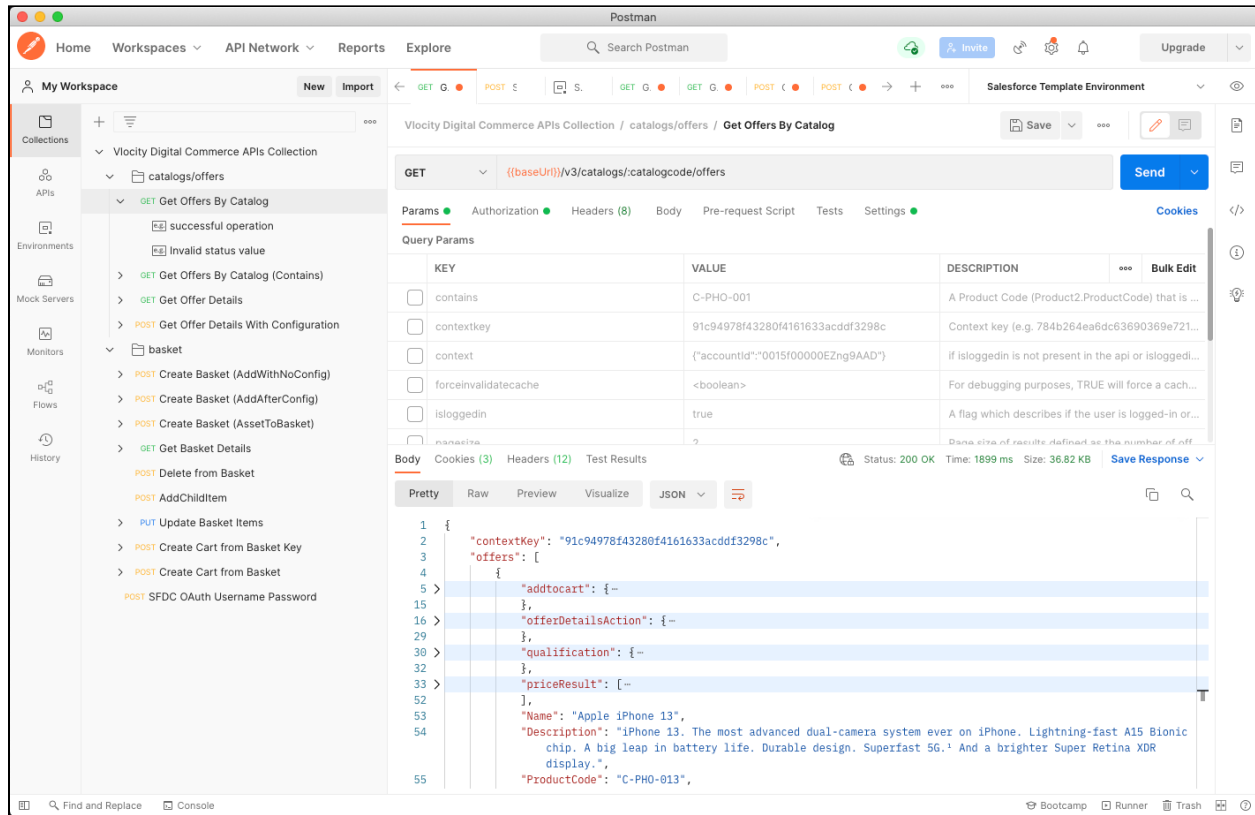
Goal

- Download and configure the Postman Environment for using the Digital Commerce APIs
- Create and configure a connected app within your training playground

Tasks

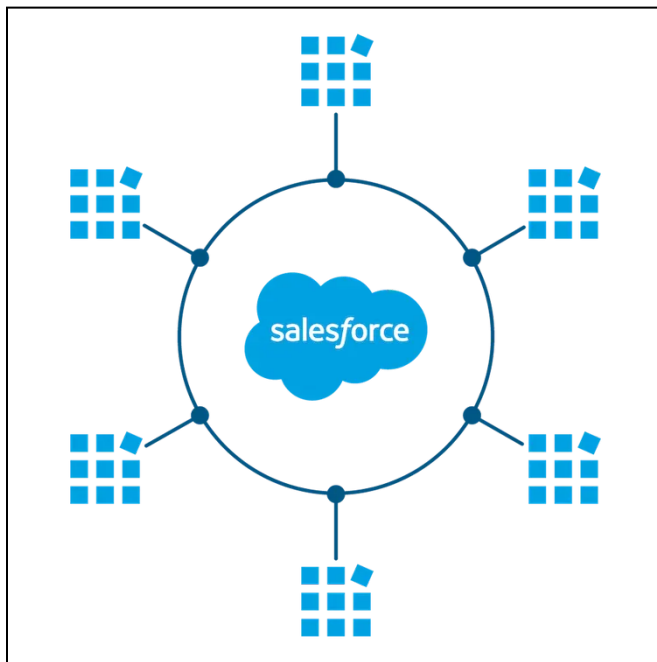
1. Download and install Postman
2. Download and install the Postman Environment
3. Download and install the Postman Collection
4. Configure the Connected App
5. Configure the Postman Environment
6. Obtain a session token

Time: 25 mins



What is Postman?

Postman is a third-party application that can be used to test APIs by sending requests and examining the responses. Postman allows you to step through each API in the Browse, Configure, Cart, and Checkout phases. It allows you to specify parameters and path variables as well so you can **quickly** compare different types of responses. Furthermore, product organizations can bundle and distribute APIs and environments. For these reasons, we will use it for these exercises.



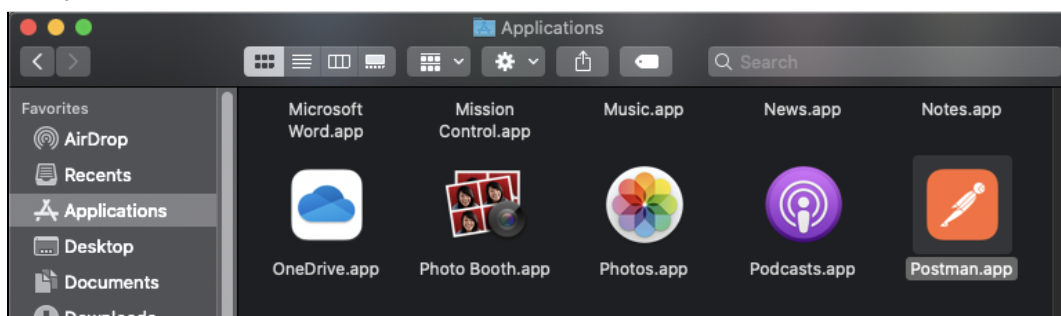
How does Postman connect to my training playground?

A [Connected App](#) is a Salesforce framework that enables an external application to integrate with Salesforce using APIs and standard protocols. Connected apps use various protocols including OAuth to authenticate external apps.

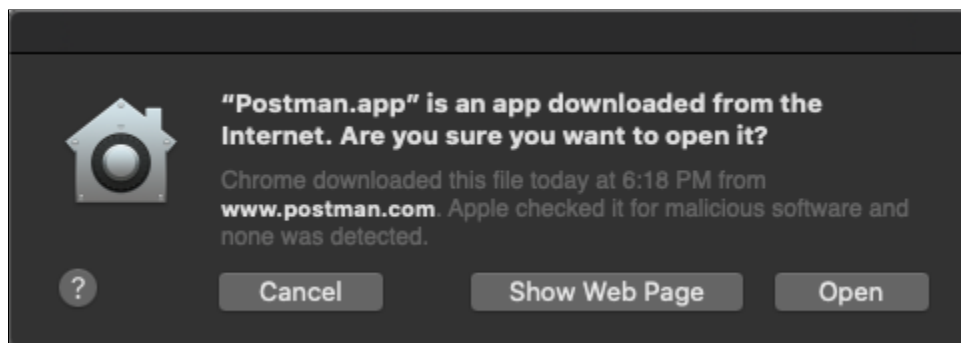
To call the Digital Commerce APIs from Postman, you must first deploy a connected app in your org. The connected app handles security (OAuth in this case) and permits access to org data from external APIs.

Task 1: Download and install Postman

1. Download **Postman** from <https://www.postman.com/downloads/>
2. Move the **Postman** app into your Applications folder or equivalent space on your computer if needed.



3. Open **Postman**.
4. Click **Open** if you receive any warnings from your operating system.



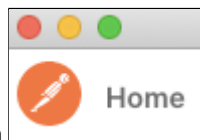
Postman should now be open.

Task 2: Download and install the Postman Environment

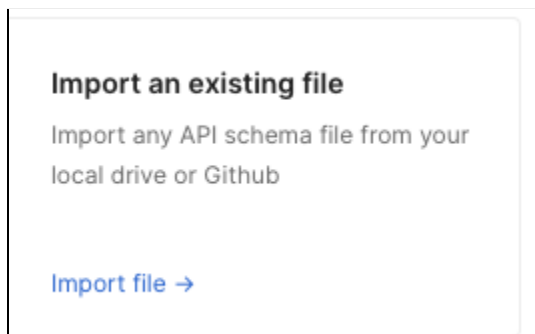
Next we'll set up our Postman **Environment**. The **Environment** contains all the details (e.g., login credentials, tokens, URLs) needed for Postman to send API requests to your training playground.

Follow the steps below to download and update a Postman configuration file that contains the necessary variables for you to run the APIs:

1. Use this [link](#) to download the **Environment** file.
2. Start Postman.



3. Click the **Home** button at the top-left corner of the app.
4. Click **Import file** -> below the **Import an existing file** heading.



5. Click **Upload Files** and select the environment file you downloaded in step 1.

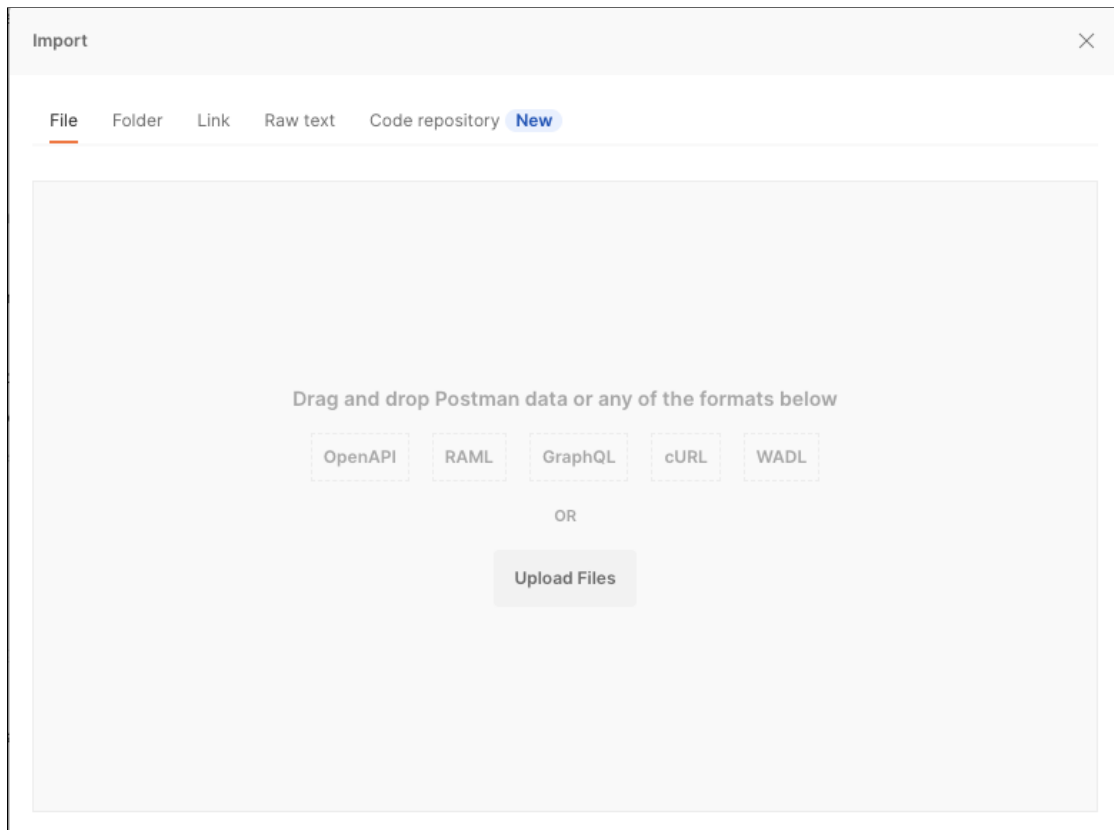
6. Configure your **Environment** with the following values:

VARIABLE	CURRENT VALUE	Notes
url	<code>https://login.salesforce.com</code>	Make sure there are no extra spaces at the beginning and ending each value if you are copying and pasting from this guide. This is the URL necessary for authentication.
baseUrl	<p>Your org domain name (e.g., analytics-value-17213) with this at the end:</p> <p><code>.my.salesforce.com/services/apexrest/vlocity_cmt</code></p> <p>For example:</p> <p><code>https://analytics-value-17213.my.salesforce.com/services/apexrest/vlocity_cmt</code></p>	<p>This is the base path needed for the Digital Commerce APIs.</p> <p>Do not use the lightning.force.com base URL, only my.salesforce.com.</p>
username	Your org username	
password	Your org password.	Avoid using the "&" character in the password; Postman does not process it correctly.

We will retrieve and configure the Connected app **clientId** and **clientSecret** values in Task 4.


Task 3: Download and install the Postman Collection

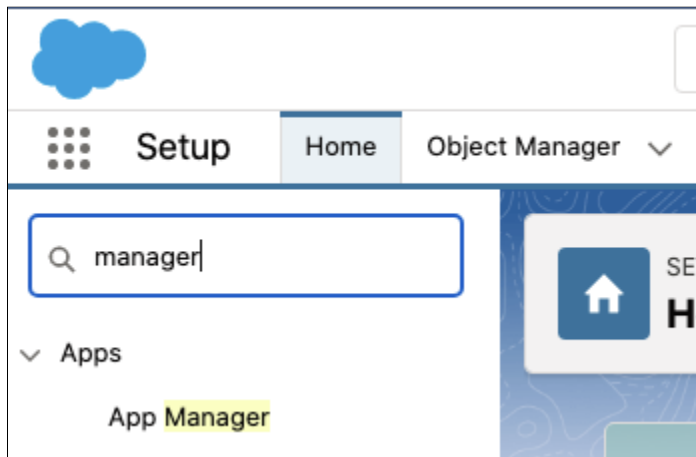
1. The file is a Postman **Collection** that contains variables and configuration settings to make calling the Digital Commerce APIs quickly and easily.
2. Use the following link to download the Postman **Collection** file:
<https://volt.my.salesforce.com/sfc/p/o00000000IKm8/a/3m00000005umv/X8mm38e.7nCeHE52342fSYgZVFni9ANWfr0Q2iJJjd8>
3. Start **Postman** (if necessary) and click **File** at the top of your screen, followed by **Import....**
4. Drag and drop the Postman collection you downloaded in step 1 into the window.



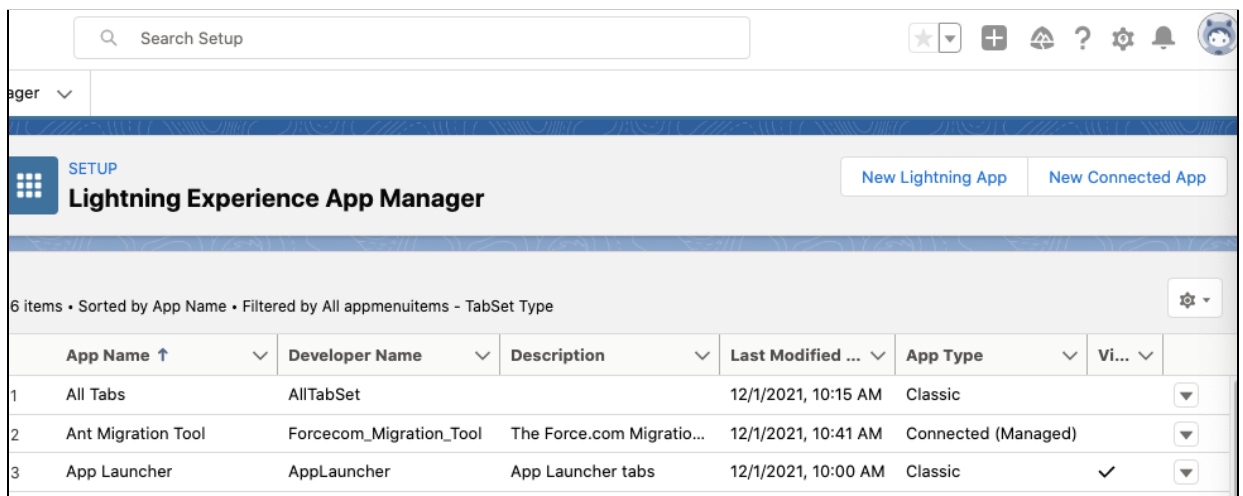
5. Click the orange **Import** button .

Task 4: Configure the Connected App and Finish Configuring Postman

1. Log into your training playground.
2. Click the **gear**  in the top-right corner and then click **Setup**.
3. In Setup, in **Quick Find**, type `manager`, then click **App Manager**.



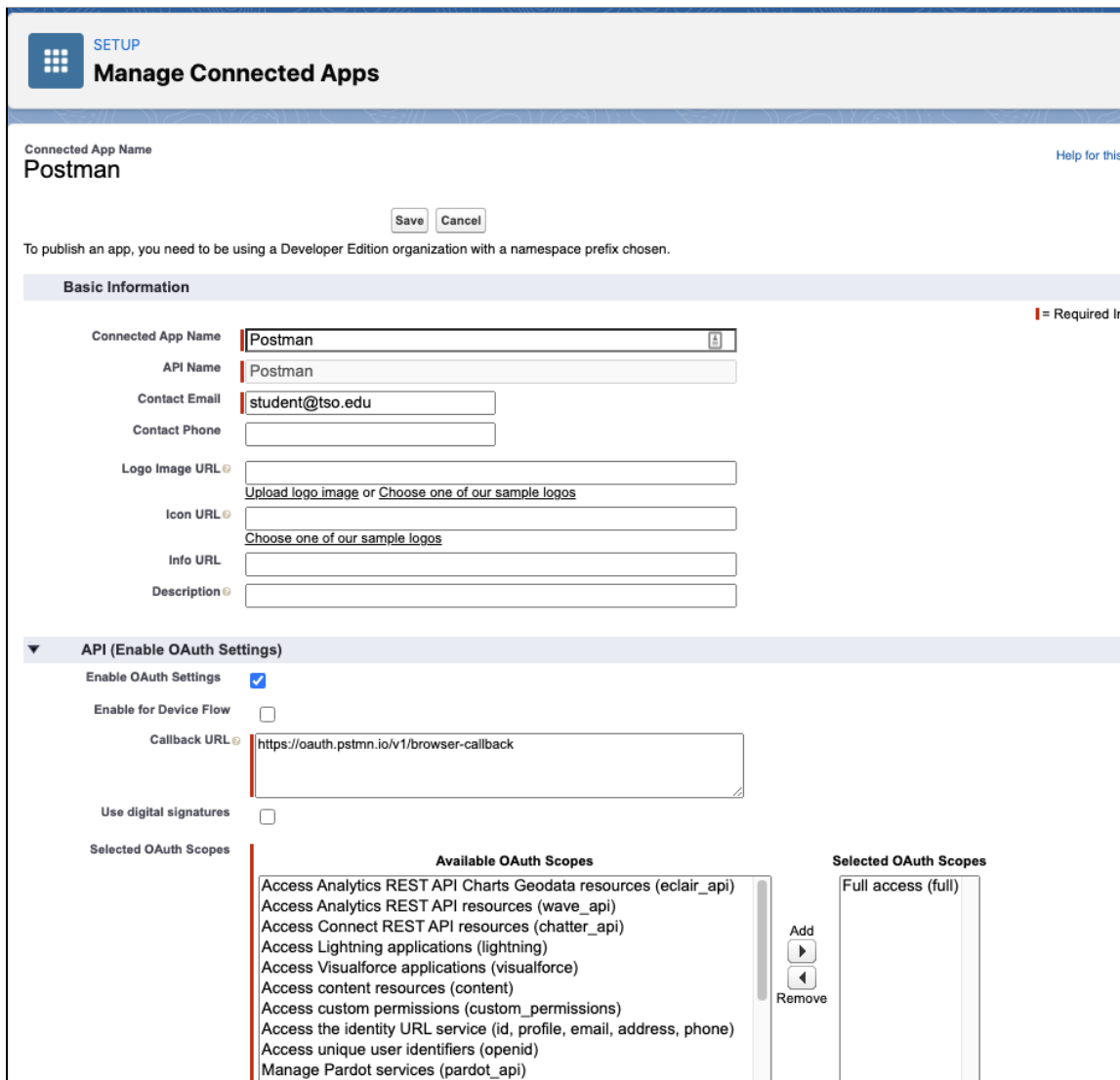
4. Click **New Connected App** on the right side of the page.



5. Fill in the following fields:

Field	Value	Notes
Connected App Name	Postman	This value will automatically be copied into the API Name field. Any time you go into App Manager, you can find this app's settings there listed as "Postman".
Contact Email	[Your Email]	
Enable OAuth Settings	Checked	We will be authenticating using OAuth.
Callback URL	<code>https://oauth.postman.io/v1/browser-callback</code>	This is provided by Postman here . This is where Postman redirects after it has successfully authenticated with your training playground.
Selected OAuth Scopes	Full Access	Because we are using a training playground, we can allow Full Access, for the purposes of this exercise. For production, please select an appropriate setting by following the guidance in the documentation: Enable OAuth Settings for API Integration .

Your page should resemble the following image:




SETUP
Manage Connected Apps

Connected App Name
Postman [Help for this](#)

To publish an app, you need to be using a Developer Edition organization with a namespace prefix chosen.

Basic Information

Connected App Name 

API Name

Contact Email

Contact Phone

Logo Image URL
[Upload logo image](#) or [Choose one of our sample logos](#)

Icon URL
[Choose one of our sample logos](#)

Info URL

Description

API (Enable OAuth Settings)

Enable OAuth Settings ☒

Enable for Device Flow ☐

Callback URL

Use digital signatures ☐

Selected OAuth Scopes

Available OAuth Scopes

- Access Analytics REST API Charts Geodata resources (eclair_api)
- Access Analytics REST API resources (wave_api)
- Access Connect REST API resources (chatter_api)
- Access Lightning applications (lightning)
- Access Visualforce applications (visualforce)
- Access content resources (content)
- Access custom permissions (custom_permissions)
- Access the identity URL service (id, profile, email, address, phone)
- Access unique user identifiers (openid)
- Manage Pardot services (pardot_api)

Selected OAuth Scopes

- Full access (full)

6. Click **Save**.

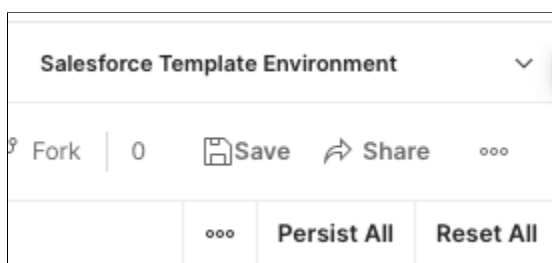
The following message displays:

Changes can take up to 10 minutes to take effect. Deleting a parent org also deletes all connected apps with OAuth settings enabled.

7. Open **Postman**, reopen your **Salesforce Template Environment** within **Environments** if it's not already open.

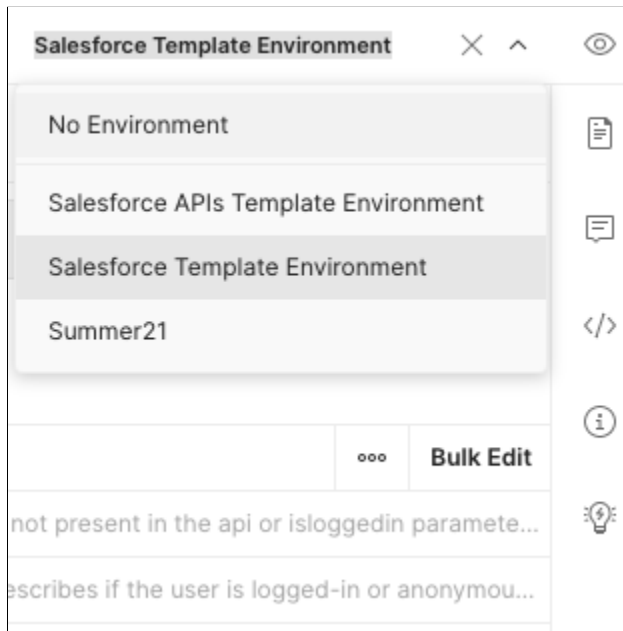
8. From the training playground open in your browser, next to **Consumer Key**, click **Copy**, and then in **Postman**, paste the value into the **clientId** CURRENT VALUE field in the environment file you imported in Task 2.
9. From the training playground open in your browser, next to **Consumer Secret**, click the **Click to reveal** link, click **Copy**, and then in **Postman**, paste the value into the **clientSecret** CURRENT VALUE field in the environment file you imported in Task 2.

10. Click the **Save** button  in the top-right corner of the screen.

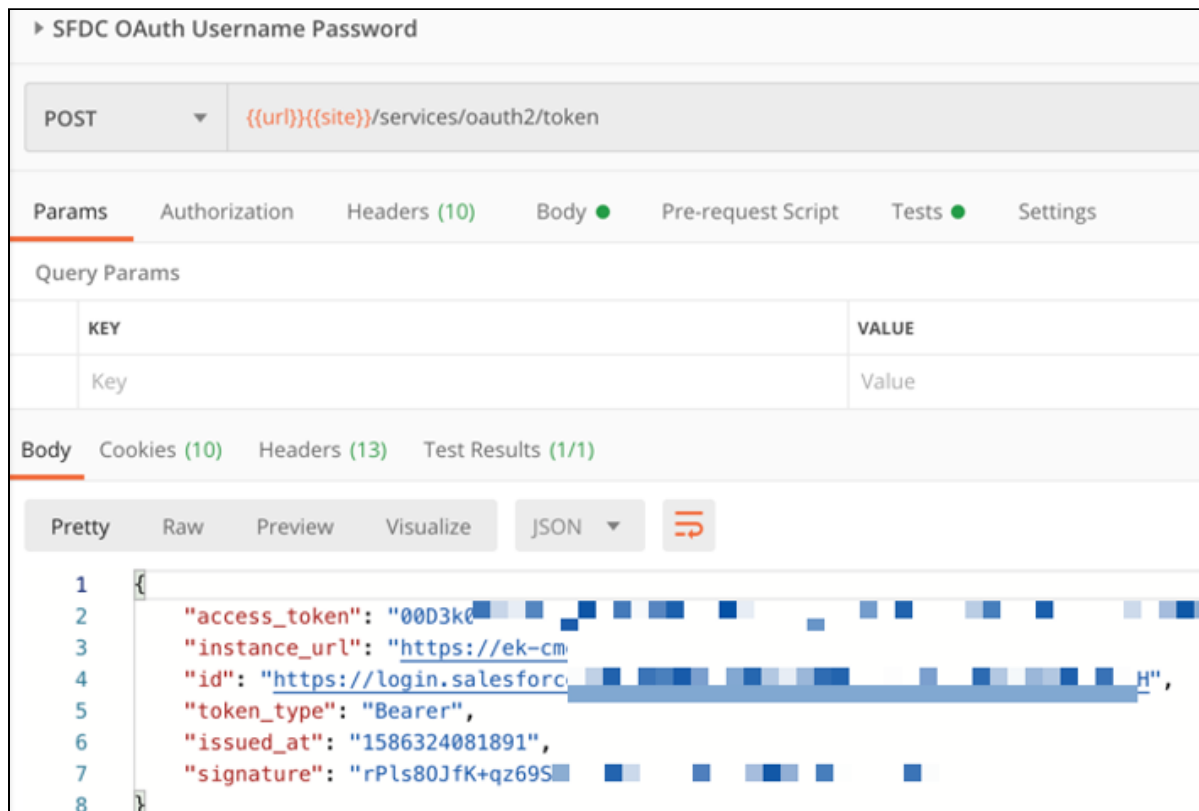


Task 5: Obtain an access token

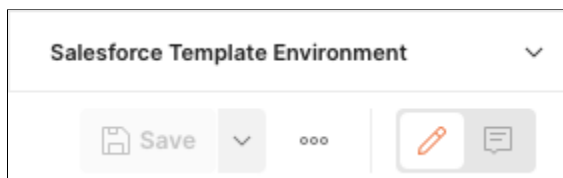
1. Click **Collections**, expand **Vlocity Digital Commerce APIs Collection**, and click **SFDC OAuth Username Password**.
2. Change **No Environment** to **Salesforce Template Environment**.



3. Click **Send**. Your screen should resemble the image below:



If your response did not succeed, you will not receive an `access_token`. Double check that your **Environment** settings are correct, and that your **Environment** is selected in the top-right corner of Postman:




NOTE:

If you created your Connected App less than ten minutes ago, you may need to wait a few minutes before trying to authenticate again.

ALERT:

This authentication is temporary. For example, if you stop using the APIs and come back after 30 minutes, you may receive an `INVALID_SESSION_ID` `errorCode` when attempting any API calls.

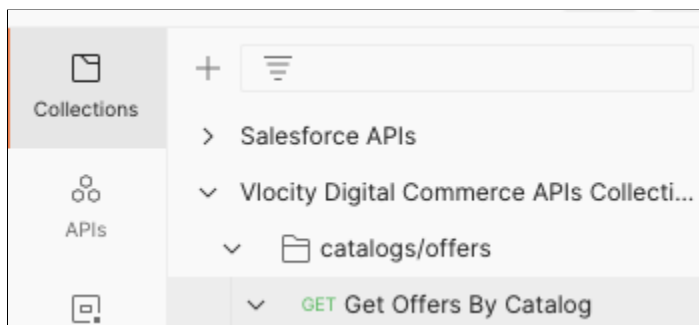


```
Body  Cookies (3)  Headers (9)  Test Results
Pretty  Raw  Preview  Visualize  JSON  ↺
1  [
2    {
3      "message": "Session expired or invalid",
4      "errorCode": "INVALID_SESSION_ID"
5    }
6  ]
```

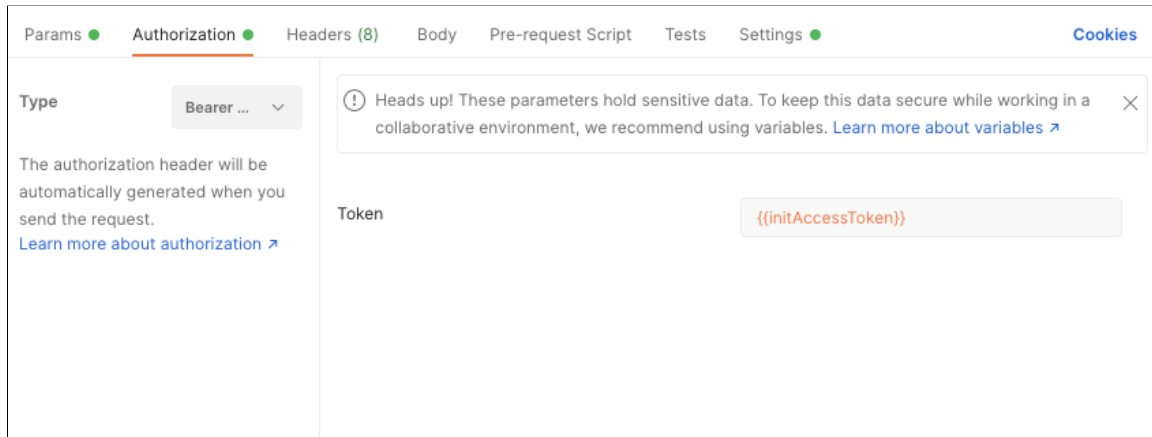
You must run this SFDC OAuth Username Password again to reauthenticate.

4. Click on the **Collections** button  on the left-side.

5. Expand **catalogs/offers** and then click **Get Offers By Catalog**.



- Click the **Authorization** header below the URL address bar. Notice that it has the value of `{{initAccessToken}}` in the **Token** field. This variable was generated during our previous authentication and stored in our Salesforce Template Environment. This gives us the flexibility to switch between environments (Salesforce orgs) without having to manually update each API with its access token.



The screenshot shows the 'Authorization' tab of a REST client interface. The 'Type' is set to 'Bearer ...'. A warning message states: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)'. The 'Token' field contains the value `{{initAccessToken}}`. On the left, a note says: 'The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)'.

Exercise 7-3: Explore the Digital Commerce APIs

Scenario

The time is here! Eliza is finally going to explore the Digital Commerce APIs. She decides that going through a buy flow with the APIs will teach her all about the inputs and outputs possible with the APIs. She decides that she will purchase an iPhone as an anonymous user through the Browse, Configure, and Basket phases, and then go through the Checkout phase using one of the Accounts in her playground.

Goal

- Test each of the available Digital Commerce APIs with Postman in order to create and submit an order
- Simulate the backend API calls for a buy flow scenario

Tasks

1. Execute Get Offers By Catalog
2. Execute Get Offers Details
3. Execute Post Offer Details with Configuration
4. Execute Create Basket with BasketAction
5. Execute Create Cart From Basket

Time: 25 mins

The Phases of the Digital Commerce Experience

There are four phases that the Digital Commerce experience is divided into. Each of these phases represents where the end user is in the buy flow. As developers, we also use these phases to categorize and conceptualize the Digital Commerce APIs. In chronological order, the phases are Browse, Configure, Cart, and Checkout. In this exercise, we will explore the Digital Commerce APIs in this order.

The Browse Phase APIs

The **Get Offers by Catalog API** provides the ability to retrieve products and promotions that are modeled within the Vlocity Shared Catalog.

The specified catalog code identifies the catalog that contains offers (products and promotions). You can optionally specify a page size to limit the number of results returned. The sort order of the result is based on the sequence field or another specified field. You can search either by name or description.

The **Get Offer Details API** retrieves detailed information about a specific offer (either product or promotion). The API returns the full hierarchy of a product or promotion as defined in the product catalog.

The API response is created from the pre-cached results present in the sObject (current caching layer). The API also handles cache misses. Along with returning the offer detail, the API caches the data for future requests. Prices, adjustments, and overrides are returned along with offers. The API does not use the Salesforce ID as part of the request and response parameters.



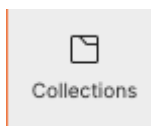
QUESTION:

Not getting the API response you expect? As you begin to use these APIs, refer to our [API Troubleshooting page](#) when you run into issues.

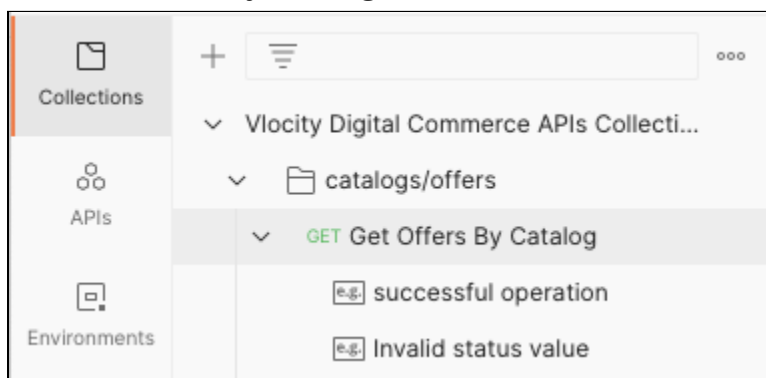
Task 1: Execute Get Offers By Catalog

Recall the catalog we created in the first exercise? We are going to use the catalog code to execute this API.

1. Open **Postman** if it's not open already.



2. Select the **Collections** button on the left side of the window.
3. Click **Get Offers By Catalog**.



4. Below **Path Variables**, click the **VALUE** box for **catalogcode**.
5. Type `DC_MOBILE`.

Path Variables				
	KEY	VALUE	DESCRIPTION	...
	catalogcode	DC_MOBILE	Identifies the catalog of interest to a custom...	Bulk Edit

6. Click the **Send** button



BEST PRACTICE:

Collapse/uncollapse all arrays/objects in the JSON within the Postman Body response with the following keyboard shortcuts.

Collapse:

Alt + 0 (Windows)

Command + Option + 0 (Mac)

Uncollapse:

Alt + Shift + 0 (Windows)

Command + Option + Shift + 0 (Mac)

7. Examine the response.

Path Variables

KEY	VALUE	DESCRIPTION	...	Bulk Edit
catalogcode	DC_MOBILE	Identifies the catalog of interest to a custo...		

Body Cookies (3) Headers (12) Test Results 200 OK 1462 ms 22.85 KB Save Response

Pretty Raw Preview Visualize JSON

```

1  {
2    "contextKey": "122357d20eed46c1f199521e062238bb",
3    "offers": [
4      {
5        "addtocart": {
6          "rest": {
7            "params": {
8              "basketAction": "AddWithNoConfig",
9              "offer": "C-PHO-013"
10             },
11            "link": "v3/catalogs/DC_MOBILE/basket",
12            "method": "POST"
13          }
14        },
15        "offerDetailsAction": {
16          "rest": {
17            "params": {},
18            "method": "GET",
19            "link": "v3/catalogs/DC_MOBILE/offers/C-PHO-013"
20          },
21          "remote": {
22            "params": {}
23          },
24          "client": {
25            "records": [],
26            "params": {}
27          }
28        },
29        "priceResult": [
30          {
31            "adjustments": [],
32            "overrides": [],
33            "chargeamount": 799.99000,
34            "baseamount": 799.99000,
35            "Amount__c": 799.99000,
36            "IsVirtualPrice__c": false,
37            "IsBasePrice__c": true,
38            "effectiveuntildatespec": null,
39            "effectivefromdatespec": null,
40            "SubType__c": "Standard",
41            "Type__c": "Price",
42            "RecurringFrequency__c": null,
43            "ChargeType__c": "One-time",
44            "DisplayText__c": "Starting at $799.99",
45            "pleEffectiveFrom": "2021-11-30T05:00:00.000Z",
46            "UnitPrice": 799.99000
47          }
48        ],
49        "Name": "Apple iPhone 13",

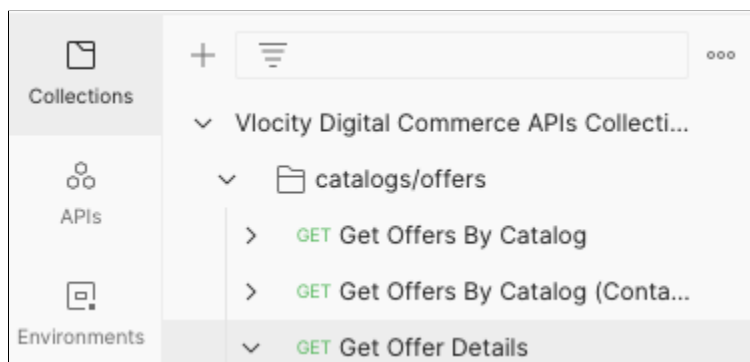
```

Notice that we are receiving all of the offers in the catalog that we created earlier.

Not only are we getting some of the offer details like pricing, we are getting **addtocart** and **offerDetailsAction** nodes which offer some options for subsequent API calls we make with these products. By default you are also getting **Attachments** (e.g., URLs to product images). With this information rich JSON, we can render an attractive product offers page. If you want to see a Digital Commerce LWC example, please see the Guided Selling course in Build Configure, Price, Quote (CPQ) Solutions for Industries.

Task 2: Execute Get Offer Details

1. Click **Get Offer Details**.



2. Below **Path Variables**, click the **VALUE** box for **catalogcode**.
3. Type `DC_MOBILE`.
4. Below **Path Variables**, click the **VALUE** box for **offercode**.
5. Type `C-PHO-013`.

Path Variables		
	KEY	VALUE
	catalogcode	DC_SAMPLE_MOBILE
	offercode	C-PHO-013

6. Click the **Send** button .

7. Examine the response.

```
Body Cookies (3) Headers (12) Test Results 200 OK 1820 ms 10.53 KB
Pretty Raw Preview Visualize JSON
1  {
2    "contextKey": "122357d20eed46c1f199521e062238bb",
3    "result": {
4      "offerDetails": {
5        "offer": {
6          "priceResult": [
7            {
8              "adjustments": [],
9              "overrides": [],
10             "chargeamount": 799.99000,
11             "baseamount": 799.99000,
12             "Amount__c": 799.99000,
13             "IsVirtualPrice__c": false,
14             "IsBasePrice__c": true,
15             "effectiveuntildatespec": null,
16             "effectivefromdatespec": null,
17             "SubType__c": "Standard",
18             "Type__c": "Price",
19             "RecurringFrequency__c": null,
20             "ChargeType__c": "One-time",
21             "DisplayText__c": "Starting at $799.99",
22             "pleEffectiveFrom": "2021-11-30T05:00:00.000Z",
23             "UnitPrice": 799.99000
24           }
25         ],
26         "addtocart": {
27           "rest": {
28             "params": {
29               "context": "{\"DIM_PRIORITY\":\"null\",\"Region\":\"None\"}",
30               "basketAction": "AddWithNoConfig",
31               "offer": "C-PHO-013"
32             },
33             "link": "v3/catalogs/DC_SAMPLE_MOBILE/basket",
```

In this response, we receive pricing, configuration, and other attribute data. Similar to the Get Offers response, we also receive **addtocart** and **offerDetailsAction** nodes which offer some options for subsequent API calls we make with this product.

You may notice within the **addtocart** node, it has a **context** with two context dimensions and their corresponding default values. Because these two context dimensions are designated as cacheable and because we did not specify them with a context parameter, they are giving us their default values. We will explore the **context** parameter later in this guide.

Configure Phase APIs


The **Post Offer Details with Configuration API** provides the ability to configure an offer (either a product or promotion) and view updated prices, quantities, and attributes. Returns the configured product details.

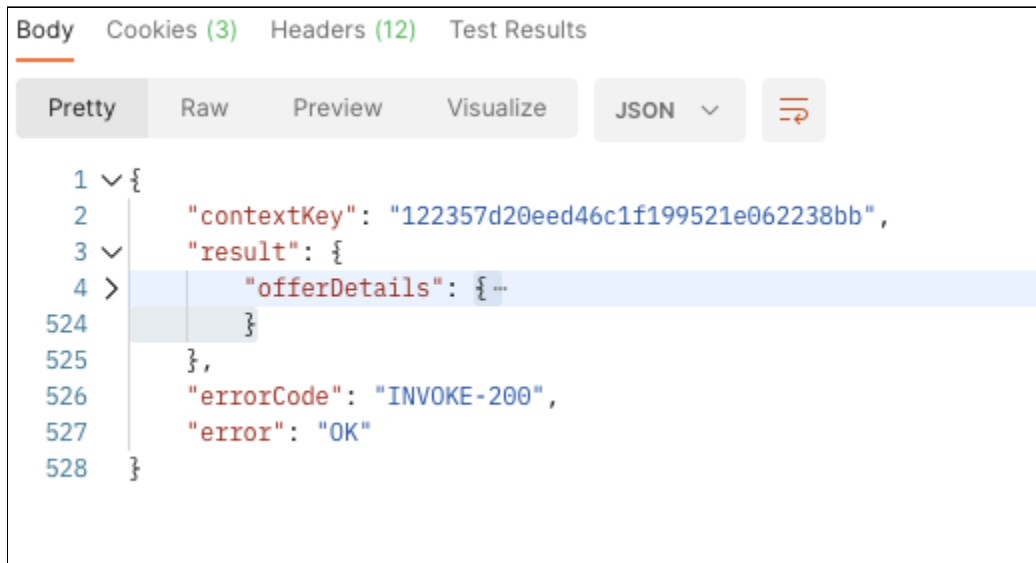
The configured product details are cached so that subsequent requests for the same configuration are retrieved with the data in the current cache layer.

Validations, prices, adjustments, overrides, rules, context and so on are resolved internally. The Salesforce ID is not used as part of the request or response parameters.

If validation fails, the OfferDetails structure is returned with an appropriate error message.

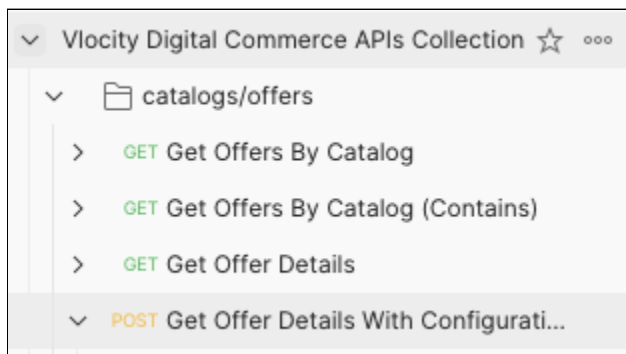
Task 3: Execute Post Offer Details with Configuration

1. Let's make a minor configuration to the iPhone 13 response we received in the last task. Within the **Body** window of the **Get Offer Details** response, highlight and copy all of the contents of the **offerDetails** node. You can do this easily by collapsing the node by click  on the line number.

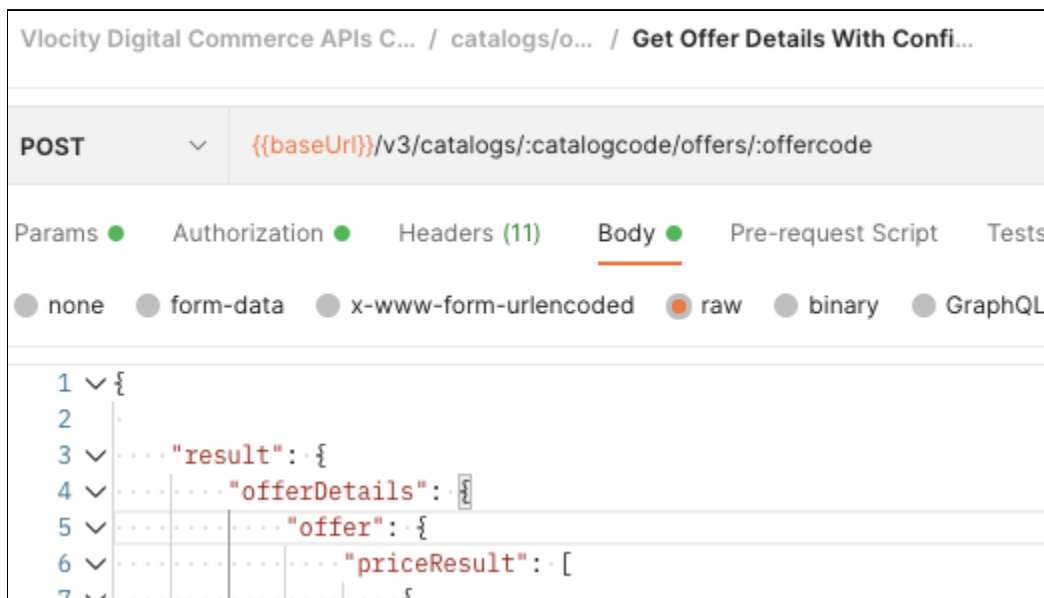


```
Body Cookies (3) Headers (12) Test Results
Pretty Raw Preview Visualize JSON
1 {
2   "contextKey": "122357d20eed46c1f199521e062238bb",
3   "result": {
4     "offerDetails": {
524   }
525 },
526 "errorCode": "INVOKE-200",
527 "error": "OK"
528 }
```

2. Click **Get Offer Details With Configuration**.



3. Click the **Body** header below the URL bar.
4. Within the **offerDetails** bracket, paste the response you copied earlier. Carefully check to ensure you do not have duplicate brackets or “offerDetails” within the **offerDetails** brackets.



5. Within the **offer** node, change the **Quantity** to 2. This is a very simple configuration change we are making to the product configuration JSON we received from the

previous API.

```
2 | -
3 | ✓ ..... "result": {
4 | ✓ ..... "offerDetails": {
5 | ✓ ..... "offer": {
6 | > ..... "priceResult": [ ...
25 | ..... ],
26 | > ..... "addtocart": { ...
36 | ..... },
37 | ..... "Name": "Apple iPhone 13",
38 | ..... "ProductCode": "C-PHO-013",
39 | ..... "Product2Id": "01t5f000000vqTpAAI",
40 | ..... "PricebookEntryId": "01u5f000000yUEyAAM",
41 | ..... "Quantity": 2.0,
```

6. Click the **Params** header below the URL bar.
7. Below **Path Variables**, click the **VALUE** box for **catalogcode**.
8. Type DC_MOBILE.
9. Below **Path Variables**, click the **VALUE** box for **offercode**.
10. Type C-PHO-013.

11. Click the **Send** button .

12. Examine the response.

```
"contextKey": "3ec21192a46c1198576d6bea7aef357b",
"result": {
  "offerDetails": {
    "offer": {
      "addtocart": {
        "rest": {
          "method": "POST",
          "link": "v3/catalogs/DC_MOBILE/basket",
          "params": {
            "offer": "C-PHO-013",
            "basketAction": "AddWithNoConfig",
            "context": "{ \"DIM_PRIORITY\": \"null\", \"Region\": \"None\" }"
          }
        }
      }
    },
    "priceResult": [
      {
        "adjustments": [],
        "overrides": [],
        "chargeamount": 799.99000,
        "baseamount": 799.99000,
        "Amount__c": 799.99000,
        "IsVirtualPrice__c": false,
        "IsBasePrice__c": true,
        "effectiveuntildatespec": null,
        "effectivefromdatespec": null,
        "SubType__c": "Standard",
        "Type__c": "PRICE",
        "RecurringFrequency__c": null,
        "ChargeType__c": "One-time",
        "DisplayText__c": "Starting at $799.99",
        "pleEffectiveFrom": "2021-11-30T05:00:00.000Z",
        "UnitPrice": 799.99000,
        "OneTimeTotal__c": 1599.98,
        "OneTimeCharge__c": 799.99000
      }
    ]
  }
}
```

Notice that the **OneTimeTotal_c** is 1599.98 which means we are successfully getting the pricing data for two iPhone 13 products. You can further confirm this with **Quantity** still showing as **2.0** in the response.

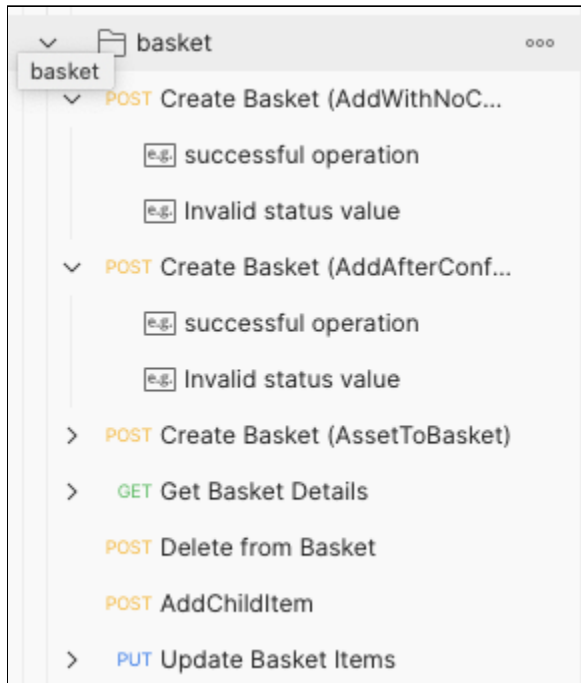
Cart Phase APIs

The **Create Basket with BasketAction API** provides the ability to create a basket from items that were added by a user during the configure operation.

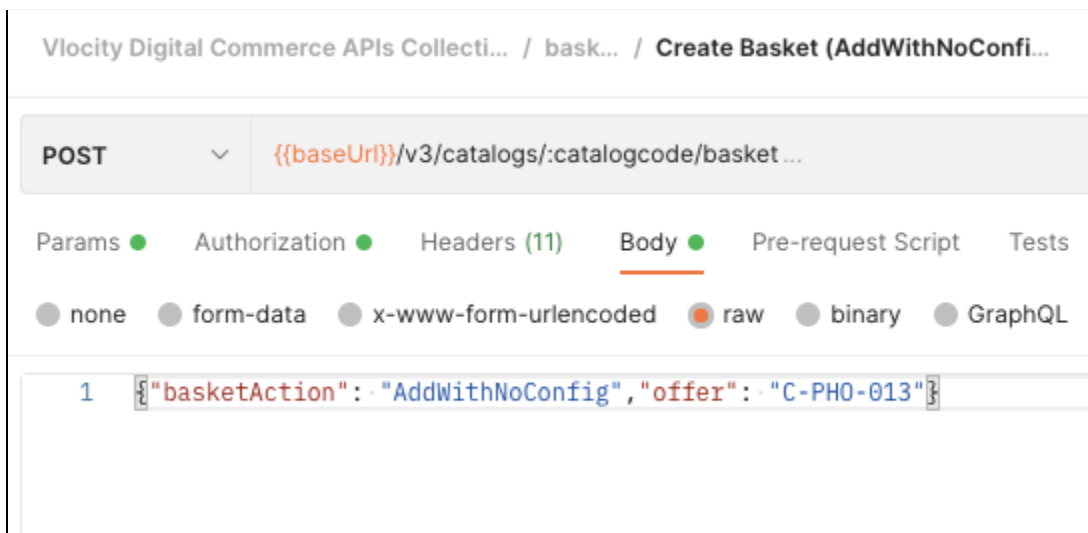
Use this API to create a basket in the following ways depending on the format of the request body: `AddWithNoConfig`, `AddAfterConfig`, and `AssetToBasket`. The `AssetToBasket` method is used for handling a customer account's assets (e.g., Cable Internet Service) using the Digital Commerce APIs. The other two methods handle adding products to a basket either with no configuration or after configuration. For the former, if the product has configurable attributes that are required and do not have a default value, they will give you an error message. We will explore this in this task.

Task 4: Execute Create Basket with BasketAction

1. Expand the **basket** folder on the left-side of **Postman**.



2. Click **Create Basket (AddWithNoConfig)**. This method will allow you to create a basket containing a product with its default configuration.



3. Below **Path Variables**, click the **VALUE** box for **catalogcode**.

4. Type `DC_MOBILE`.
5. In the URL bar, delete `/:cartContextKey`. Since we do not have an existing basket to add a product to, we do not have a `cartContextKey`. We will remove this optional variable for this reason. Your URL should look like this:

POST

`{{baseUrl}}/v3/catalogs/:catalogcode/basket`

6. Click the **Body** header below the URL bar.
7. Next to **offer**, replace **ProductCode** with `C-PHO-013`. Your Body should look like

this:

Params Authorization Headers (11) Body Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary GraphQL

1

`{"basketAction": "AddWithNoConfig", "offer": "C-PHO-013"}`

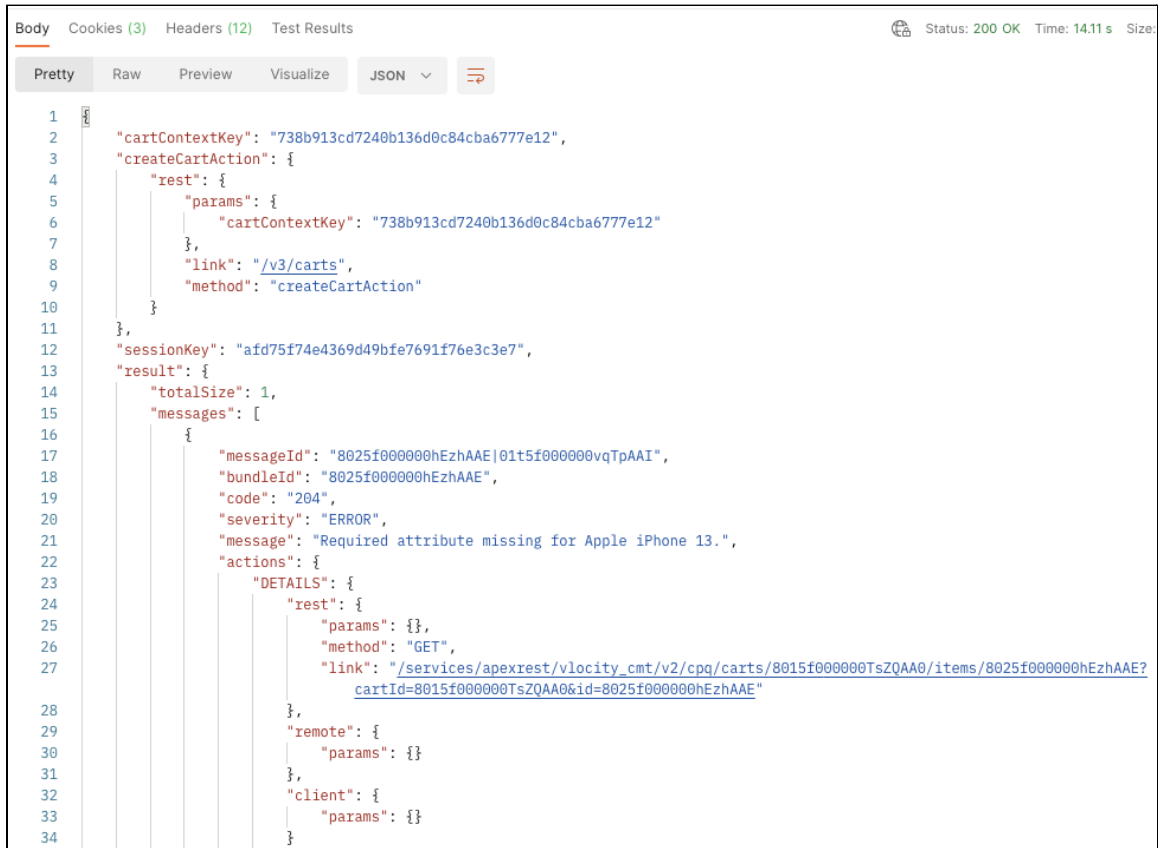
8. Click the **Send** button

A blue rectangular button with the word "Send" in white text and a small downward-pointing chevron icon to its right.

© Copyright 2022 Salesforce.com, inc. All rights reserved.

47

9. Examine the response.




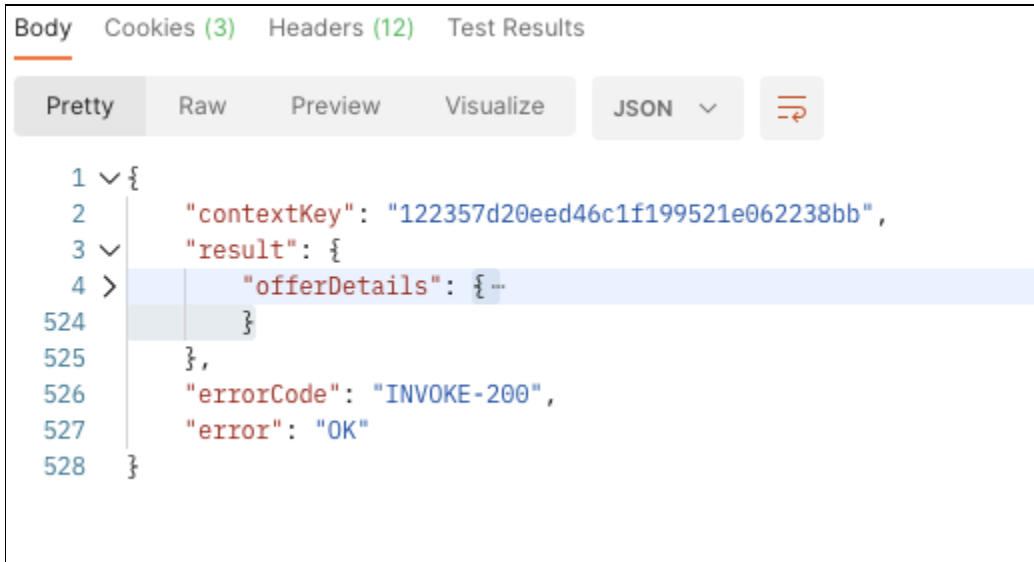
```
1  {
2    "cartContextKey": "738b913cd7240b136d0c84cba6777e12",
3    "createCartAction": {
4      "rest": {
5        "params": {
6          "cartContextKey": "738b913cd7240b136d0c84cba6777e12"
7        },
8        "link": "/v3/carts",
9        "method": "createCartAction"
10     },
11   },
12   "sessionKey": "afd75f74e4369d49bfe7691f76e3c3e7",
13   "result": {
14     "totalSize": 1,
15     "messages": [
16       {
17         "messageId": "8025f00000hEzhAAE|01t5f00000vqTpAAI",
18         "bundleId": "8025f00000hEzhAAE",
19         "code": "204",
20         "severity": "ERROR",
21         "message": "Required attribute missing for Apple iPhone 13.",
22         "actions": {
23           "DETAILS": {
24             "rest": {
25               "params": {},
26               "method": "GET",
27               "link": "/services/apexrest/vlocity_cmt/v2/cpq/carts/8015f00000TsZQAA0/items/8025f00000hEzhAAE?cartId=8015f00000TsZQAA0&id=8025f00000hEzhAAE"
28             },
29             "remote": {
30               "params": {}
31             },
32             "client": {
33               "params": {}
34             }
35           }
36         }
37       }
38     ]
39   }
40 }
```

Notice that we have a **cartContextKey**. We can use this parameter to retrieve this cached basket using the various basket APIs. The **createCartAction** shows us the format for which we could turn this basket into an Order.

Notice that while we have added the iPhone 13 to the basket, it is giving us the **message** “Required attribute missing for Apple iPhone 13.” If we scroll down to **“code”: “ATT_RT_CLR”**, we can see that is **“required”: true**. This means we must have a color selected for the phone before it can be purchased. Let’s do that now.

10. Click **Get Offers Details With Configuration**.

11. Within the **Body** window of the **Get Offer Details With Configuration** response, highlight and copy all of the contents of the **offerDetails** node. You can do this easily by collapsing the node by click  on the line number.



```
Body Cookies (3) Headers (12) Test Results
Pretty Raw Preview Visualize JSON
1 {
2   "contextKey": "122357d20eed46c1f199521e062238bb",
3   "result": {
4     "offerDetails": {
524   },
525 },
526 "errorCode": "INVOKE-200",
527 "error": "OK"
528 }
```

12. Click **Create Basket (AddAfterConfig)**.



13. Below **Path Variables**, click the **VALUE** box for **catalogcode**.

14. Type `DC_MOBILE`.

15. In the URL bar, delete `/:cartContextKey`. Since we do not have an existing basket to add a product to, we do not have a `cartContextKey`. We will remove this optional variable for this reason. Your URL should look like this:

POST	▼	<code>{{baseUrl}}/v3/catalogs/:catalogcode/basket</code>
------	---	--

16. Click the **Body** header under the URL bar.

17. Replace everything after “**productConfig**”: by pasting the **offerDetails** you copied previously. It should look like this:

```
POST {{baseUrl}}/v3/catalogs/:catalogcode/basket/

Params Authorization Headers (11) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON

1  {
2    "basketAction": "AddAfterConfig",
3    "productConfig": {
4      "offerDetails": {
5        "offer": {
6          "addtocart": {
7            "rest": {
8              "method": "POST",
9              "link": "v3/catalogs/DC_MOBILE/basket",
10             "params": {
11               "offer": "C-PHO-013",
12               "basketAction": "AddWithNoConfig",
13               "context": "{ \"DIM_PRIORITY\": \"null\", \"Region\": \"None\" }"
14             }
15           }
16         },
17         "priceResult": [
18           {
19             "adjustments": [],
20             "overrides": [],
21             "chargeamount": 799.99000,
22             "baseamount": 799.99000,
23             "Amount__c": 799.99000,
24             "IsVirtualPrice__c": false,
25             "IsBasePrice__c": true,
26             "effectiveuntildatespec": null,
27             "effectivefromdatespec": null,
28             "SubType__c": "Standard",
29             "Type__c": "PRICE",
30             "RecurringFrequency__c": null,
31             "ChargeType__c": "One-time",
32             "DisplayText__c": "Starting at $799.99",
33             "pleEffectiveFrom": "2021-11-30T05:00:00.000Z",
34             "UnitPrice": 799.99000,
35             "OneTimeTotal__c": 1599.98,
36             "OneTimeCharge__c": 799.99000
37           }
38         ]
39       }
40     }
41   }
```

18. At or near line 136, you should have **"userValues": null**. This is directly following the color attribute **"ATT_RT_CLR"**. This is where we can specify the color. Replace **null** with **"Green"**. This corresponds with the **value** listed directly above **userValues**.

```
286 ..... "id": "f619573b-8740-a23f-6044-10c54b5b5e1b",
287 ..... "name": "f619573b-8740-a23f-6044-10c54b5b5e1b",
288 ..... "label": "Green",
289 ..... "readonly": false,
290 ..... "disabled": false,
291 ..... "value": "Green",
292 ..... "defaultSelected": false,
293 ..... "displaySequence": 70
294 ..... }
295 ..... ],
296 ..... "userValues": "Green"
297 ..... },
```

19. Click the **Send** button



20. Examine the response.

```
{
  "cartContextKey": "2705b0e34b1d624fda08b2bd5b1dd541",
  "result": {
    "totals": {
      "EffectiveOneTimeTotal__c": 1599.98,
      "EffectiveRecurringTotal__c": 0.00
    },
    "records": [
      {
        "actions": {
          "deleteFromBasketAction": {
            "rest": {
              "params": {
                "basketAction": "deleteFromBasket",
                "lineItemKey": "6f6655de42b0dd90726803fd1f013379",
                "bundleContextKey": "6f6655de42b0dd90726803fd1f013379"
              },
              "link": "/v3/catalogs/DC_MOBILE/basket/2705b0e34b1d624fda08b2bd5b1dd541?contextKey=122357d20eed46c1f199521e062238bb",
              "method": "deleteFromBasketAction"
            }
          }
        }
      }
    ]
  }
}
```

Notice that we have a **cartContextKey**. We can use this parameter to retrieve this cached basket using the various basket APIs. The response offers different actions we can take on this basket including deleting basket items with **deleteFromBasketAction** and updating basket items with **updateBasketAction**.

```
146     "userValues": "Green",
147     "values": [
148       {
149         "displaySequence": 40,
150         "defaultSelected": false,
151         "value": "Purple",
152         "disabled": false,
153         "readonly": false,
154         "label": "Purple",
155         "name": "2f4f0bdd-14bf-5de9-5bdf-d29d5ad9cb7f",
156         "id": "2f4f0bdd-14bf-5de9-5bdf-d29d5ad9cb7f"
157       },
158     ]
159   }
160 }
```

Notice that our configured value for the color attribute is still specified. We do not receive any messages regarding product configuration as the color attribute was the only one that did not previously have a value and is designated as required in our product catalog.

Checkout Phase APIs

The **Create Cart API** provides the ability to create a cart after items have been added to a basket. The API provides the ability to convert the basket into a corresponding order and order line item.

First, offers are configured without a cart, then this API is called to add the basket contents to the cart (order and order line items) before submitting. The API creates a new cart. The API will validate and price the cart after adding the items and returns a link to the full cart contents.

The `CacheAPI.CreateCartFromContextKey` configuration setting can be used to determine the behavior of the Create Cart API. If set to true, the Create Cart API uses the `cartContextKey` passed in the input to create the cart. If set to false or not specified, the JSON result from the Basket API must be passed in the request body of the Create Cart API to create the cart. The API requires the user account, catalog code, and `cartItems` in the body of the request.

Task 5: Execute Create Cart From Basket

For this scenario, we are simulating a purchase for an Account in training playground for John Smith. In the next exercise, we'll give you some more information about anonymous vs known user API calls.

1. Copy the **cartContextKey** from **Create Basket (AddAfterConfig)**.

```
1  {
2    "cartContextKey": "2705b0e34b1d624fda08b2bd5b1dd541",
3    "result": {
4      "totals": {
5        "EffectiveOneTimeTotal__c": 1599.98,
6        "EffectiveRecurringTotal__c": 0.00
7      },
8    }
9  }
```

2. Click **Create Cart from Basket Key**.

▼ POST Create Cart from Basket Key

☐ e.g. successful operation

☐ e.g. Invalid status value

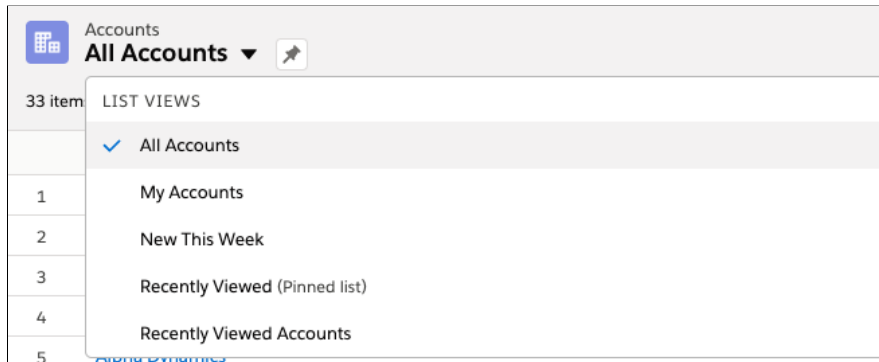


NOTE:

The Create Cart from Basket API call is similar to Create Cart from Basket Key but it requires copying the JSON response from the Create Basket call rather than just using the **cartContextKey** for the Create Cart from Basket Key API.

3. Click the **Body** header below the URL bar.
4. Replace the value for **cartContextKey** with the value you copied.
5. Replace the value for **catalogCode** with **DC_MOBILE**.
6. Open your training playground in your browser.
7. Click **Accounts** in the Lightning navigation bar.

8. Change the list view to **All Accounts**.



9. Select **John Smith**.

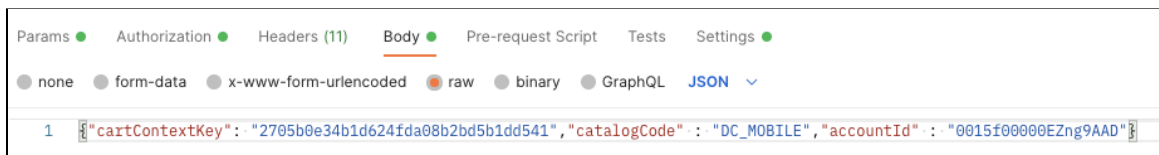
10. Click on the browser URL bar.

11. Copy the value that follows **Account/** and precedes **/view**. This is the Account ID for this user. Copy this value.

```
lightning.force.com/lightning/r/Account/0016g00000CgxXYAAZ/view
```

12. Return to **Postman**.

13. Paste John Smith's Account ID for **accountId**



14. Click the **Send** button.

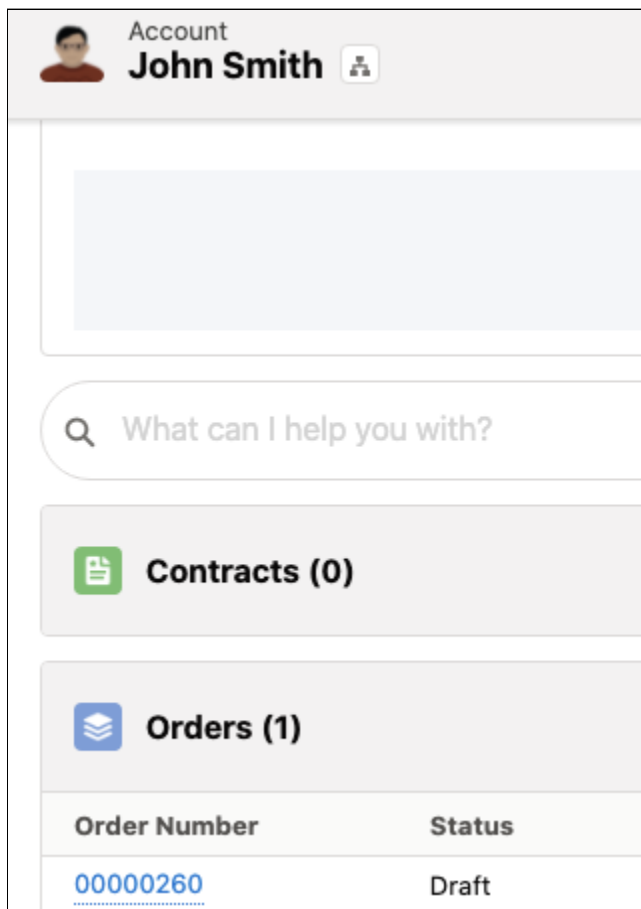


15. Examine the response.

```
1  {
2    "actions": {
3      "cartItems": {
4        "rest": {
5          "params": {},
6          "method": "GET",
7          "link": "/services/apexrest/vlocity_cmt/v2/cpq/carts/8015f000000TtCpAAK/items"
8        },
9        "remote": {
10         "params": {}
11       },
12       "client": {
13         "records": [],
14         "params": {}
15       }
16     },
17     "orderNumber": "00000260",
18     "orderId": "8015f000000TtCpAAK",
19     "errorCode": "INVOKE-200",
20     "error": "OK"
21   }
22 }
```

Our Order has been successfully been created!

If we refresh our Account page for John Smith, we can see the order.



Notice that while we created an Order, it is in Draft because it has not been submitted and activated. Let's perform the same call but have it both created and submitted by using the **createAsset** parameter.

16. Return back to **Postman**.

17. Click the **Params** header below the URL bar.

18. Where it says **Key**, type in `createAsset`.

19. Next to **createAsset**, in the **VALUE** column, type in `true`.

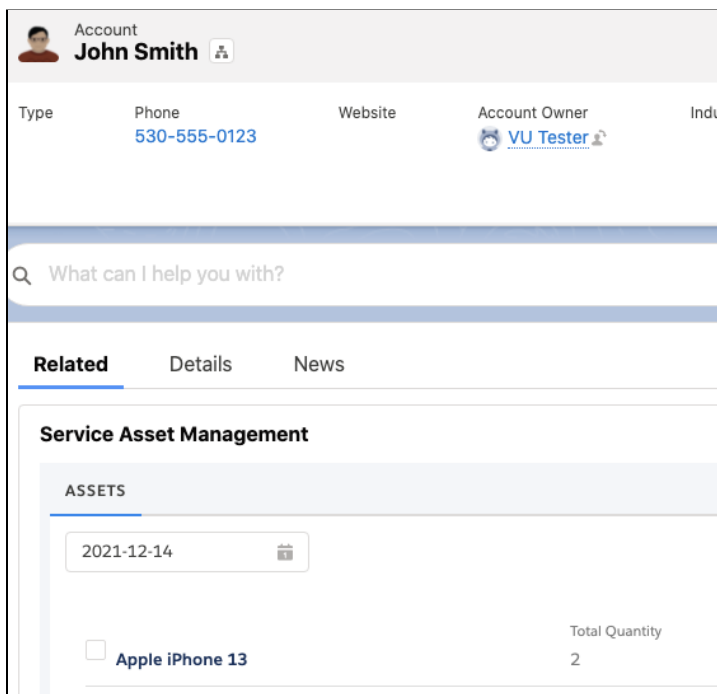
20. Click the **Send** button



21. Examine the response.

```
1  {
2    "totalSize": 1,
3    "messages": [
4      {
5        "code": "SUBMIT-1008",
6        "severity": "INFO",
7        "message": "Order 8015f000000TtDEAA0 was assetized and activated."
8      }
9    ],
10   "records": [
11     {
12       "messages": [],
13       "displaySequence": -1,
14       "id": "0015f00000EZng9AAD",
15       "objectType": "Account",
16       "clonedObjectIds": ["02i5f000000Qg4zAAC"]
17     }
18   ]
19 }
```

Notice this time we receive a **message** giving us the Order number and letting us now that it has been assetized and activated. We can refresh John Smith's account page to see this in action:



Type	Phone	Website	Account Owner	Indu
	530-555-0123		VU Tester	

What can I help you with?

Related Details News

Service Asset Management

ASSETS

ASSETS	Total Quantity
2021-12-14	
<input type="checkbox"/> Apple iPhone 13	2

Exercise 7-4: Try Out Context Rules with the Digital Commerce APIs

Scenario

Eliza knows that her company will require promotions for customers designated as high priority. In that regard, she's eager to test context rules with the Digital Commerce APIs. She decides to create an iPhone 13 promotion with a cacheable context rule and then regenerate her cache to account for this change. Finally, she will figure out how to execute Digital Commerce APIs using the context parameter.

Goal

- Create a cacheable context rule and regenerate the cache
- Execute Browse phase Digital Commerce APIs using the context parameter to see the enforcement of the previously created context rule

Tasks

1. Verify the context rule and make it cacheable
2. Execute Get Offers by Catalog with the context parameter

Time: 20 mins

Anonymous and Known User Browsing

In the last exercise, we made a purchase for a known user named John. Up until that point, we had been making API calls for anonymous users but in order to create an Order, we had to make the last call for a known user. In reality, your commerce experience will be making API calls for both known and anonymous users at various points in the Browse, Configure, Cart, and Checkout phases. In this exercise, we will explore context eligibility rules for a logged-in user, which is just one use case for known user browsing.



RESOURCE:

We **highly** recommend reading the documentation for [Anonymous and Known User Browsing](#) as well as its subpages.

Context Rules and Digital Commerce APIs

Context rules can be used to control product availability and pricing eligibility for your users. When it comes to Digital Commerce APIs, this is achieved by appending a user context parameter to API calls. For example, if you wanted to restrict a new subscriber promotion to only new customers, you could create a context rule that qualifies that promotion for only new customers.







RESOURCE:

Please see the [Context Rules](#) course within [Build CPQ Solutions for Industries](#) to learn how to create context rules.

One prominent Digital Commerce design pattern in the Browse phase is making select offers and promotions available to select users, based on a variety of criteria. In order to accomplish this, you will need to create and maintain context rules in your shared catalog, for the products and promotions that are in your sales catalog. Once you have context rules set up with some of your Digital Commerce Catalog products, you can test them using the **Get Offers By Catalog** API.

For the purposes of this exercise, we have already created the context rule. However, you will need to make it cacheable by modifying its context dimension. The context rule in this exercise is designed to override the iPhone 13 product's price in order to offer it for \$100 less than its standard price. This is accomplished through the use of a promotion. Finally, the context rule is applied to that promotion to restrict it to high priority customers only.

Task 1: Verify the context rule and make it cacheable

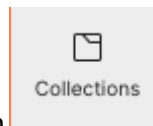
1. In the Lightning navigation bar, right-click on **Vlocity Product Console** and select **Open link in new tab**.
2. In the Dashboard under **Product Management**, click the **search icon**  next to **Promotion**.
3. Click the **search icon** .
4. Select the **iPhone 13 High Pri Customer Promo** row.
5. Click on **Product Adjustments** in the left sidebar.
6. Select the **Apple iPhone 13** row.
7. Click **PRODUCT PRICING** on the right sidebar.
8. Click **OVERRIDES** just below PRODUCT PRICING. Notice the **PRICING ELEMENT** enabling the price to be \$699.99, \$100 less than the standard price for this product.
9. Click on **Context Rules** in the left sidebar. Notice that the **High Priority Customers** rule set has been applied to this promotion. At this point you may look at this rule set and corresponding rule via the Vlocity Product Console. Once complete, continue on with this task.
10. Click **Dashboard** on the top-left area of the page, to the left of **Search Promotion**.
11. In the Dashboard under **Rules**, click the **search icon**  next to **Context Dimension**.
12. Click the **search icon** .
13. Select the **Priority** row.

14. Notice the checkbox is checked next to **Cacheable Mode**. This ensures that this rule and its **Values for caching** will be cached when the Digital Commerce API cacheable jobs/APIs are executed. In our case, this will ensure that there is a cached entry for the iPhone 13 High Pri Customer Promo that is only available to high priority customers.

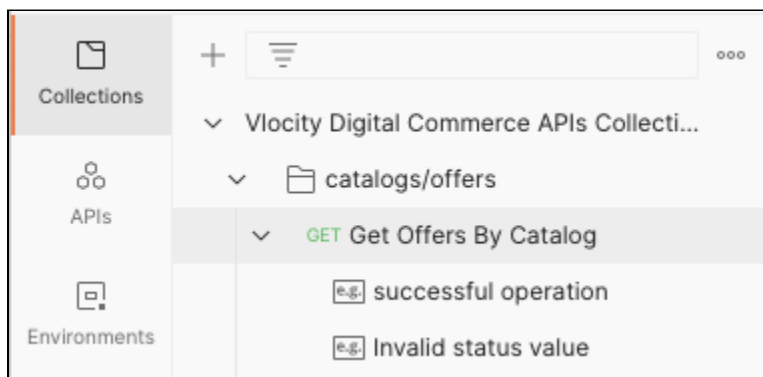
Notice the **Default Value** is null and the **Values for caching** are High,Medium,Low. [Default Value](#) specifies the value that will be used for the Context Dimension (and passed to the Context Rules) when the input context parameters do not have a given value. The Values for caching are the three possible values that a customer may have designated for their optional Priority field on their Account. The null value will be used for customers that do not have a designated Priority value.

Task 2: Execute Get Offers by Catalog with the context parameter

1. For this scenario, we are simulating a logged-in user, John Smith, browsing products, and the product list reveals a product that he specifically is eligible for. Open **Postman**.



2. Select the **Collections** button on the left side of the window.
3. Click **Get Offers By Catalog**.



4. Below Query Params, click the **checkbox** ☐ next to **context**.
5. To the right of **context**, click the **VALUE** box.
6. Type in `{"accountId" : "[JOHN_SMITH_ACCOUNT_ID]"}`. Do not copy and paste this from this guide into Postman, type it manually.

Replace `[JOHN_SMITH_ACCOUNT_ID]` with the value you used in Exercise 7-3 Task 5. Make sure you type this is and do not copy and paste from this guide, otherwise you may receive an error.

We should receive a list of products that John Smith is qualified for.

7. Below Query Params, click the **checkbox** ☐ next to **isloggedin**.
8. To the right of **isloggedin**, in the **VALUE** box, type `true`.

Send

9. Click the **Send** button .

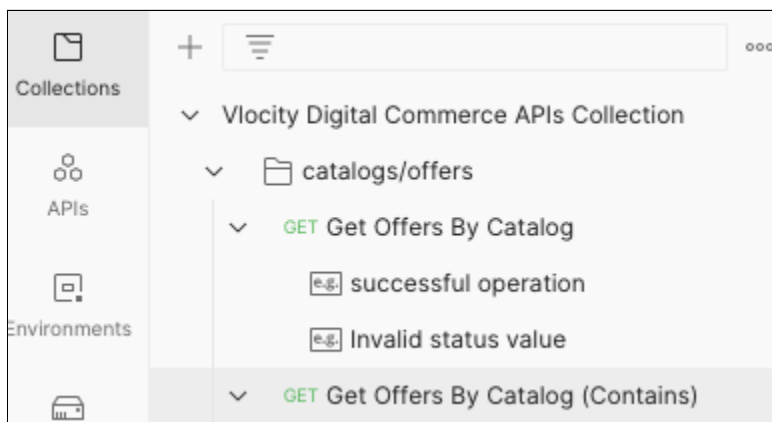
10. Examine the response.

```
"Name": "iPhone 13 High Pri Customer Promo",
"vlocity_cmt__Code__c": "PRO-003",
"vlocity_cmt__EffectiveStartDate__c": "2021-11-30T05:00:00.000Z",
"vlocity_cmt__Description__c": "High priority customers receive an additional $100 off
    the new iPhone 13.",
"vlocity_cmt__IsActive__c": true,
"vlocity_cmt__IsOrderable__c": true,
"Product": [
  {
    "priceResult": [
      {
        "adjustments": [],
        "overrides": [
          {
            "Amount__c": 699.99000,
            "Source__c": "promotion",
            "Id": "a215f000000d018AAE",
            "effectiveuntildatespec": null,
            "effectivefromdatespec": null,
            "SubType__c": "Standard",
            "Type__c": "Price",
            "RecurringFrequency__c": null,
            "ChargeType__c": "One-time",
            "DisplayText__c": "$699.99",
            "pleEffectiveFrom": "2021-11-30T05:00:00.000Z",
            "UnitPrice": 699.99000
          }
        ],
        "chargeamount": 699.99000,
        "baseamount": 699.99000,
        "Amount__c": 799.99000,
```

Notice that we are receiving the iPhone 13 High Pri Customer Promo as John Smith has qualifies for it. If you uncheck the **context** and **isloggedin** parameters and send this request again, you will see the iPhone 13 High Pri Customer Promo disappear because it is only available to “High” Priority customers thanks to our context rule.

11. Now let's try the **contains** parameter since it is a slight variation of the Get Offers By Catalog call. The "Phones" sales catalog contains an iPhone X promotion. C-PHO-001 is the offer code for the Apple iPhone X. We will simulate searching for promotions/bundles that *contain* the Apple iPhone X.

Click **Get Offers By Catalog (Contains)**.



12. Next to the **contains** parameter below **Query Params**, enter C-PHO-001.


<input checked="" type="checkbox"/>	contains	C-PHO-001
-------------------------------------	----------	-----------

13. Below Query Params, click the **checkbox** ☐ next to **context**.

14. Paste the following into the **VALUE** box next to **context** in this format:
`{"accountId":"[JOHN_SMITH_ACCOUNT_ID]"}` Replace the value [JOHN_SMITH_ACCOUNT_ID] with the value for your John Smith Account Id used previously.

15. Below Query Params, click the **checkbox** ☐ next to **isloggedin**.

16. In the **VALUE** box next to **isloggedin**, enter `true`.

	<input checked="" type="checkbox"/>	contains	C-PHO-001
	<input type="checkbox"/>	contextkey	<string>
	<input checked="" type="checkbox"/>	context	{"accountId":"0015f00000EZng9AAD"}
	<input type="checkbox"/>	forceinvalidatecache	<boolean>
	<input checked="" type="checkbox"/>	isloggedin	true

17. Below **Path Variables**, click the **VALUE** box for **catalogcode**.

18. Type `Phones`.

19. Click the **Send** button



20. Examine the response.

```
    "Name": "iPhone X plus Mobile Streaming Channels Offer",
    "Description": "iPhone X plus voice and data plan with streaming channels",
    "ProductCode": "C-OFF-020",
    "vlocity_cmt__SellingStartDate__c": "2018-01-01T08:00:00.000Z",
    "vlocity_cmt__IsOrderable__c": true,
    "IsActive": true,
    "vlocity_cmt__SpecificationType__c": "Offer",
    "Attachments": null,
    "offerType": "Product"
  },
  {
    "addtocart": {
      "rest": {
        "params": {
          "basketAction": "AddWithNoConfig",
          "offer": "C-PHO-001"
        },
        "link": "v3/catalogs/Phones/basket",
        "method": "POST"
      }
    }
  }
}
```

Notice that we only get the **iPhone X plus Mobile Streaming Channels Offer** promotion that contains the **Apple iPhone X** and its offer details, as well as the offer details for the **Apple iPhone X** itself. We do not receive any other products unrelated to C-PHO-001 (i.e., Apple iPhone X).

Exercise 7-5: Explore Digital Commerce API Parameters

Scenario

Eliza started looking at the Digital Commerce API documentation and quickly got overwhelmed with all of the different possible parameters available. She decides to systematically test each type of parameter to better understand how exactly they work.

Goal

- Execute calls with the contextkey, pagesize, offset, includeAttachment, and validatebasket parameters.

Tasks

1. Execute Get Offers by Catalog with the pagesize and offset parameters
2. Execute Get Basket with the contextkey parameter
3. Execute Get Basket with the includeAttachment parameter
4. Execute Create Cart from Basket with the validate parameter

Time: 25 mins

Digital Commerce Cacheable API Parameters

Cacheable API calls can be modified with many different parameters. This allows you to customize the functionality of your guided selling experience. Here a few examples:

No parameters - This API call shown here simply returns a list of offers for the “Phones” sales catalog. This API call has no parameters applied to it.

Return a list of offers from the “Phones” sales catalog:
`vlocity_cmt/v3/catalogs/Phones/offers`

Specify a context - The API call shown here appends a context, in this case it is specifying the context dimension of “Status” and the value of “Active”. In other words, it will retrieve all

offers with a status of active. This is specifying an eligibility for products and it works similarly for promotions.

You can also determine the eligibility for specific price list entries by applying a context eligibility rule. However, applying a context eligibility rule for a price list entry is only available for known user browsing specific to the Get Offer Details API. The price list entry is returned as a part of the response.

Return a list of offers and specify a context:
`vlocity_cmt/v3/catalogs/Phones/offers?context={"Status":"Active"}`

Specify a page size - The API call shown here returns a list of offers with page size of 20. This means that up to 20 offers will load on the user's initial page load. If more than 20 offers exist in the "Phones" catalog, the user will need to click a button to load the rest.

Return a list of offers and specify a page size of 20:
`vlocity_cmt/v3/catalogs/Phones/offers?pageSize=20`

Specify a logged-in user - The API call shown here retrieves a list of offers specific for a particular authenticated (logged-in) user.

Return a list of offers for a logged-in user:
`vlocity_cmt/v3/catalogs/Phones/offers?
context={"accountId": "0014x00000Di4O1AAJ"}&isloggedin=true`

What are Context, Basket, and Cart Context Keys?

Context keys are identifiers created for anonymous users based on applicable cacheable context dimensions for your sales catalogs. Context keys are added to the cache and reused for customers with the same context scope. In essence, this is how the API will return a response very quickly for anonymous users with a generic context since it does not need to calculate a response (i.e., process rules, validation, etc.).

```
/services/apexrest/cacheableapis/v3/catalogs/Mobiles/offers?  
context={"AccountSla":"1234", "Region":"CA"}  
&offset=0&pageSize=20&contextKey=d9ac6e46f20b05bd940d6f3d894e86e6
```

For logged-in users, context keys are identifiers created based on the user's account, contracts, and assets. For `getOffers` and `getOfferDetails` calls, the `catalogCode` and `offerCode` are also used to generate a Context key.

Context keys do not need to be unique for subsequent calls (e.g., a `getOffers` call may have the same context key as a `getOffersDetails` call).

Basket keys are identifiers that point to the cached entry for the user's cart items. If a cached entry does not already exist for the user's cart contents, a new entry will be created in the cache with a new basket key.

```
/services/apexrest/cacheableapis/v3/catalogs/Mobiles/basket/  
784b264ea6dc63690369e721ffd8e112
```

This basket key can be used with the **Get Basket Details API** to see and use the basket's contents.

Cached baskets, similar to context keys, may be reused for multiple user responses so long as the basket contents being requested are the same.

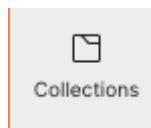
Cart context keys serve a similar purpose to basket keys but they are used for converting a basket to a cart (i.e., Order). Unlike basket keys, they are not used in the API URI, but rather included in the request body.

```
URI:  
/services/apexrest/cacheableapis/v3/carts  
  
Header:  
{  
  "accountId": "0014R00002k5vPG",  
  "catalogCode": "Phones",  
  "cartContextKey": "1b42e612a401a072324c7ca026bc4e2f"  
}
```

This cart context key is used with the **Create Cart from Basket API** to convert the basket into a corresponding order and order line item.

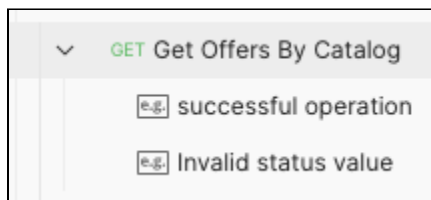
Task 1: Execute Get Offers by Catalog with the pagesize and offset parameters

1. Open **Postman**.



2. Select the **Collections** button on the left side of the window if it's not already selected.

3. Click **Get Offers By Catalog**.



4. Below Query Params, click the **checkbox** ☐ next to **pagesize**. Notice the URL appends the parameter.
5. To the right of **pagesize**, click the **VALUE** box.
6. Enter 2.
7. Below Query Params, click the **checkbox** ☐ next to **offset**. Notice the URL appends the parameter.
8. To the right of **offset**, click the **VALUE** box.
9. Enter 2.
10. Below Path Variables, modify the VALUE box to the right of **catalogcode** by replacing its contents with **DC_SAMPLE_MOBILE**. This sales catalog has the following products in it: Apple iPhone X, Apple iPhone Xs, Individual Simple Choice Plan, and Family Simple Choice Plan.

Send

11. Click the **Send** button .

12. Examine the response.

```
3    "offers": [  
4      {  
5 >    "addtocart": { ...  
14    },  
15 >    "offerDetailsAction": { ...  
28    },  
29 >    "priceResult": { ...  
47    },  
48    "Name": "Individual Simple Choice Plan",  
49    "Description": "You are rocking just you!",  
50    "ProductCode": "C-MPP-001",  
51    "vlocity_cmt__SellingStartDate__c": "2017-10-01T07:00:00.000Z",  
52    "vlocity_cmt__IsOrderable__c": true,  
53    "IsActive": true,  
54    "vlocity_cmt__SpecificationType__c": "Product",  
55 >    "AttributeCategory": { ...  
64    },  
65 >    "Attachments": { ...  
76    },  
77    "offerType": "Product"  
78  },  
79  }  
80 >    "addtocart": { ...  
89    },  
90 >    "offerDetailsAction": { ...  
93    },  
94 >    "priceResult": { ...  
22  },  
  ]  
]
```

Notice it is only returning the second page of offers because the offset is 2 and each page contains 2 offers each. Therefore, it is skipping the first page (two offers) and returning the second page (last two offers in the catalog).

13. Below Query Params, click the **checkbox** ☒ next to **offset**. Let's see what the default behavior of this API will do in comparison.

Send

14. Click the **Send** button .

15. Examine the response.

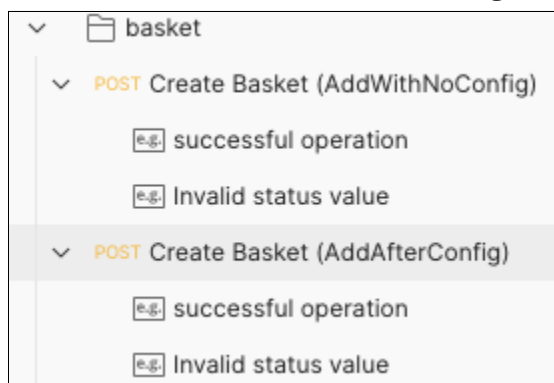
```
3      "offers": [  
4        {  
5 >         "addtocart": { ...  
14        },  
15 >         "offerDetailsAction": { ...  
28        },  
29 >         "priceResult": [ ...  
48        ],  
49         "Name": "Apple iPhone X",  
50         "Description": "With iPhone X, the device is the display. An all-new 5.8-inch Super  
                    Retina screen fills the hand and dazzles the eyes.",  
51         "ProductCode": "C-PHO-001",  
52         "vlocity_cmt__SellingStartDate__c": "2017-10-01T07:00:00.000Z",  
53         "vlocity_cmt__IsOrderable__c": true,  
54         "IsActive": true,  
55         "vlocity_cmt__SpecificationType__c": "Product",  
56 >         "AttributeCategory": { ...  
405        },  
406 >         "Attachments": [ ...  
437        ],  
438         "offerType": "Product"  
439        },  
440        {  
441 >         "addtocart": { ...  
450        },  
451 >         "offerDetailsAction": { ...  
464        },  
465 >         "priceResult": [ ...  
484        ],  
485         "Name": "Apple iPhone Xs",  
486         "Description": "A Super Retina OLED display. Even faster Face ID. And a breakthrough
```

Now the response contains just the first page of offers. The pagesize is still defined as 2 so we are only seeing the first two offers.

Task 2: Execute Get Basket with the cartContextKey parameter

1. Let's get a **cartContextKey** by reviewing our response from the Create Basket API we used in Exercise 7-3 Task 4. We can use that cartContextKey for this task. If you no longer have that cartContextKey, please create a new basket with the Create Basket API following the steps in Exercise 7-3:

- a. Click **Create Basket (AddAfterConfig)**.



- b. Within the **Body** response, copy the value for **cartContextKey**.

```
1 {  
2   "cartContextKey": "2705b0e34b1d624fda08b2bd5b1dd541",  
3   "result": {  
4     "totals": {  
5       "EffectiveOneTimeTotal__c": 1599.98,
```

2. Click **Get Basket Details** from the left sidebar.

- a. Below Path Variables, modify the following values:

KEY	VALUE
catalogcode	DC_MOBILE
cartContextKey	[Paste the value copied in the previous step]

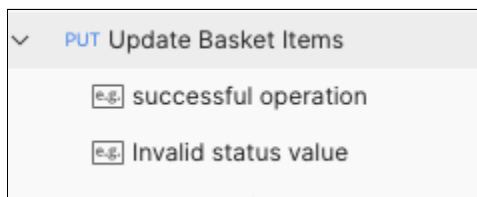
- b. Click the **Send** button .

c. Examine the response.

```
{
  "bundleContextKey": "6f6655de42b0dd90726803fd1f013379",
  "lineItemKey": "6f6655de42b0dd90726803fd1f013379",
  "actions": {
    "updateBasketAction": {
      "rest": {
        "method": "updateBasketAction",
        "link": "/v3/catalogs/DC_MOBILE/basket/2705b0e34b1d624fda08b2bd5b1dd541?contextKey=122357d20eed46c1f199521e062238bb",
        "params": {
          "bundleContextKey": "6f6655de42b0dd90726803fd1f013379",
          "lineItemKey": "6f6655de42b0dd90726803fd1f013379",
          "basketAction": "updateBasket"
        }
      }
    },
    "deleteFromBasketAction": {
      "rest": {
        "method": "deleteFromBasketAction",
        "link": "/v3/catalogs/DC_MOBILE/basket/2705b0e34b1d624fda08b2bd5b1dd541?contextKey=122357d20eed46c1f199521e062238bb",
        "params": {
          "bundleContextKey": "6f6655de42b0dd90726803fd1f013379",
          "lineItemKey": "6f6655de42b0dd90726803fd1f013379",
          "basketAction": "deleteFromBasket"
        }
      }
    }
  }
}
```

Beyond simply seeing the basket details again, we can also examine a couple of the possible actions we didn't explore before. Notice under **updateBasketAction**, we are given a **link**. The string that immediately follows **/v3/catalogs/DC_MOBILE/basket/** is the **basketKey**. In our example, the basketKey is 2705b0e34b1d624fda08b2bd5b1dd541.

3. Now let's try updating the basket:
 - a. Below **updateBasketAction**, copy the **bundleContextKey**, **lineItemKey**, and **basketAction** to a text editor or notepad file so that you can paste them in the next step.
 - b. Click **Update Basket Items** from the left sidebar.



- c. Below Path Variables, modify the following values:

KEY	VALUE
catalogcode	DC_MOBILE
cartContextKey	[Paste the value copied in the step 1b]

- d. Click **Body** below the URL bar.
- e. Paste the appropriate values for **bundleContextKey**, **lineItemKey**, and **basketAction**.
- f. For **Quantity**, type in 3.
- g. Delete the comma after “3” as well as **attributeCategories** on the following line as we are not modifying attributes. Your Body should look like this:

```
{
  "bundleContextKey": "6f6655de42b0dd90726803fd1f013379",
  "lineItemKey": "6f6655de42b0dd90726803fd1f013379",
  "basketAction": "updateBasket",
  "Quantity": "3"
}
```

- h. Click the **Send** button .

- i. Examine the response.

```
{
  "totals": {
    "EffectiveRecurringTotal__c": 0.00,
    "EffectiveOneTimeTotal__c": 2399.97
  }
}
```

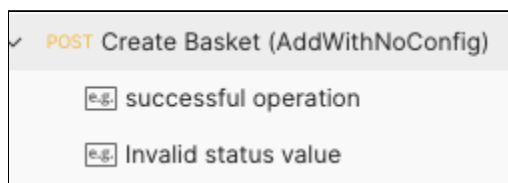
Notice that we get back an updated basket in the response. Our iPhone 13 now has a quantity of 3 and the total price of the basket has been updated.

Task 3: Execute Get Basket with the includeAttachment parameter

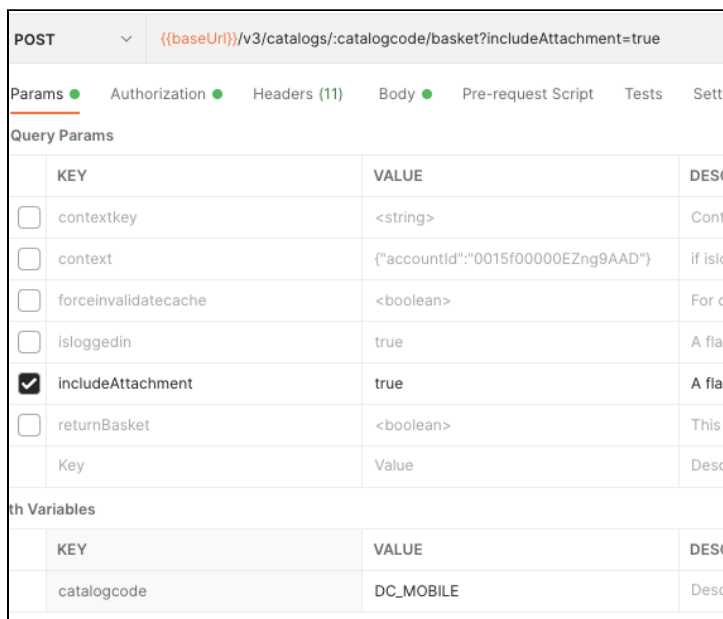
The **includeAttachment** parameter determines whether the API returns product attachment URIs or not. If **includeAttachment** is true, the basket API returns a URI to the attachments. If **includeAttachment** is false, or the **isloggedin** flag is false, the basket API does not return the attachments.

The default value is false.

1. Click **Create Basket (AddWithNoConfig)**.



2. Click **Params** below the URL bar.
3. Click the ☐ checkbox next to **includeAttachment**. Notice the URL appends the parameter.
4. In the **VALUE** box next to **includeAttachment**, type in `true`.



- Click the **Body** header below the URL bar.

Your Body should look like this:

Params ● Authorization ● Headers (11) **Body ●** Pre-request Script Tests

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL

1 [{"basketAction": "AddWithNoConfig", "offer": "C-PHO-013"}]

- Click the **Send** button .

- Examine the response.

```
111  ✓      "Attachments": [  
112  ✓      |  
113      |   "url": "https://www.salesforce.com/content/dam/web/en_us/www/images/  
114      |     resources/training-images/iPhoneX.png",  
115      |   "title": "Hero_shot",  
116      |   "sequenceNumber": 10,  
117      |   "mimeType": "image/png",  
118      |   "displayConditionsData": null,  
119      |   "description": null,  
120      |   "defaultAttachment": true,  
121      |   "contentType": "Image",  
122  ✓      |  
123      |   "url": "https://www.salesforce.com/content/dam/web/en_us/www/images/  
124      |     resources/training-images/Apple-iPhoneX-Silver.jpg",  
125      |   "title": "Side_view",  
126      |   "sequenceNumber": 20,  
127      |   "mimeType": "image/jpeg",  
128      |   "displayConditionsData": null,  
129      |   "description": null,  
130      |   "defaultAttachment": false,  
131      |   "contentType": "Image",  
132  ✓      |  
133      |   "url": "https://www.salesforce.com/content/dam/web/en_us/www/images/  
134      |     resources/training-images/Apple-iPhoneX-Silver-2.jpg",  
135      |   "title": "Back_view",  
136      |   "sequenceNumber": 30,  
137      |   "mimeType": "image/jpeg",  
138      |   "displayConditionsData": null,  
139      |   "description": null,  
140      |   "defaultAttachment": false,  
141      |   "contentType": "Image",  
142  ✓      |  
143      |   ],  
144      |  
145      |   ],  
146      |  
147      |   ],  
148      |  
149      |   ],  
150      |  
151      |   ],  
152      |  
153      |   ],  
154      |  
155      |   ],  
156      |  
157      |   ],  
158      |  
159      |   ],  
160      |  
161      |   ],  
162      |  
163      |   ],  
164      |  
165      |   ],  
166      |  
167      |   ],  
168      |  
169      |   ],  
170      |  
171      |   ],  
172      |  
173      |   ],  
174      |  
175      |   ],  
176      |  
177      |   ],  
178      |  
179      |   ],  
180      |  
181      |   ],  
182      |  
183      |   ],  
184      |  
185      |   ],  
186      |  
187      |   ],  
188      |  
189      |   ],  
190      |  
191      |   ],  
192      |  
193      |   ],  
194      |  
195      |   ],  
196      |  
197      |   ],  
198      |  
199      |   ],  
200      |  
201      |   ],  
202      |  
203      |   ],  
204      |  
205      |   ],  
206      |  
207      |   ],  
208      |  
209      |   ],  
210      |  
211      |   ],  
212      |  
213      |   ],  
214      |  
215      |   ],  
216      |  
217      |   ],  
218      |  
219      |   ],  
220      |  
221      |   ],  
222      |  
223      |   ],  
224      |  
225      |   ],  
226      |  
227      |   ],  
228      |  
229      |   ],  
230      |  
231      |   ],  
232      |  
233      |   ],  
234      |  
235      |   ],  
236      |  
237      |   ],  
238      |  
239      |   ],  
240      |  
241      |   ],  
242      |  
243      |   ],  
244      |  
245      |   ],  
246      |  
247      |   ],  
248      |  
249      |   ],  
250      |  
251      |   ],  
252      |  
253      |   ],  
254      |  
255      |   ],  
256      |  
257      |   ],  
258      |  
259      |   ],  
260      |  
261      |   ],  
262      |  
263      |   ],  
264      |  
265      |   ],  
266      |  
267      |   ],  
268      |  
269      |   ],  
270      |  
271      |   ],  
272      |  
273      |   ],  
274      |  
275      |   ],  
276      |  
277      |   ],  
278      |  
279      |   ],  
280      |  
281      |   ],  
282      |  
283      |   ],  
284      |  
285      |   ],  
286      |  
287      |   ],  
288      |  
289      |   ],  
290      |  
291      |   ],  
292      |  
293      |   ],  
294      |  
295      |   ],  
296      |  
297      |   ],  
298      |  
299      |   ],  
300      |  
301      |   ],  
302      |  
303      |   ],  
304      |  
305      |   ],  
306      |  
307      |   ],  
308      |  
309      |   ],  
310      |  
311      |   ],  
312      |  
313      |   ],  
314      |  
315      |   ],  
316      |  
317      |   ],  
318      |  
319      |   ],  
320      |  
321      |   ],  
322      |  
323      |   ],  
324      |  
325      |   ],  
326      |  
327      |   ],  
328      |  
329      |   ],  
330      |  
331      |   ],  
332      |  
333      |   ],  
334      |  
335      |   ],  
336      |  
337      |   ],  
338      |  
339      |   ],  
340      |  
341      |   ],  
342      |  
343      |   ],  
344      |  
345      |   ],  
346      |  
347      |   ],  
348      |  
349      |   ],  
350      |  
351      |   ],  
352      |  
353      |   ],  
354      |  
355      |   ],  
356      |  
357      |   ],  
358      |  
359      |   ],  
360      |  
361      |   ],  
362      |  
363      |   ],  
364      |  
365      |   ],  
366      |  
367      |   ],  
368      |  
369      |   ],  
370      |  
371      |   ],  
372      |  
373      |   ],  
374      |  
375      |   ],  
376      |  
377      |   ],  
378      |  
379      |   ],  
380      |  
381      |   ],  
382      |  
383      |   ],  
384      |  
385      |   ],  
386      |  
387      |   ],  
388      |  
389      |   ],  
390      |  
391      |   ],  
392      |  
393      |   ],  
394      |  
395      |   ],  
396      |  
397      |   ],  
398      |  
399      |   ],  
400      |  
401      |   ],  
402      |  
403      |   ],  
404      |  
405      |   ],  
406      |  
407      |   ],  
408      |  
409      |   ],  
410      |  
411      |   ],  
412      |  
413      |   ],  
414      |  
415      |   ],  
416      |  
417      |   ],  
418      |  
419      |   ],  
420      |  
421      |   ],  
422      |  
423      |   ],  
424      |  
425      |   ],  
426      |  
427      |   ],  
428      |  
429      |   ],  
430      |  
431      |   ],  
432      |  
433      |   ],  
434      |  
435      |   ],  
436      |  
437      |   ],  
438      |  
439      |   ],  
440      |  
441      |   ],  
442      |  
443      |   ],  
444      |  
445      |   ],  
446      |  
447      |   ],  
448      |  
449      |   ],  
450      |  
451      |   ],  
452      |  
453      |   ],  
454      |  
455      |   ],  
456      |  
457      |   ],  
458      |  
459      |   ],  
460      |  
461      |   ],  
462      |  
463      |   ],  
464      |  
465      |   ],  
466      |  
467      |   ],  
468      |  
469      |   ],  
470      |  
471      |   ],  
472      |  
473      |   ],  
474      |  
475      |   ],  
476      |  
477      |   ],  
478      |  
479      |   ],  
480      |  
481      |   ],  
482      |  
483      |   ],  
484      |  
485      |   ],  
486      |  
487      |   ],  
488      |  
489      |   ],  
490      |  
491      |   ],  
492      |  
493      |   ],  
494      |  
495      |   ],  
496      |  
497      |   ],  
498      |  
499      |   ],  
500      |  
501      |   ],  
502      |  
503      |   ],  
504      |  
505      |   ],  
506      |  
507      |   ],  
508      |  
509      |   ],  
510      |  
511      |   ],  
512      |  
513      |   ],  
514      |  
515      |   ],  
516      |  
517      |   ],  
518      |  
519      |   ],  
520      |  
521      |   ],  
522      |  
523      |   ],  
524      |  
525      |   ],  
526      |  
527      |   ],  
528      |  
529      |   ],  
530      |  
531      |   ],  
532      |  
533      |   ],  
534      |  
535      |   ],  
536      |  
537      |   ],  
538      |  
539      |   ],  
540      |  
541      |   ],  
542      |  
543      |   ],  
544      |  
545      |   ],  
546      |  
547      |   ],  
548      |  
549      |   ],  
550      |  
551      |   ],  
552      |  
553      |   ],  
554      |  
555      |   ],  
556      |  
557      |   ],  
558      |  
559      |   ],  
560      |  
561      |   ],  
562      |  
563      |   ],  
564      |  
565      |   ],  
566      |  
567      |   ],  
568      |  
569      |   ],  
570      |  
571      |   ],  
572      |  
573      |   ],  
574      |  
575      |   ],  
576      |  
577      |   ],  
578      |  
579      |   ],  
580      |  
581      |   ],  
582      |  
583      |   ],  
584      |  
585      |   ],  
586      |  
587      |   ],  
588      |  
589      |   ],  
590      |  
591      |   ],  
592      |  
593      |   ],  
594      |  
595      |   ],  
596      |  
597      |   ],  
598      |  
599      |   ],  
600      |  
601      |   ],  
602      |  
603      |   ],  
604      |  
605      |   ],  
606      |  
607      |   ],  
608      |  
609      |   ],  
610      |  
611      |   ],  
612      |  
613      |   ],  
614      |  
615      |   ],  
616      |  
617      |   ],  
618      |  
619      |   ],  
620      |  
621      |   ],  
622      |  
623      |   ],  
624      |  
625      |   ],  
626      |  
627      |   ],  
628      |  
629      |   ],  
630      |  
631      |   ],  
632      |  
633      |   ],  
634      |  
635      |   ],  
636      |  
637      |   ],  
638      |  
639      |   ],  
640      |  
641      |   ],  
642      |  
643      |   ],  
644      |  
645      |   ],  
646      |  
647      |   ],  
648      |  
649      |   ],  
650      |  
651      |   ],  
652      |  
653      |   ],  
654      |  
655      |   ],  
656      |  
657      |   ],  
658      |  
659      |   ],  
660      |  
661      |   ],  
662      |  
663      |   ],  
664      |  
665      |   ],  
666      |  
667      |   ],  
668      |  
669      |   ],  
670      |  
671      |   ],  
672      |  
673      |   ],  
674      |  
675      |   ],  
676      |  
677      |   ],  
678      |  
679      |   ],  
680      |  
681      |   ],  
682      |  
683      |   ],  
684      |  
685      |   ],  
686      |  
687      |   ],  
688      |  
689      |   ],  
690      |  
691      |   ],  
692      |  
693      |   ],  
694      |  
695      |   ],  
696      |  
697      |   ],  
698      |  
699      |   ],  
700      |  
701      |   ],  
702      |  
703      |   ],  
704      |  
705      |   ],  
706      |  
707      |   ],  
708      |  
709      |   ],  
710      |  
711      |   ],  
712      |  
713      |   ],  
714      |  
715      |   ],  
716      |  
717      |   ],  
718      |  
719      |   ],  
720      |  
721      |   ],  
722      |  
723      |   ],  
724      |  
725      |   ],  
726      |  
727      |   ],  
728      |  
729      |   ],  
730      |  
731      |   ],  
732      |  
733      |   ],  
734      |  
735      |   ],  
736      |  
737      |   ],  
738      |  
739      |   ],  
740      |  
741      |   ],  
742      |  
743      |   ],  
744      |  
745      |   ],  
746      |  
747      |   ],  
748      |  
749      |   ],  
750      |  
751      |   ],  
752      |  
753      |   ],  
754      |  
755      |   ],  
756      |  
757      |   ],  
758      |  
759      |   ],  
760      |  
761      |   ],  
762      |  
763      |   ],  
764      |  
765      |   ],  
766      |  
767      |   ],  
768      |  
769      |   ],  
770      |  
771      |   ],  
772      |  
773      |   ],  
774      |  
775      |   ],  
776      |  
777      |   ],  
778      |  
779      |   ],  
780      |  
781      |   ],  
782      |  
783      |   ],  
784      |  
785      |   ],  
786      |  
787      |   ],  
788      |  
789      |   ],  
790      |  
791      |   ],  
792      |  
793      |   ],  
794      |  
795      |   ],  
796      |  
797      |   ],  
798      |  
799      |   ],  
800      |  
801      |   ],  
802      |  
803      |   ],  
804      |  
805      |   ],  
806      |  
807      |   ],  
808      |  
809      |   ],  
810      |  
811      |   ],  
812      |  
813      |   ],  
814      |  
815      |   ],  
816      |  
817      |   ],  
818      |  
819      |   ],  
820      |  
821      |   ],  
822      |  
823      |   ],  
824      |  
825      |   ],  
826      |  
827      |   ],  
828      |  
829      |   ],  
830      |  
831      |   ],  
832      |  
833      |   ],  
834      |  
835      |   ],  
836      |  
837      |   ],  
838      |  
839      |   ],  
840      |  
841      |   ],  
842      |  
843      |   ],  
844      |  
845      |   ],  
846      |  
847      |   ],  
848      |  
849      |   ],  
850      |  
851      |   ],  
852      |  
853      |   ],  
854      |  
855      |   ],  
856      |  
857      |   ],  
858      |  
859      |   ],  
860      |  
861      |   ],  
862      |  
863      |   ],  
864      |  
865      |   ],  
866      |  
867      |   ],  
868      |  
869      |   ],  
870      |  
871      |   ],  
872      |  
873      |   ],  
874      |  
875      |   ],  
876      |  
877      |   ],  
878      |  
879      |   ],  
880      |  
881      |   ],  
882      |  
883      |   ],  
884      |  
885      |   ],  
886      |  
887      |   ],  
888      |  
889      |   ],  
890      |  
891      |   ],  
892      |  
893      |   ],  
894      |  
895      |   ],  
896      |  
897      |   ],  
898      |  
899      |   ],  
900      |  
901      |   ],  
902      |  
903      |   ],  
904      |  
905      |   ],  
906      |  
907      |   ],  
908      |  
909      |   ],  
910      |  
911      |   ],  
912      |  
913      |   ],  
914      |  
915      |   ],  
916      |  
917      |   ],  
918      |  
919      |   ],  
920      |  
921      |   ],  
922      |  
923      |   ],  
924      |  
925      |   ],  
926      |  
927      |   ],  
928      |  
929      |   ],  
930      |  
931      |   ],  
932      |  
933      |   ],  
934      |  
935      |   ],  
936      |  
937      |   ],  
938      |  
939      |   ],  
940      |  
941      |   ],  
942      |  
943      |   ],  
944      |  
945      |   ],  
946      |  
947      |   ],  
948      |  
949      |   ],  
950      |  
951      |   ],  
952      |  
953      |   ],  
954      |  
955      |   ],  
956      |  
957      |   ],  
958      |  
959      |   ],  
960      |  
961      |   ],  
962      |  
963      |   ],  
964      |  
965      |   ],  
966      |  
967      |   ],  
968      |  
969      |   ],  
970      |  
971      |   ],  
972      |  
973      |   ],  
974      |  
975      |   ],  
976      |  
977      |   ],  
978      |  
979      |   ],  
980      |  
981      |   ],  
982      |  
983      |   ],  
984      |  
985      |   ],  
986      |  
987      |   ],  
988      |  
989      |   ],  
990      |  
991      |   ],  
992      |  
993      |   ],  
994      |  
995      |   ],  
996      |  
997      |   ],  
998      |  
999      |   ],  
1000     |  
1001     |   ],  
1002     |  
1003     |   ],  
1004     |  
1005     |   ],  
1006     |  
1007     |   ],  
1008     |  
1009     |   ],  
1010     |  
1011     |   ],  
1012     |  
1013     |   ],  
1014     |  
1015     |   ],  
1016     |  
1017     |   ],  
1018     |  
1019     |   ],  
1020     |  
1021     |   ],  
1022     |  
1023     |   ],  
1024     |  
1025     |   ],  
1026     |  
1027     |   ],  
1028     |  
1029     |   ],  
1030     |  
1031     |   ],  
1032     |  
1033     |   ],  
1034     |  
1035     |   ],  
1036     |  
1037     |   ],  
1038     |  
1039     |   ],  
1040     |  
1041     |   ],  
1042     |  
1043     |   ],  
1044     |  
1045     |   ],  
1046     |  
1047     |   ],  
1048     |  
1049     |   ],  
1050     |  
1051     |   ],  
1052     |  
1053     |   ],  
1054     |  
1055     |   ],  
1056     |  
1057     |   ],  
1058     |  
1059     |   ],  
1060     |  
1061     |   ],  
1062     |  
1063     |   ],  
1064     |  
1065     |   ],  
1066     |  
1067     |   ],  
1068     |  
1069     |   ],  
1070     |  
1071     |   ],  
1072     |  
1073     |   ],  
1074     |  
1075     |   ],  
1076     |  
1077     |   ],  
1078     |  
1079     |   ],  
1080     |  
1081     |   ],  
1082     |  
1083     |   ],  
1084     |  
1085     |   ],  
1086     |  
1087     |   ],  
1088     |  
1089     |   ],  
1090     |  
1091     |   ],  
1092     |  
1093     |   ],  
1094     |  
1095     |   ],  
1096     |  
1097     |   ],  
1098     |  
1099     |   ],  
1100     |  
1101     |   ],  
1102     |  
1103     |   ],  
1104     |  
1105     |   ],  
1106     |  
1107     |   ],  
1108     |  
1109     |   ],  
1110     |  
1111     |   ],  
1112     |  
1113     |   ],  
1114     |  
1115     |   ],  
1116     |  
1117     |   ],  
1118     |  
1119     |   ],  
1120     |  
1121     |   ],  
1122     |  
1123     |   ],  
1124     |  
1125     |   ],  
1126     |  
1127     |   ],  
1128     |  
1129     |   ],  
1130     |  
1131     |   ],  
1132     |  
1133     |   ],  
1134     |  
1135     |   ],  
1136     |  
1137     |   ],  
1138     |  
1139     |   ],  
1140     |  
1141     |   ],  
1142     |  
1143     |   ],  
1144     |  
1145     |   ],  
1146     |  
1147     |   ],  
1148     |  
1149     |   ],  
1150     |  
1151     |   ],  
1152     |  
1153     |   ],  
1154     |  
1155     |   ],  
1156     |  
1157     |   ],  
1158     |  
1159     |   ],  
1160     |  
1161     |   ],  
1162     |  
1163     |   ],  
1164     |  
1165     |   ],  
1166     |  
1167     |   ],  
1168     |  
1169     |   ],  
1170     |  
1171     |   ],  
1172     |  
1173     |   ],  
1174     |  
1175     |   ],  
1176     |  
1177     |   ],  
1178     |  
1179     |   ],  
1180     |  
1181     |   ],  
1182     |  
1183     |   ],  
1184     |  
1185     |   ],  
1186     |  
1187     |   ],  
1188     |  
1189     |   ],  
1190     |  
1191     |   ],  
1192     |  
1193     |   ],  
1194     |  
1195     |   ],  
1196     |  
1197     |   ],  
1198     |  
1199     |   ],  
1200     |  
1201     |   ],  
1202     |  
1203     |   ],  
1204     |  
1205     |   ],  
1206     |  
1207     |   ],  
1208     |  
1209     |   ],  
1210     |  
1211     |   ],  
1212     |  
1213     |   ],  
1214     |  
1215     |   ],  
1216     |  
1217     |   ],  
1218     |  
1219     |   ],  
1220     |  
1221     |   ],  
1222     |  
1223     |   ],  
1224     |  
1225     |   ],  
1226     |  
1227     |   ],  
1228     |  
1229     |   ],  
1230     |  
1231     |   ],  
1232     |  
1233     |   ],  
1234     |  
1235     |   ],  
1236     |  
1237     |   ],  
1238     |  
1239     |   ],  
1240     |  
1241     |   ],  
1242     |  
1243     |   ],  
1244     |  
1245     |   ],  
1246     |  
1247     |   ],  
1248     |  
1249     |   ],  
1250     |  
1251     |   ],  
1252     |  
1253     |   ],  
1254     |  
1255     |   ],  
1256     |  
1257     |   ],  
1258     |  
1259     |   ],  
1260     |  
1261     |   ],  
1262     |  
1263     |   ],  
1264     |  
1265     |   ],  
1266     |  
1267     |   ],  
1268     |  
1269     |   ],  
1270     |  
1271     |   ],  
1272     |  
1273     |   ],  
1274     |  
1275     |   ],  
1276     |  
1277     |   ],  
1278     |  
1279     |   ],  
1280     |  
1281     |   ],  
1282     |  
1283     |   ],  
1284     |  
1285     |   ],  
1286     |  
1287     |   ],  
1288     |  
1289     |   ],  
1290     |  
1291     |   ],  
1292     |  
1293     |   ],  
1294     |  
1295     |   ],  
1296     |  
1297     |   ],  
1298     |  
1299     |   ],  
1300     |  
1301     |   ],  
1302     |  
1303     |   ],  
1304     |  
1305     |   ],  
1306     |  
1307     |   ],  
1308     |  
1309     |   ],  
1310     |  
1311     |   ],  
1312     |  
1313     |   ],  
1314     |  
1315     |   ],  
1316     |  
1317     |   ],  
1318     |  
1319     |   ],  
1320     |  
1321     |   ],  
1322     |  
1323     |   ],  
1324     |  
1325     |   ],  
1326     |  
1327     |   ],  
1328     |  
1329     |   ],  
1330     |  
1331     |   ],  
1332     |  
1333     |   ],  
1334     |  
1335     |   ],  
1336     |  
1337     |   ],  
1338     |  
1339     |   ],  
1340     |  
1341     |   ],  
1342     |  
1343     |   ],  
1344     |  
1345     |   ],  
1346     |  
1347     |   ],  
1348     |  
1349     |   ],  
1350     |  
1351     |   ],  
1352     |  
1353     |   ],  
1354     |  
1355     |   ],  
1356     |  
1357     |   ],  
1358     |  
1359     |   ],  
1360     |  
1361     |   ],  
1362     |  
1363     |   ],  
1364     |  
1365     |   ],  
1366     |  
1367     |   ],  
1
```

Task 4: Execute Add to Basket with an invalid offer

1. Let's try to create a basket for an anonymous user that does not qualify for the iPhone 13 High Pri customer Promo.
 - a. Replace the **offer** value with PRO-003.

```
1 {"basketAction": "AddWithNoConfig", "offer": "PRO-003"}
```

- - b. Click the **Send** button .

- -
 - c. Examine the response.


```
1 {
2   "success": false,
3   "sessionKey": "683d0fbd316a8bcba16ca9e682fc9831",
4   "result": {
5     "BasketValidationResult": {
6       "IneligibleOffers": {
7         "IneligibleOffers": [
8           "iPhone 13 High Pri Customer Promo"
9         ]
10      }
11    }
12  },
13  "errorCode": "422",
14  "error": "One or more offers are invalid."
15 }
```

Notice that because we did not provide a qualified Account with the appropriate Priority value of “High” in the **context**, we are receiving an error message that this anonymous user cannot create a basket with this promotion.

Now what if we did not want validation to be processed at this stage in the user experience? We can get around this with the **validatebasket** parameter.

- -
 -
 - d. Click **Params** below the URL bar.

- e. Type in `validatebasket` in the **KEY** column below **Query Params**.
- f. In the **VALUE** column next to **validatebasket**, type in `false`.

- g. Click the **Send** button .

- h. You may receive a **multiTransactionKey** in your response. When the MTSEnabled CPQ configuration setting is set to true as it is in our training playground, the Multi-Transaction Service splits Apex transactions into multiple transactions to avoid exceeding Salesforce Governor limits. If the API requires more than one transaction, an intermediate response includes the **multiTransactionKey** value, which is used to retrieve the next transaction.

Copy the value of the **multiTransactionKey**.

```
"nexttransaction": {  
  "rest": {  
    "params": {  
      "multiTransactionKey": "ed0335597d4e2a83af39dffd6acaff27",  
      "method": "addWithNoConfig"  
    }  
  }  
}
```

- i. Click **Body** below the URL bar.
- j. Add **multiTransactionKey** and its value into the Body.



```
1 {"basketAction": "AddWithNoConfig", "offer": "PRO-003", "multiTransactionKey": "ed0335597d4e2a83af39dffd6acaff27"}
```

- k. Click the **Send** button .

I. Examine the response.

```
1  ✓ "cartContextKey": "5e8e1b261af9b16216ef39d6ee9fa253",  
2  
3  ✓ "result": {  
4  ✓   "totals": {  
5     "EffectiveOneTimeTotal__c": 699.99,  
6     "EffectiveRecurringTotal__c": 0.00  
7   },  
8  ✓   "records": [  
9  ✓     {  
10     ✓   "actions": {  
11     ✓     "deleteFromBasketAction": {  
12     ✓       "rest": {  
13     ✓         "params": {  
14             "basketAction": "deleteFromBasket",  
15             "lineItemKey": "b15811d063385cf676f50fe7bb47051e",  
16             "bundleContextKey": "b15811d063385cf676f50fe7bb47051e"  
17           },  
18           "link": "/v3/catalogs/DC_MOBILE/basket/  
5e8e1b261af9b16216ef39d6ee9fa253?  
contextKey=122357d20eed46c1f199521e062238bb&  
includeAttachment=true",  
19           "method": "deleteFromBasketAction"  
20         }  
21       },  
22     ✓   "updateBasketAction": {  
23     ✓     "rest": {  
24     ✓       "params": {  
25             "basketAction": "updateBasket",  
26             "lineItemKey": "b15811d063385cf676f50fe7bb47051e",
```

Notice that we are now able to create the basket despite the anonymous user not being qualified for this promotion. However, if we try to create this order using its **cartContextKey** for an Account that does not qualify for this offer, you will receive an error message from basket validation.

```
"success": false,  
"cartContextKey": "52ef9511c817df55976a639821779d0c",  
"result": {  
  "BasketValidationResult": {  
    "isBasketValid": false,  
    "basketValidationMessage": "One or more offers are ineligible.",  
    "IneligibleOffers": [  
      "iPhone 13 High Pri Customer Promo"  
    ]  
}
```



Yay! All done!