Microsoft Dynamics 365 Commerce

Lab: Headless Commerce

Hands-on lab step-by-step

April 2025

Version 1.0

Contents

This lab is subject to the Terms of Use found at the end of this document.

This comprehensive lab is designed to provide you with hands-on experience in accessing and utilizing Commerce Headless. You will begin with a guided tour to familiarize yourself with Commerce Headless APIs. Next, you will explore different APIs that support various business processes, ensuring you understand how they function and can apply them effectively. Finally, you will perform practical exercises to access these APIs on your own, gaining a solid grasp of how to navigate and utilize the Commerce Headless system with different security roles and permissions.

Goals for this lab

After this lab you will be able to: • Understand how to access Headless Commerce using different security roles. • Learn the differences in access and permissions based on security roles. • Understand different APIs that support various business processes. • Perform a hands-on lab to access the APIs on your own and learn how they work.	The time to complete this lab is 60 **minutes**.

Prerequisites

Ensure that Insomnia is downloaded and installed on your workstation. Access to your own workstation and an active internet connection are required to complete this lab successfully.

Note

This document introduces the Headless commerce API for lab use. It does not cover all APIs. For the latest details, see the Open API documentation for each service version.

Lab 01: Configure Insomnia

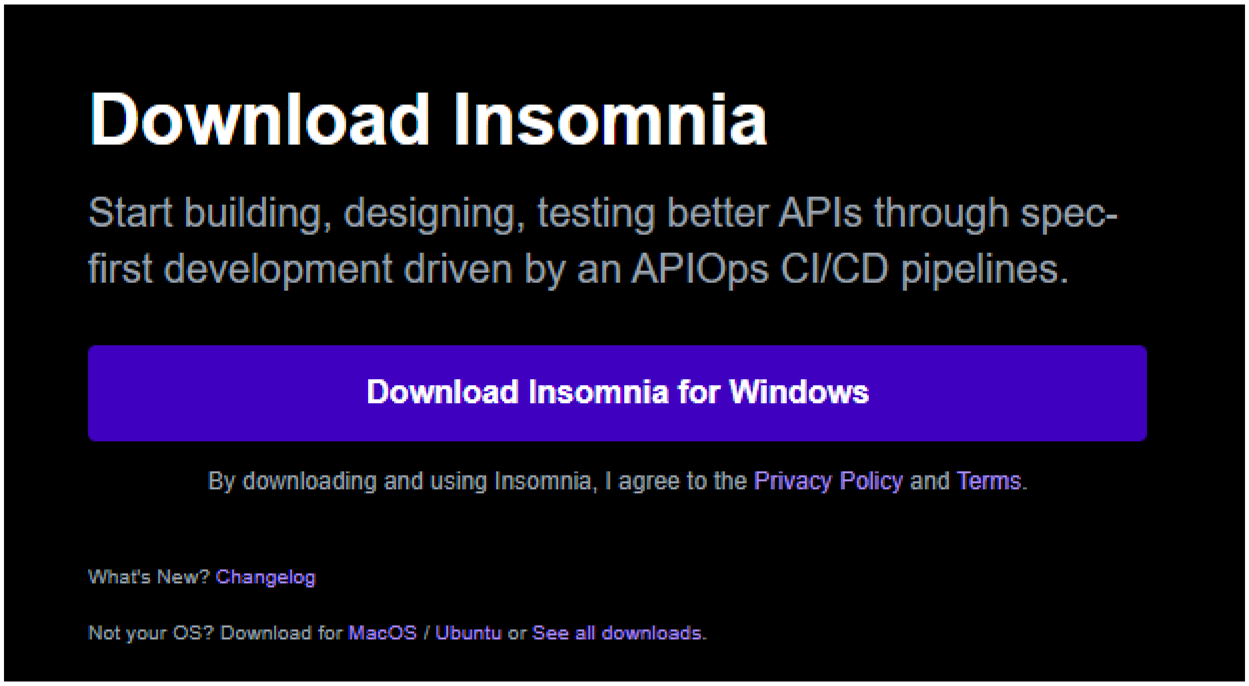
Exercise 1: Download - Insomnia and installation.

Insomnia is an API client that helps developers test, debug, and interact with REST, GraphQL, and gRPC APIs. It offers a user-friendly interface for API requests, authentication management, and workflow organization. Its simplicity and support for various authentication methods like OAuth, API keys, and JWT tokens make it popular for API development and testing.

Using Postman Instead of Insomnia
If you are familiar with Postman, it is an alternative option to Insomnia for this lab. Both tools allow users to send API requests, inspect responses, and debug APIs, though their interfaces and some features may differ. If you opt for Postman, make sure to configure authentication, headers, and request bodies as specified for this lab.

Note: For this lab, Insomnia will be used exclusively.

1. To download and install Insomnia, visit Download - Insomnia.


Download Insomnia

Start building, designing, testing better APIs through spec-first development driven by an APIOps CI/CD pipelines.

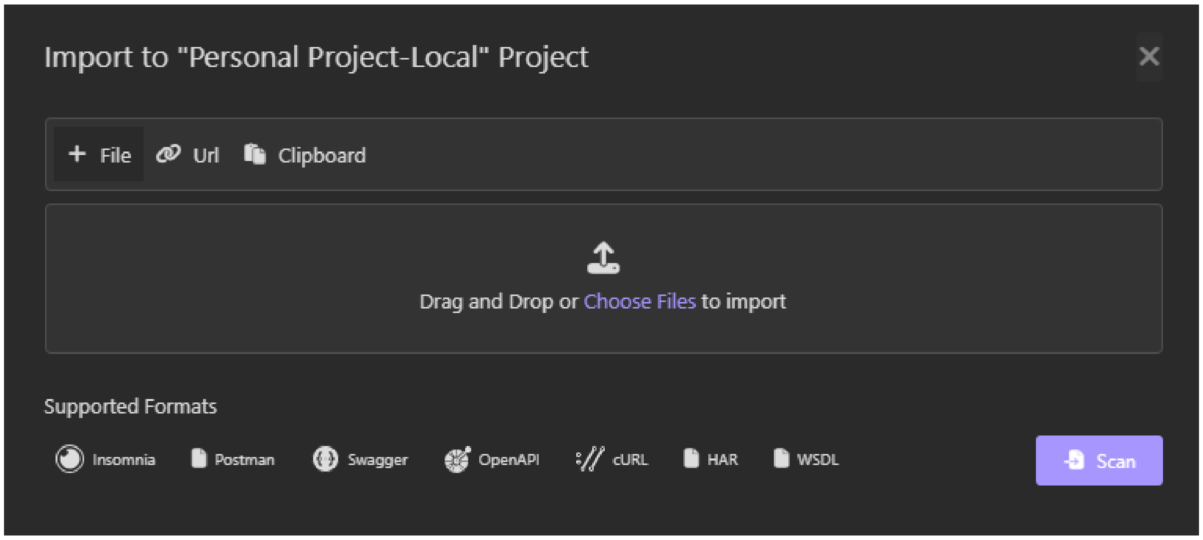
By downloading and using Insomnia, I agree to the Privacy Policy and Terms.

What's New? Changelog

Not your OS? Download for MacOS / Ubuntu or See all downloads.

Exercise 2: Import the API Request Collections

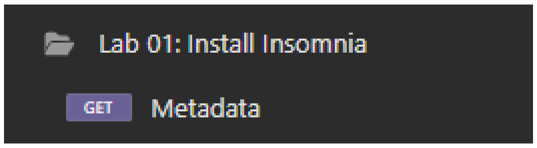
- Navigate and download example YAML file ("*Headless Commerce Lab - Insomnia collection.yaml*")
- Access your personal workspace (It is recommended to signup) and select 'Import'.
- Open the collections.
- Verify that the Request collections and Environment variables are visible.


Import to "Personal Project-Local" Project
+ File
Url
Clipboard
Drag and Drop or Choose Files to import
Supported Formats
Insomnia
Postman
Swagger
OpenAPI
cURL
HAR
WSDL
Scan

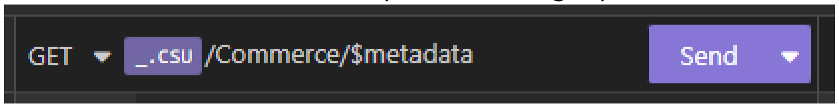
Exercise 3: Verify the Imported Collection

Ensure the environment variables (Base environment on the left corner in insomnia) are correctly updated, including CSU URLs, operating unit numbers, and channel references etc.

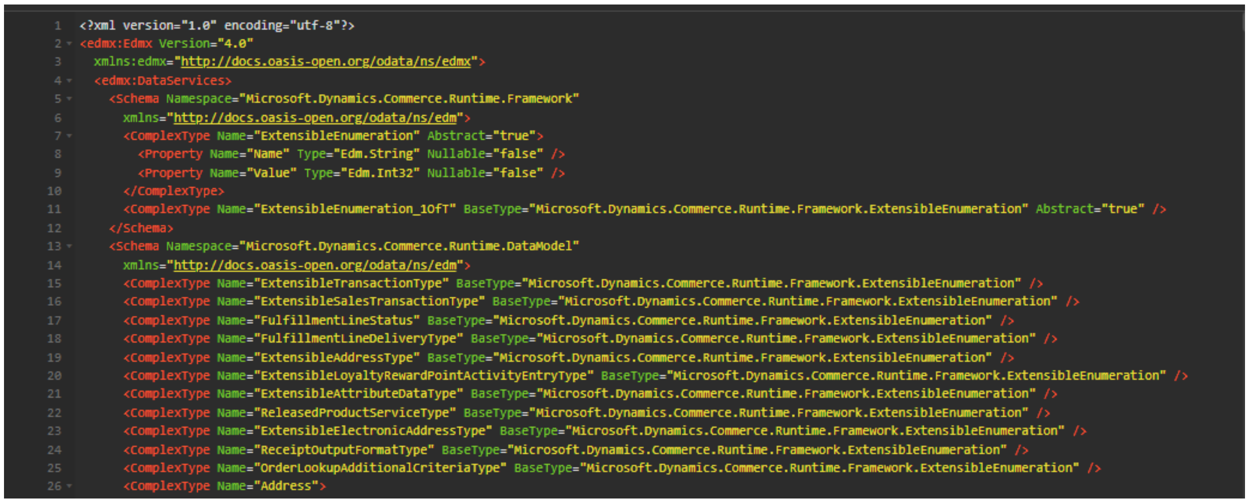
- Go to Base environment and update the CSU URL
- Select "Metadata"



- Click on “Send" and check the response in the right panel



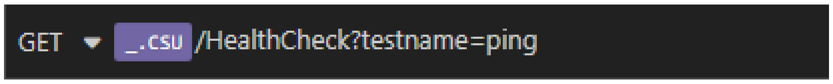
Pro tip: Open the Retail Server URL below in a web browser to view all Retail Server APIs and their input and output parameters: https://RS-URL/Commerce/$metadata



```
<?xml version="1.0" encoding="utf-8"?>
<edmx:Edmx Version="4.0"
  xmlns:edmx="http://docs.oasis-open.org/odata/ns/edmx">
  <edmx:DataServices>
    <Schema Namespace="Microsoft.Dynamics.Commerce.Runtime.Framework"
      xmlns="http://docs.oasis-open.org/odata/ns/edm">
      <ComplexType Name="ExtensibleEnumeration" Abstract="true">
        <Property Name="Name" Type="Edm.String" Nullable="false" />
        <Property Name="Value" Type="Edm.Int32" Nullable="false" />
      </ComplexType>
      <ComplexType Name="ExtensibleEnumeration_1OfT" BaseType="Microsoft.Dynamics.Commerce.Runtime.Framework.ExtensibleEnumeration" Abstract="true" />
    </Schema>
    <Schema Namespace="Microsoft.Dynamics.Commerce.Runtime.DataModel"
      xmlns="http://docs.oasis-open.org/odata/ns/edm">
      <ComplexType Name="ExtensibleTransactionType" BaseType="Microsoft.Dynamics.Commerce.Runtime.Framework.ExtensibleEnumeration" />
      <ComplexType Name="ExtensibleSalesTransactionType" BaseType="Microsoft.Dynamics.Commerce.Runtime.Framework.ExtensibleEnumeration" />
      <ComplexType Name="FulfillmentLineStatus" BaseType="Microsoft.Dynamics.Commerce.Runtime.Framework.ExtensibleEnumeration" />
      <ComplexType Name="FulfillmentLineDeliveryType" BaseType="Microsoft.Dynamics.Commerce.Runtime.Framework.ExtensibleEnumeration" />
      <ComplexType Name="ExtensibleAddressType" BaseType="Microsoft.Dynamics.Commerce.Runtime.Framework.ExtensibleEnumeration" />
      <ComplexType Name="ExtensibleLoyaltyRewardPointActivityEntryType" BaseType="Microsoft.Dynamics.Commerce.Runtime.Framework.ExtensibleEnumeration" />
      <ComplexType Name="ExtensibleAttributeDataType" BaseType="Microsoft.Dynamics.Commerce.Runtime.Framework.ExtensibleEnumeration" />
      <ComplexType Name="ReleasedProductServiceType" BaseType="Microsoft.Dynamics.Commerce.Runtime.Framework.ExtensibleEnumeration" />
      <ComplexType Name="ExtensibleElectronicAddressType" BaseType="Microsoft.Dynamics.Commerce.Runtime.Framework.ExtensibleEnumeration" />
      <ComplexType Name="ReceiptOutputFormatType" BaseType="Microsoft.Dynamics.Commerce.Runtime.Framework.ExtensibleEnumeration" />
      <ComplexType Name="OrderLookupAdditionalCriteriaType" BaseType="Microsoft.Dynamics.Commerce.Runtime.Framework.ExtensibleEnumeration" />
      <ComplexType Name="Address">
```

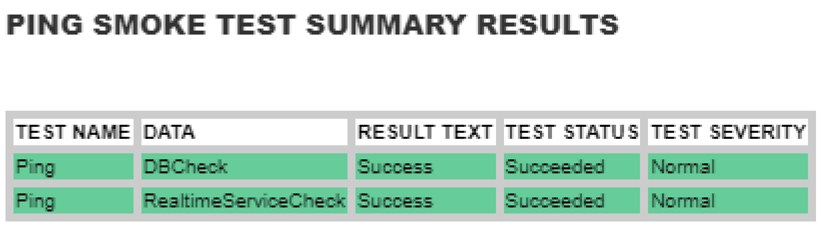
Exercise 4: Verify Commerce Scale Unit Health

1. Click on Health check request and then click on 'send' button.



```
GET _.csu /HealthCheck?testname=ping
```

Response:



PING SMOKE TEST SUMMARY RESULTS

TEST NAME	DATA	RESULT TEXT	TEST STATUS	TEST SEVERITY
Ping	DBCheck	Success	Succeeded	Normal
Ping	RealtimeServiceCheck	Success	Succeeded	Normal

Lab 02: Anonymous - role

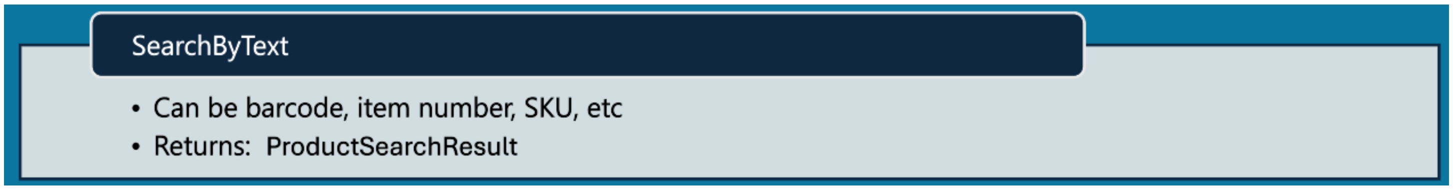
Note: Anonymous access to the CSU is disabled by default
- Self-hosted CSU can enable anonymous access manually
- Cloud-hosted CSUs require a support request

The Anonymous role is used for requests that represent an e-commerce customer who hasn't signed in.

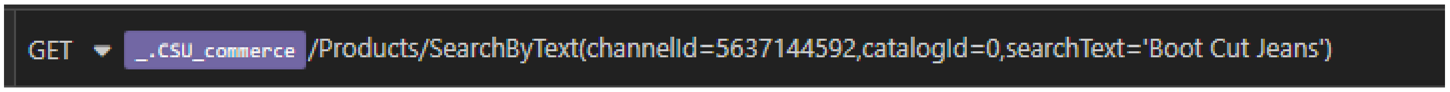
Pro tip: Check the documentation for a list of APIs. The 'supported commerce role' shows if an API is accessible to anonymous users. Commerce Scale Unit customer and consumer APIs - Commerce | Dynamics 365 | Microsoft Learn

Exercise 1: SearchByText

"Search by text" allows product searches based on a text string. Results are retrieved from CSU based on SQL search or Azure AI search (formerly Azure Cognitive Search), depending on configurations. Therefore, results may vary if ACS is enabled on the server.

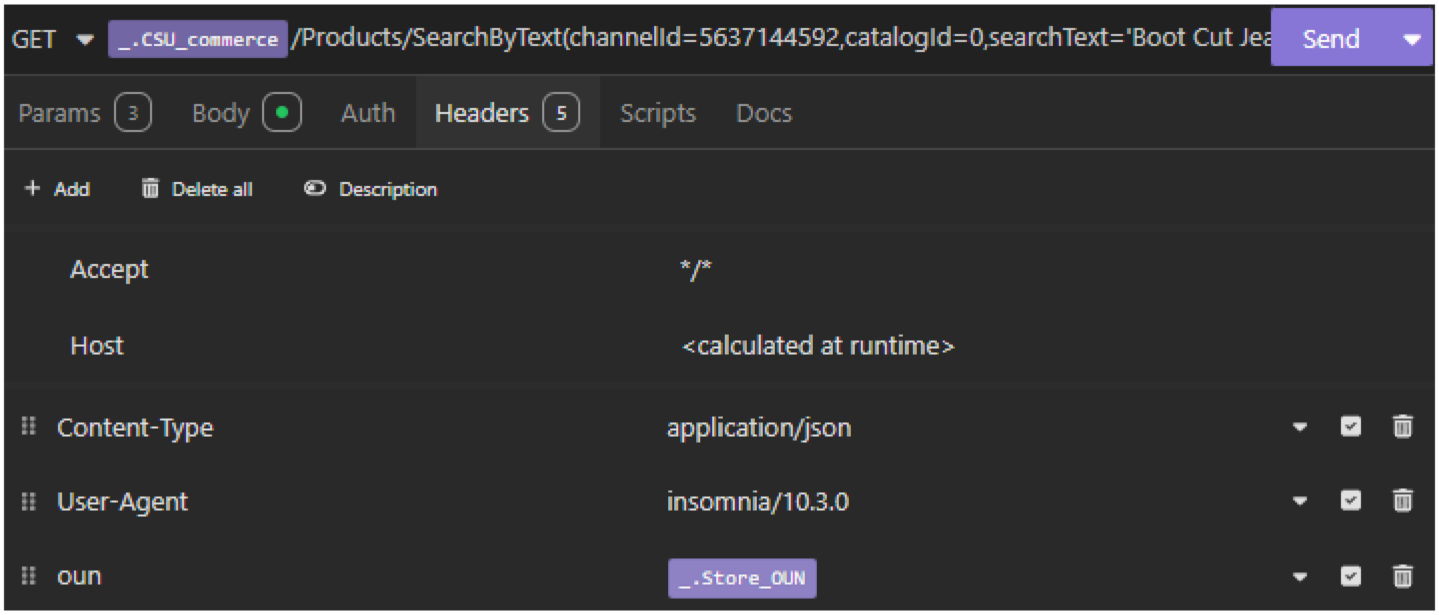

SearchByText
- Can be barcode, item number, SKU, etc
- Returns: ProductSearchResult

- Execute the “SearchByText” request and review the ‘Headers’ tab.


```
GET _.csu_commerce/Products/SearchByText(channelId=5637144592,catalogId=0,searchText='Boot Cut Jeans')
```

Pro tip: When "Anonymous" access is enabled on CSU, any user can use the "Search text" API to get the product list without an Authorization token.

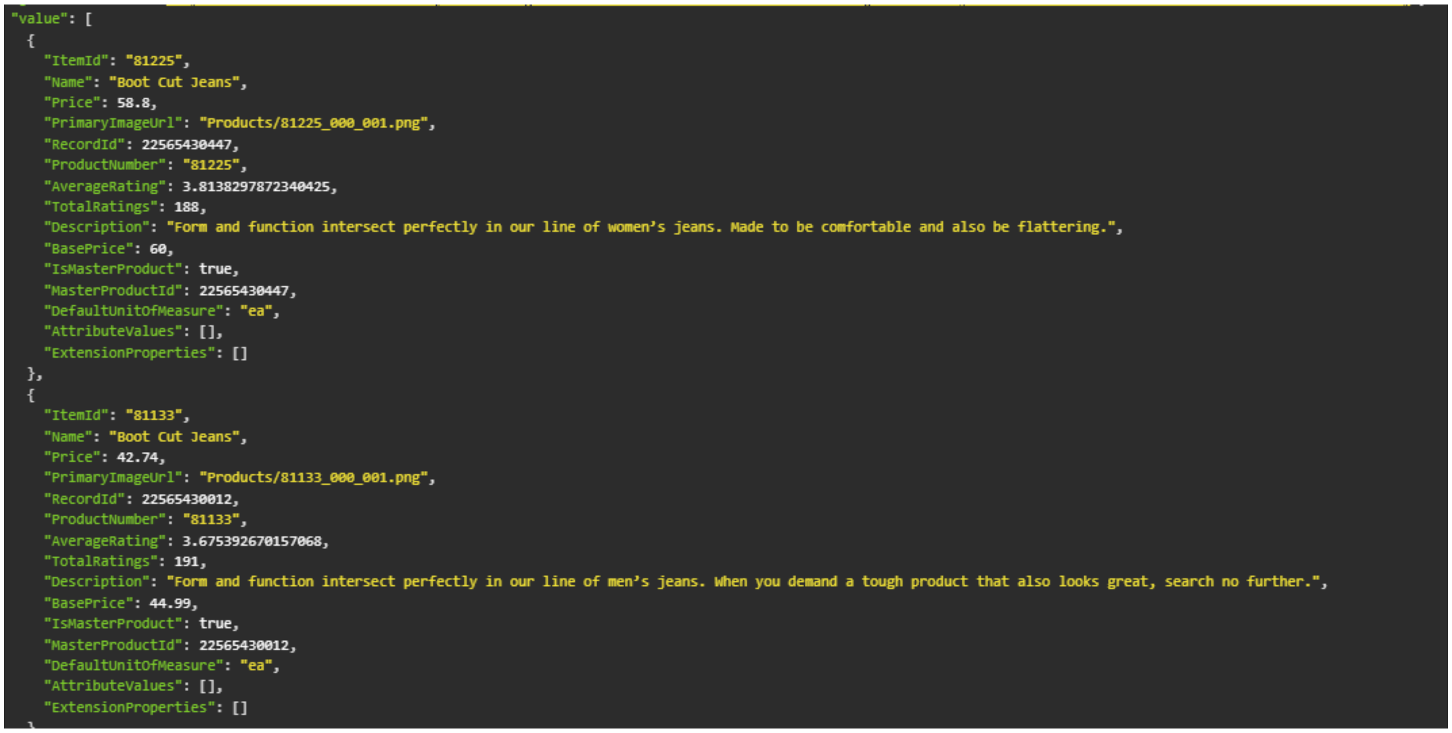
Pro tip: It is important to pass the OUN (Operating Unit Number) in all request headers while accessing the headless commerce APIs. Headless commerce APIs operate within an OUN context, meaning each API is expected to have either a brick-and-mortar store or an online store in the background.



```
GET _.CSU_commerce /Products/SearchByText(channelId=5637144592,catalogId=0,searchText='Boot Cut Jea
```

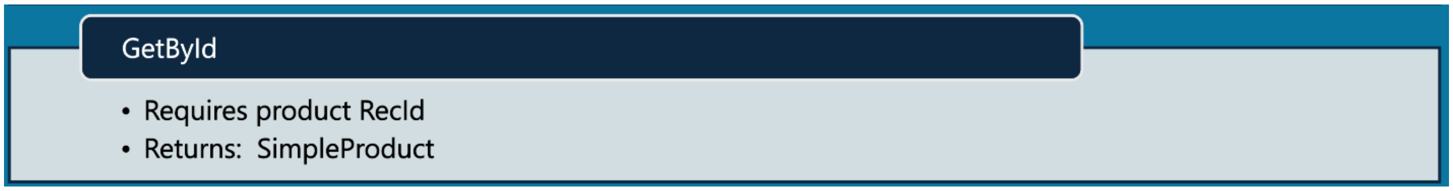
Accept	*/*
Host	<calculated at runtime>
Content-Type	application/json
User-Agent	insomnia/10.3.0
oun	_.Store_OUN

Product list retrieved from CSU



```
"value": [
  {
    "ItemId": "81225",
    "Name": "Boot Cut Jeans",
    "Price": 58.8,
    "PrimaryImageUrl": "Products/81225_000_001.png",
    "RecordId": 22565430447,
    "ProductNumber": "81225",
    "AverageRating": 3.8138297872340425,
    "TotalRatings": 188,
    "Description": "Form and function intersect perfectly in our line of women’s jeans. Made to be comfortable and also be flattering.",
    "BasePrice": 60,
    "IsMasterProduct": true,
    "MasterProductId": 22565430447,
    "DefaultUnitOfMeasure": "ea",
    "AttributeValues": [],
    "ExtensionProperties": []
  },
  {
    "ItemId": "81133",
    "Name": "Boot Cut Jeans",
    "Price": 42.74,
    "PrimaryImageUrl": "Products/81133_000_001.png",
    "RecordId": 22565430012,
    "ProductNumber": "81133",
    "AverageRating": 3.675392670157068,
    "TotalRatings": 191,
    "Description": "Form and function intersect perfectly in our line of men’s jeans. When you demand a tough product that also looks great, search no further.",
    "BasePrice": 44.99,
    "IsMasterProduct": true,
    "MasterProductId": 22565430012,
    "DefaultUnitOfMeasure": "ea",
    "AttributeValues": [],
    "ExtensionProperties": []
```

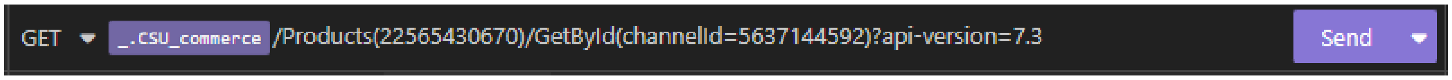
Exercise 2: GetById



GetById

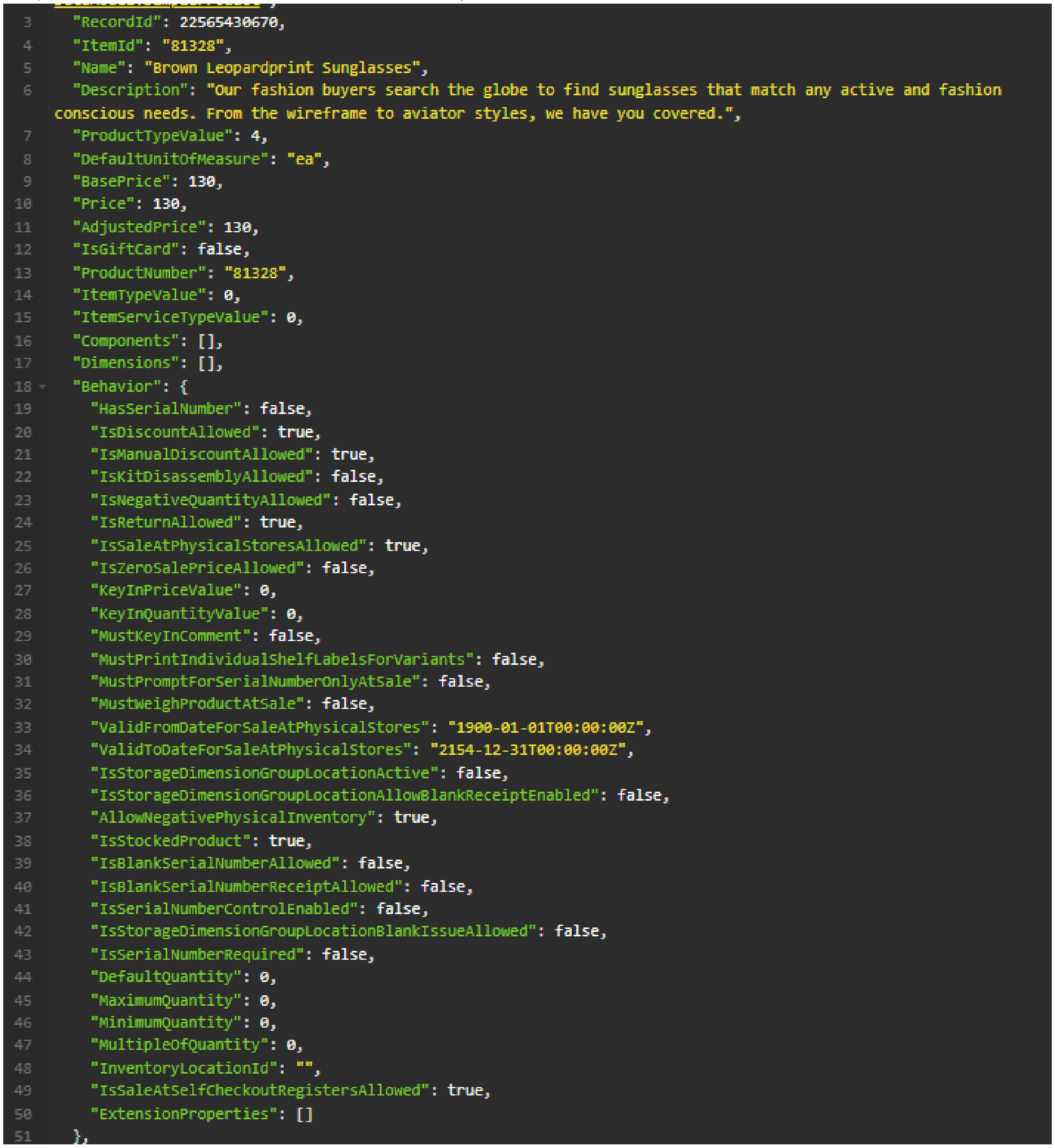
- Requires product RecId
- Returns: SimpleProduct

- Execute the “GetById” request.



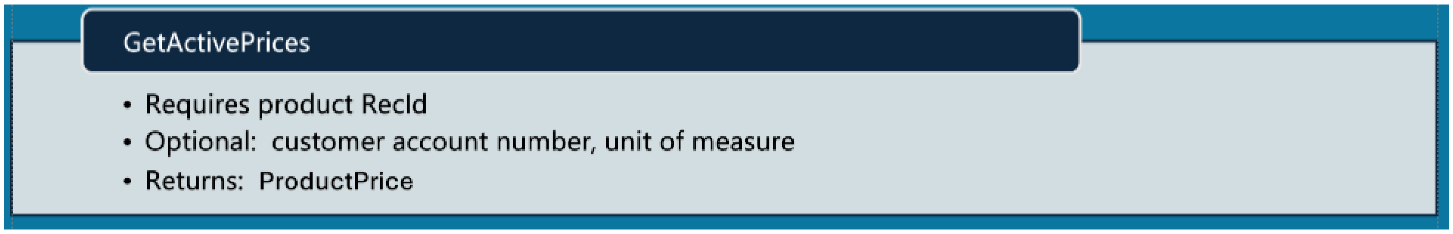
```
GET _.CSU_commerce /Products(22565430670)/GetById(channelId=5637144592)?api-version=7.3
```

Response: *Product details retrieved based on the product id.*



```
  "RecordId": 22565430670,
  "ItemId": "81328",
  "Name": "Brown Leopardprint Sunglasses",
  "Description": "Our fashion buyers search the globe to find sunglasses that match any active and fashion
conscious needs. From the wireframe to aviator styles, we have you covered.",
  "ProductTypeValue": 4,
  "DefaultUnitOfMeasure": "ea",
  "BasePrice": 130,
  "Price": 130,
  "AdjustedPrice": 130,
  "IsGiftCard": false,
  "ProductNumber": "81328",
  "ItemTypeValue": 0,
  "ItemServiceTypeValue": 0,
  "Components": [],
  "Dimensions": [],
  "Behavior": {
    "HasSerialNumber": false,
    "IsDiscountAllowed": true,
    "IsManualDiscountAllowed": true,
    "IsKitDisassemblyAllowed": false,
    "IsNegativeQuantityAllowed": false,
    "IsReturnAllowed": true,
    "IsSaleAtPhysicalStoresAllowed": true,
    "IsZeroSalePriceAllowed": false,
    "KeyInPriceValue": 0,
    "KeyInQuantityValue": 0,
    "MustKeyInComment": false,
    "MustPrintIndividualShelfLabelsForVariants": false,
    "MustPromptForSerialNumberOnlyAtSale": false,
    "MustWeighProductAtSale": false,
    "ValidFromDateForSaleAtPhysicalStores": "1900-01-01T00:00:00Z",
    "ValidToDateForSaleAtPhysicalStores": "2154-12-31T00:00:00Z",
    "IsStorageDimensionGroupLocationActive": false,
    "IsStorageDimensionGroupLocationAllowBlankReceiptEnabled": false,
    "AllowNegativePhysicalInventory": true,
    "IsStockedProduct": true,
    "IsBlankSerialNumberAllowed": false,
    "IsBlankSerialNumberReceiptAllowed": false,
    "IsSerialNumberControlEnabled": false,
    "IsStorageDimensionGroupLocationBlankIssueAllowed": false,
    "IsSerialNumberRequired": false,
    "DefaultQuantity": 0,
    "MaximumQuantity": 0,
    "MinimumQuantity": 0,
    "MultipleOfQuantity": 0,
    "InventoryLocationId": "",
    "IsSaleAtSelfCheckoutRegistersAllowed": true,
    "ExtensionProperties": []
  },
```

Exercise 3: GetActivePrices



GetActivePrices

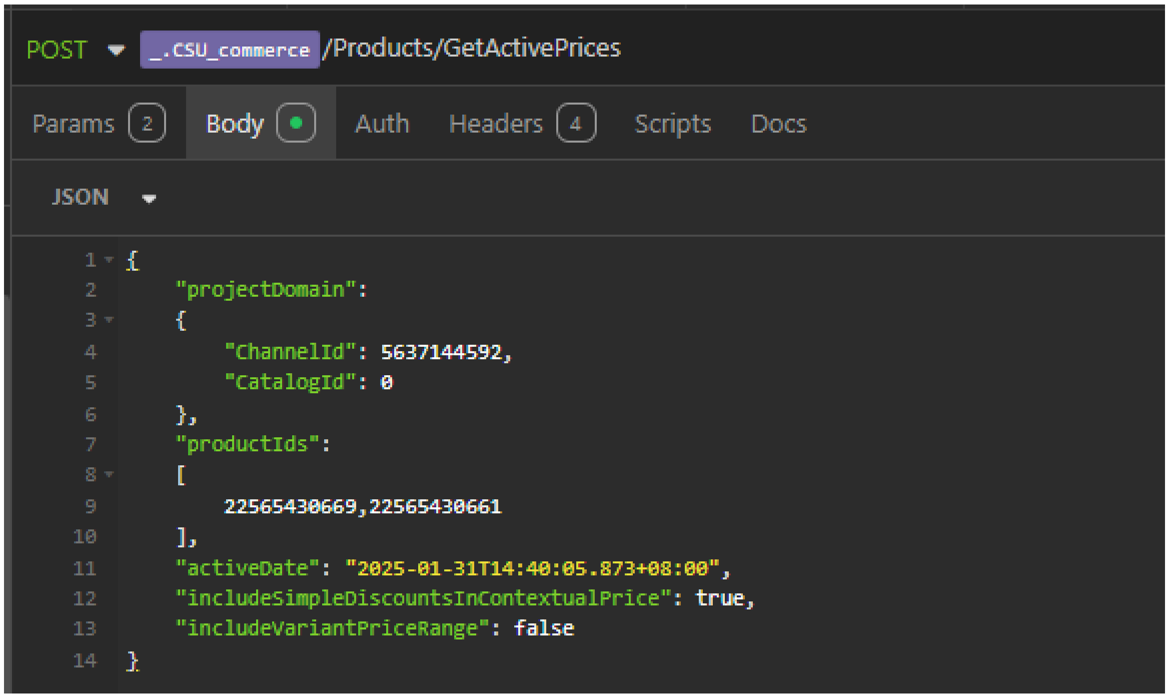
- Requires product RecId
- Optional: customer account number, unit of measure
- Returns: ProductPrice

Pro tip:
The Dynamics 365 Commerce pricing engine exposes the following Retail Server APIs for external applications to support diverse pricing scenarios:

- **GetActivePrices** – Retrieves the calculated price of a product, including simple discounts.
- **CalculateSalesDocument** – Computes prices and discounts based on product quantities and combinations. For e.g. to calculate the Mix and Match discount
- **GetAvailablePromotions** – Returns applicable discounts for products in the cart.
- **AddCoupons** – This API adds coupons to a cart.
- **RemoveCoupons** – This API removes coupons from a cart.

Reference: Commerce pricing APIs - Commerce | Dynamics 365 | Microsoft Learn

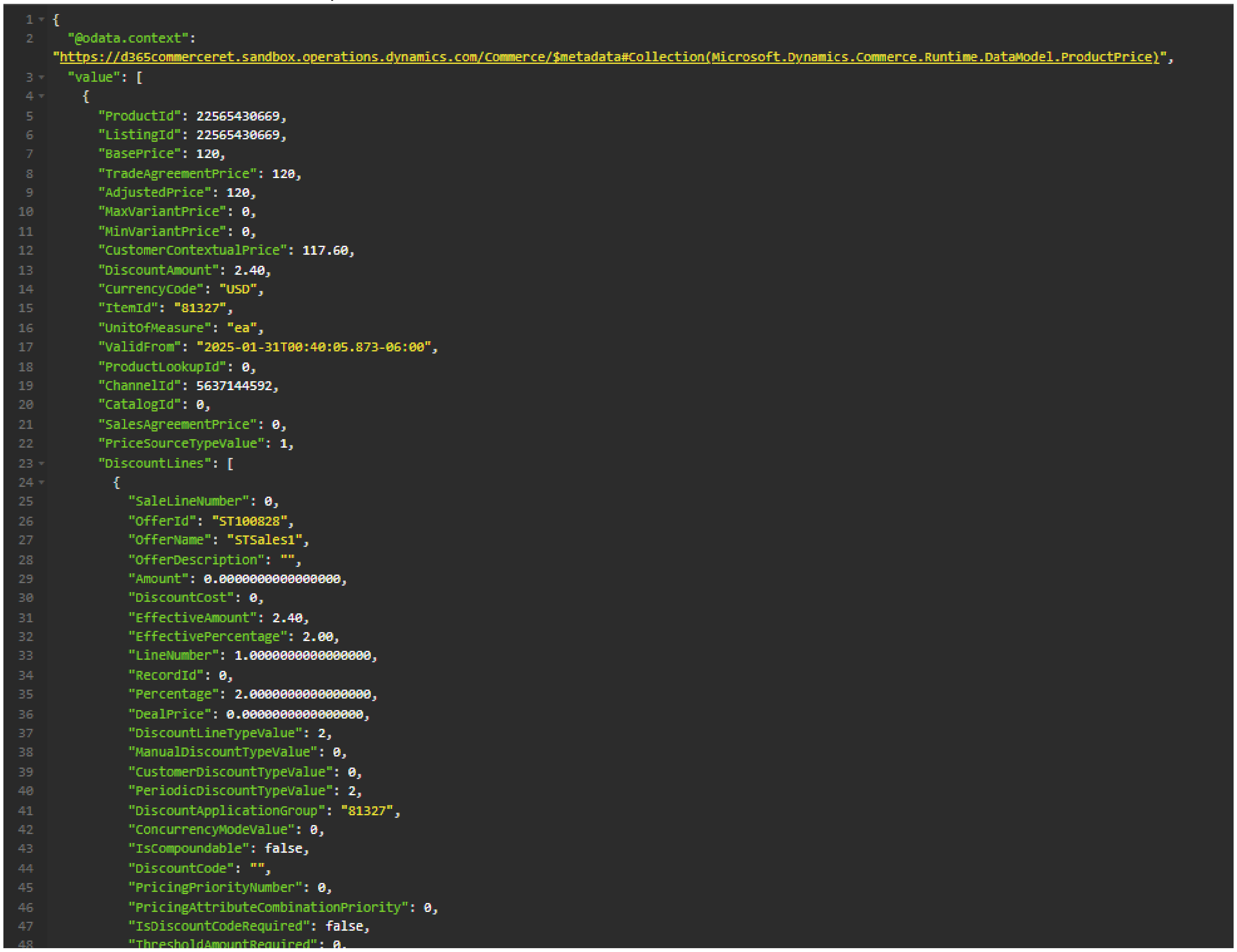
Execute the "GetActivePrices" request



```
/Products/GetActivePrices
```

```
{
    "projectDomain":
    {
        "ChannelId": 5637144592,
        "CatalogId": 0
    },
    "productIds":
    [
        22565430669,22565430661
    ],
    "activeDate": "2025-01-31T14:40:05.873+08:00",
    "includeSimpleDiscountsInContextualPrice": true,
    "includeVariantPriceRange": false
}
```

GetActivePrices retrieves current prices and discounts.



```
{
  "@odata.context":
"https://d365commerceret.sandbox.operations.dynamics.com/Commerce/$metadata#Collection(Microsoft.Dynamics.Commerce.Runtime.DataModel.ProductPrice)",
  "value": [
    {
      "ProductId": 22565430669,
      "ListingId": 22565430669,
      "BasePrice": 120,
      "TradeAgreementPrice": 120,
      "AdjustedPrice": 120,
      "MaxVariantPrice": 0,
      "MinVariantPrice": 0,
      "CustomerContextualPrice": 117.60,
      "DiscountAmount": 2.40,
      "CurrencyCode": "USD",
      "ItemId": "81327",
      "UnitOfMeasure": "ea",
      "ValidFrom": "2025-01-31T00:40:05.873-06:00",
      "ProductLookupId": 0,
      "ChannelId": 5637144592,
      "CatalogId": 0,
      "SalesAgreementPrice": 0,
      "PriceSourceTypeValue": 1,
      "DiscountLines": [
        {
          "SaleLineNumber": 0,
          "OfferId": "ST100828",
          "OfferName": "STSales1",
          "OfferDescription": "",
          "Amount": 0.0000000000000000,
          "DiscountCost": 0,
          "EffectiveAmount": 2.40,
          "EffectivePercentage": 2.00,
          "LineNumber": 1.0000000000000000,
          "RecordId": 0,
          "Percentage": 2.0000000000000000,
          "DealPrice": 0.0000000000000000,
          "DiscountLineTypeValue": 2,
          "ManualDiscountTypeValue": 0,
          "CustomerDiscountTypeValue": 0,
          "PeriodicDiscountTypeValue": 2,
          "DiscountApplicationGroup": "81327",
          "ConcurrencyModeValue": 0,
          "IsCompoundable": false,
          "DiscountCode": "",
          "PricingPriorityNumber": 0,
          "PricingAttributeCombinationPriority": 0,
          "IsDiscountCodeRequired": false,
          "ThresholdAmountRequired": 0,
```

Lab 03: Application - role

> Access to APIs that are associated with this role requires application-level authentication, such as Microsoft Entra service-to-service authentication.

Exercise 1: Retrieve the access token

To access the APIs, external application integration needs only the **Application** authentication flow. Register the external application in Microsoft Entra ID App registration and add its details in Commerce Headquarters before accessing the APIs.

> **Pro tip**: Documentation on setting up the Application registration: [Consume Retail Server APIs in external applications - Commerce | Dynamics 365 | Microsoft Learn]

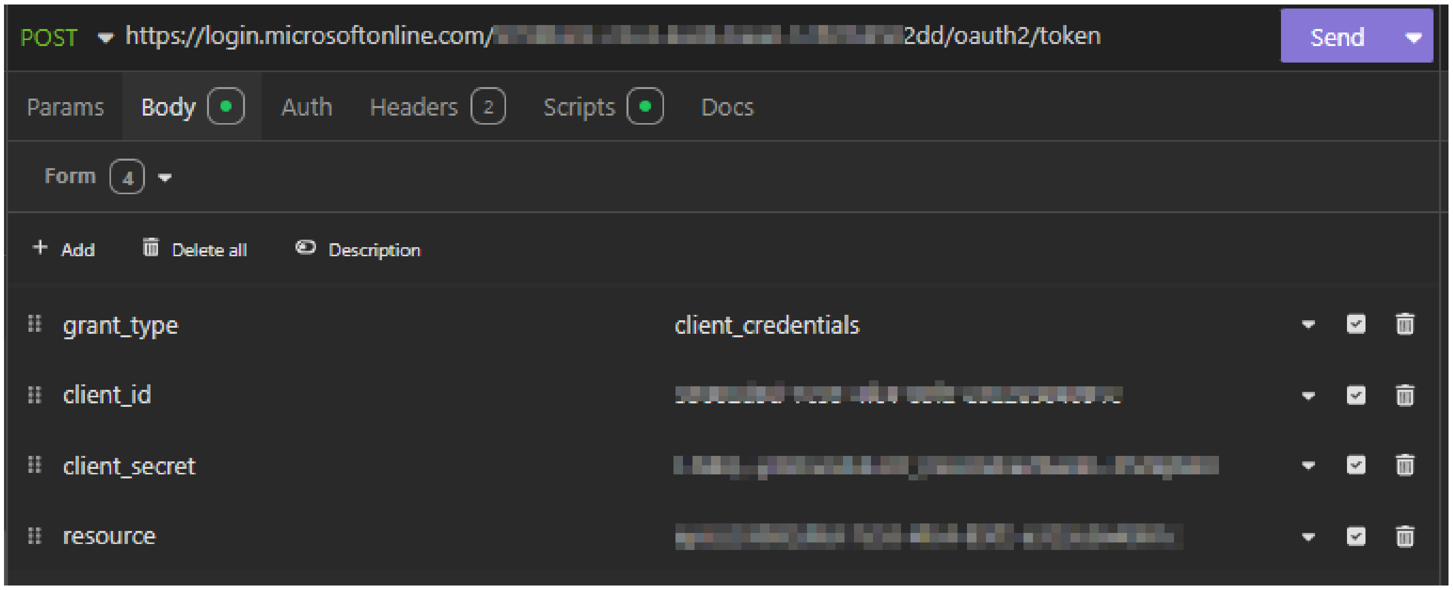
- Execute the "Application token" request

Please ensure that the parameters are updated as instructed (As shown).

> Ensure the correct tenant ID is updated in the URL.

Ensure that the following information is included in the "Body":

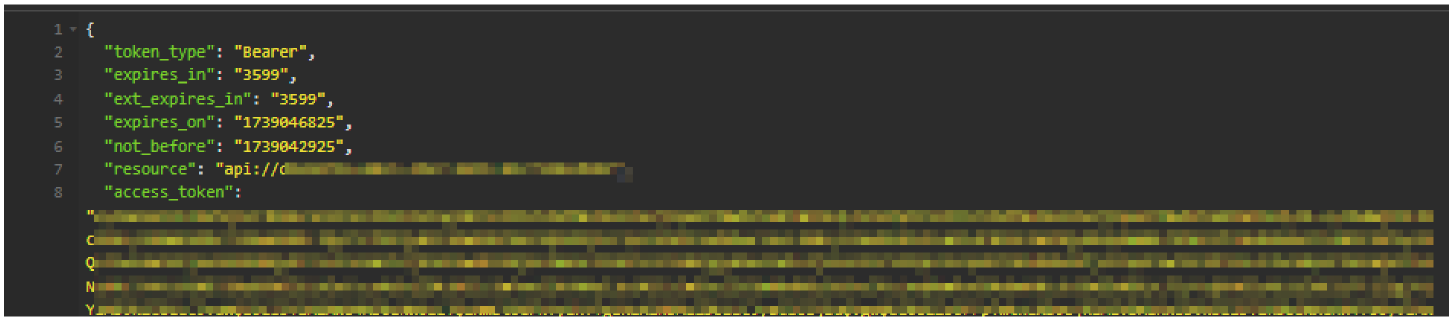
1.	grant_type	<Copy from the secrets file>/ Configure as mentioned in the Docs
2.	client_id	<Copy from the secrets file>/ Configure as mentioned in the Docs
3.	client_secret	<Copy from the secrets file>/ Configure as mentioned in the Docs
4.	resource	<Copy from the secrets file>/ Configure as mentioned in the Docs



POST https://login.microsoftonline.com/[redacted]2dd/oauth2/token

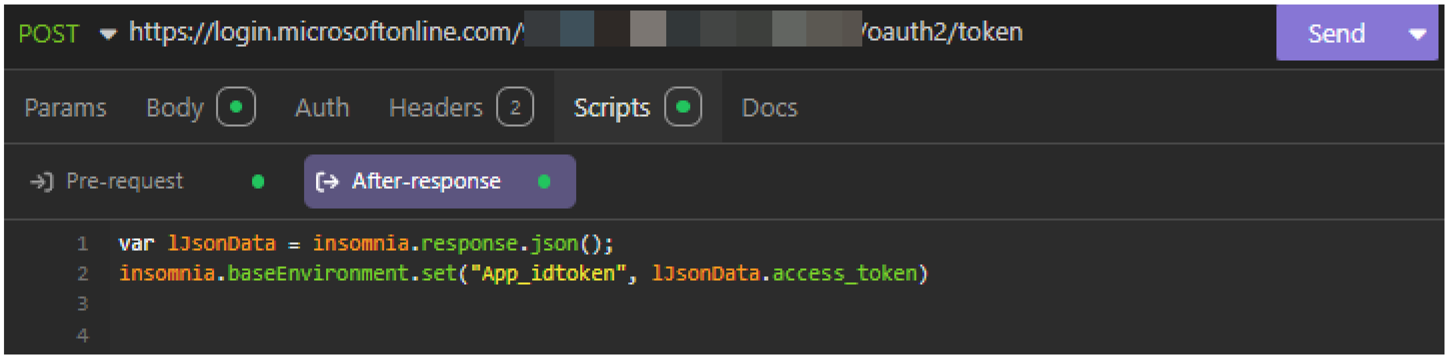
grant_type	client_credentials
client_id	[redacted]
client_secret	[redacted]
resource	[redacted]

This API response contains an access token with an expiration time. This token is used in subsequent exercises.



```
{
  "token_type": "Bearer",
  "expires_in": "3599",
  "ext_expires_in": "3599",
  "expires_on": "1739046825",
  "not_before": "1739042925",
  "resource": "api://d[redacted]
  "access_token":
"[redacted]
```

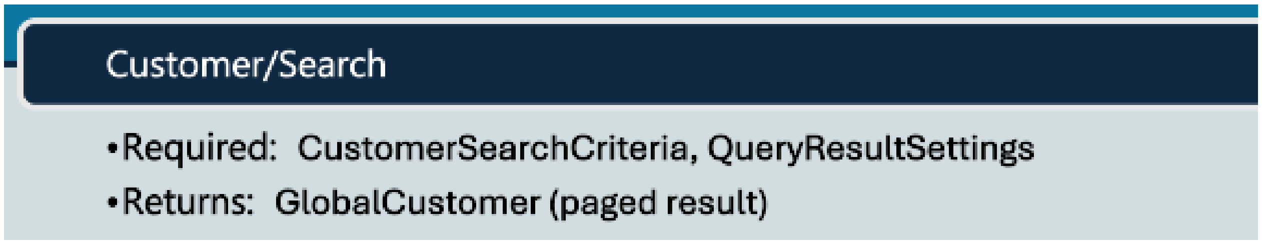
Upon execution of the Authentication API, the access token is updated in the Base environment's variables. This updated token can then be utilized in subsequent API executions.



```
POST https://login.microsoftonline.com/[redacted]/oauth2/token
```

```
var lJsonData = insomnia.response.json();
insomnia.baseEnvironment.set("App_idtoken", lJsonData.access_token)
```

Exercise 2: Search Customer

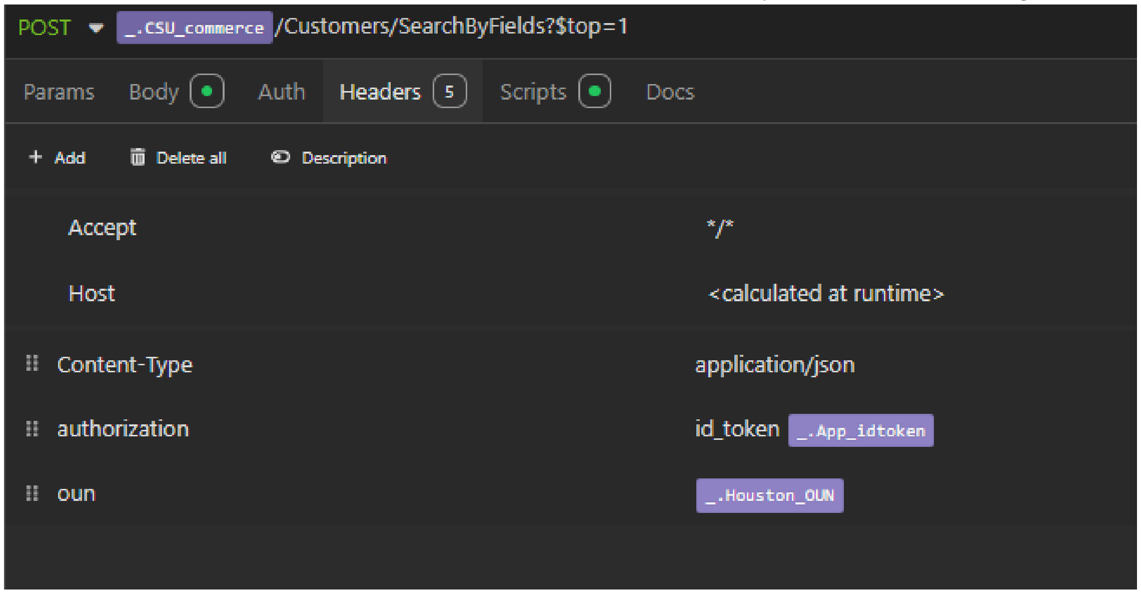


Customer/Search

- Required: CustomerSearchCriteria, QueryResultSettings
- Returns: GlobalCustomer (paged result)

> Documentation indicates that the supported commerce roles for "Search by Fields" are limited to "Employee", "Customer", or "application".
>
> | | | | |
> |---|---|---|---|
> | SearchByFields | Customer entity | PageResult<GlobalCustomer> | Employee, Customer, Application |

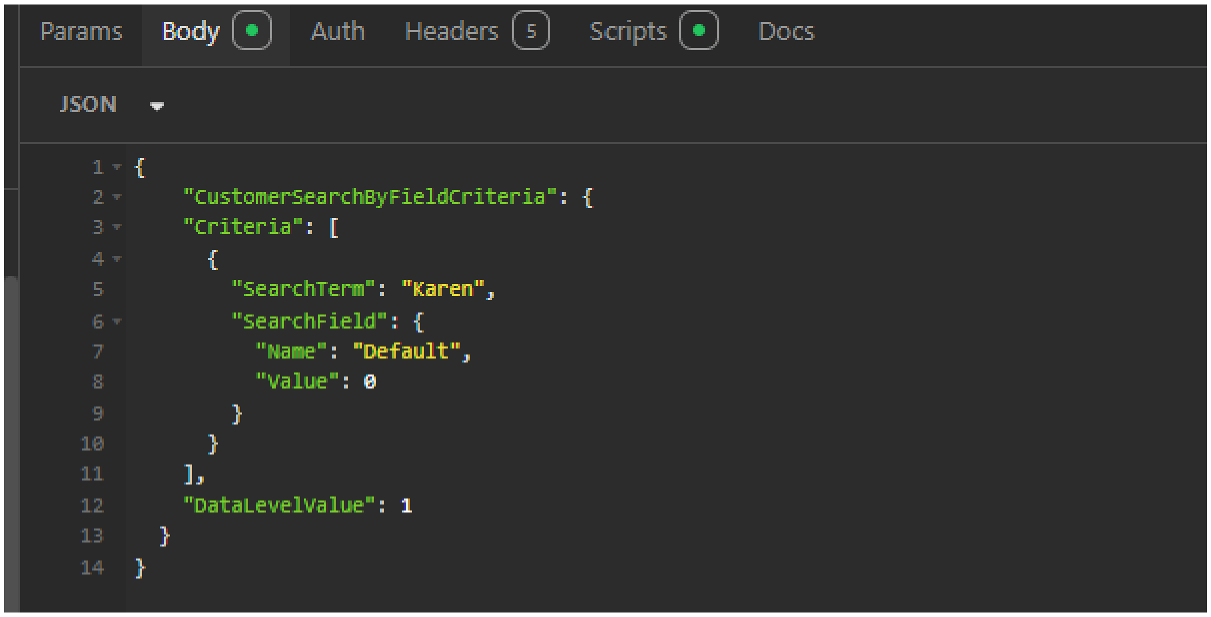
Note: Include the access token in the "authorization" header as specified before making the call.



POST _.CSU_commerce/Customers/SearchByFields?$top=1

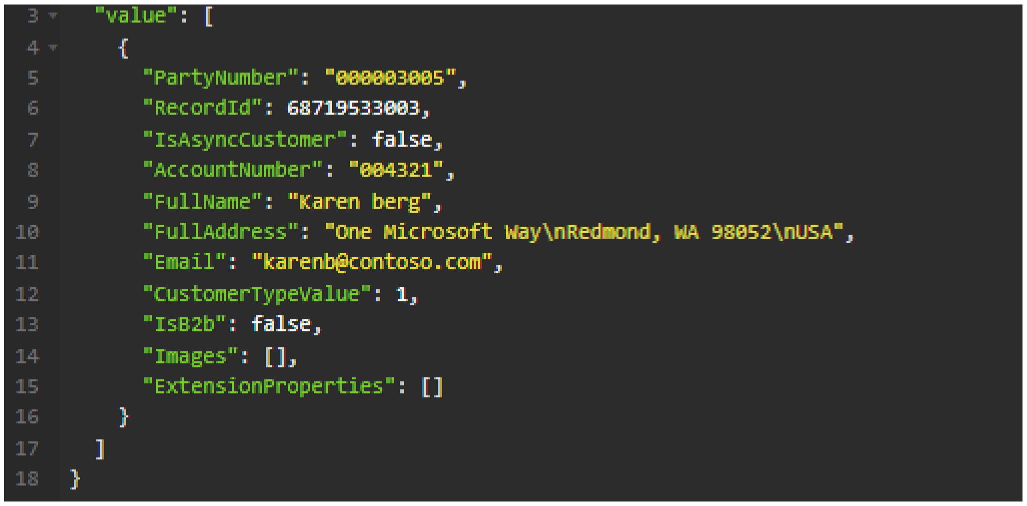
Accept	*/*
Host	<calculated at runtime>
Content-Type	application/json
authorization	id_token _.App_idtoken
oun	_.Houston_OUN

Include the Search term in the Request body.



```
{
    "CustomerSearchByFieldCriteria": {
    "Criteria": [
      {
        "SearchTerm": "Karen",
        "SearchField": {
          "Name": "Default",
          "Value": 0
        }
      }
    ],
    "DataLevelValue": 1
  }
}
```

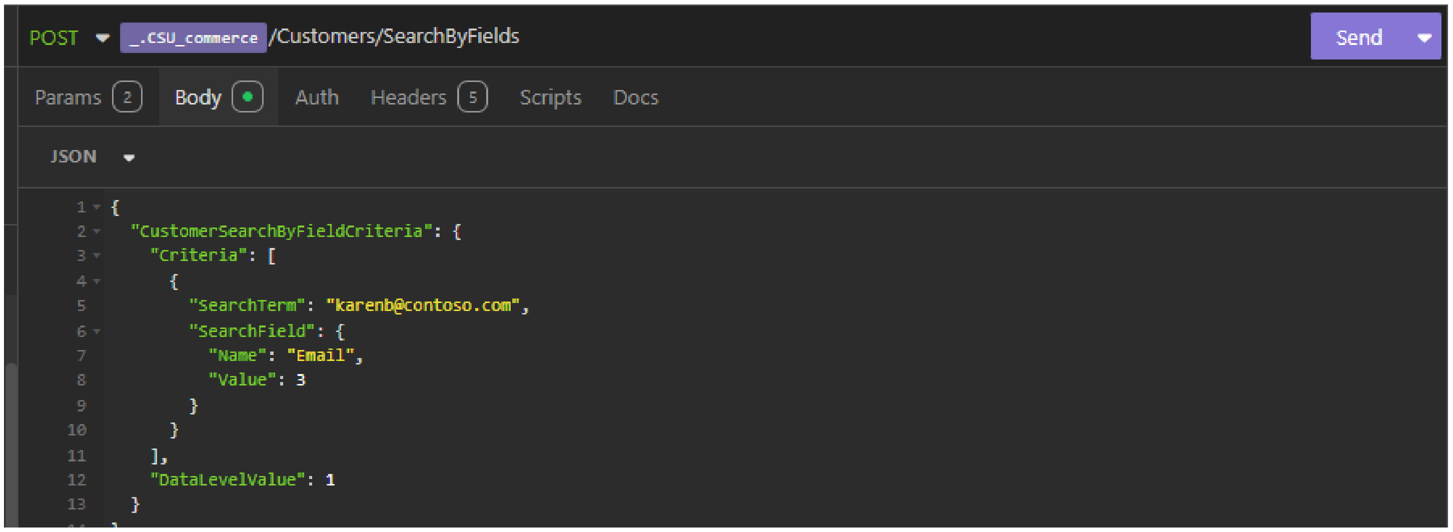
Response: The customer entity has been successfully retrieved.



```
  "value": [
    {
      "PartyNumber": "000003005",
      "RecordId": 68719533003,
      "IsAsyncCustomer": false,
      "AccountNumber": "004321",
      "FullName": "Karen berg",
      "FullAddress": "One Microsoft Way\nRedmond, WA 98052\nUSA",
      "Email": "karenb@contoso.com",
      "CustomerTypeValue": 1,
      "IsB2b": false,
      "Images": [],
      "ExtensionProperties": []
    }
  ]
}
```

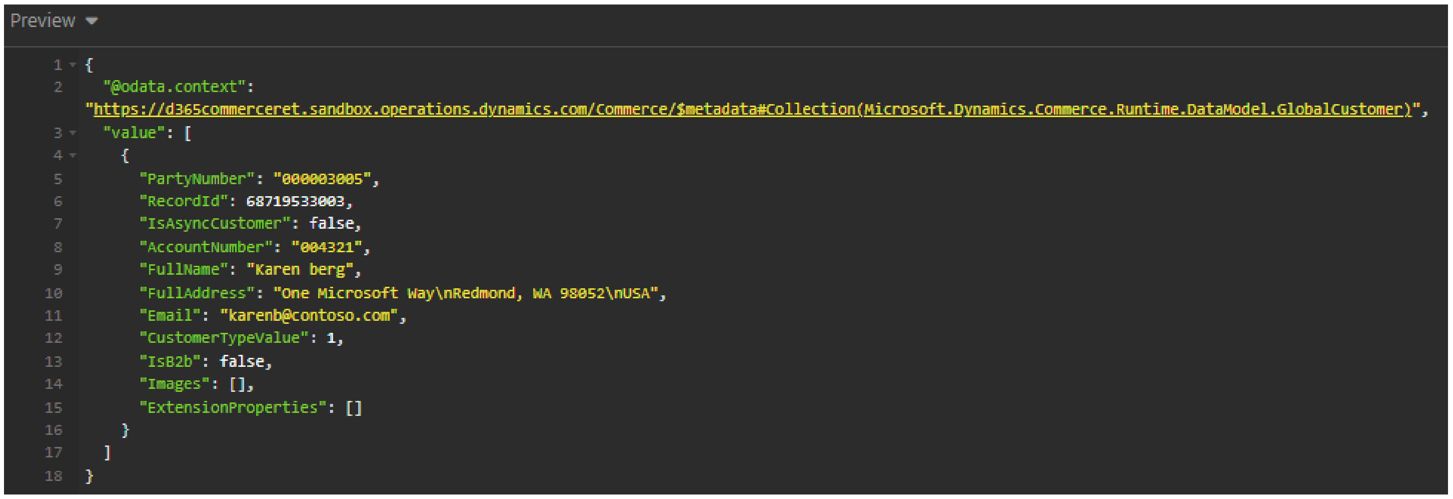
Exercise 3: Get customer by Email

Execute the "Search customer by email" request to retrieve customer details using the Email ID.



```
{
  "CustomerSearchByFieldCriteria": {
    "Criteria": [
      {
        "SearchTerm": "karenb@contoso.com",
        "SearchField": {
          "Name": "Email",
          "Value": 3
        }
      }
    ],
    "DataLevelValue": 1
  }
```

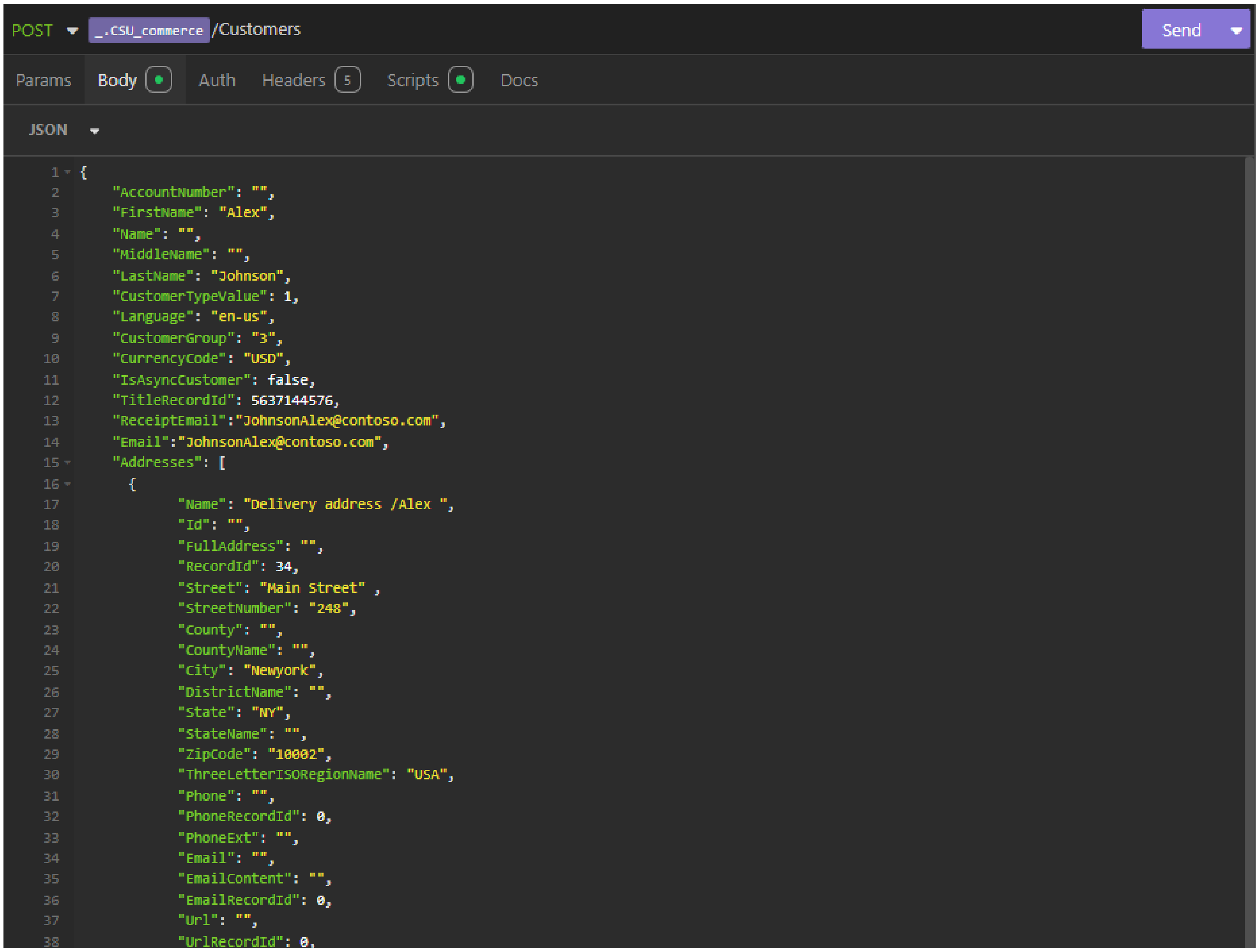
The customer list is retrieved by searching "Email".



```
{
  "@odata.context":
"https://d365commerceret.sandbox.operations.dynamics.com/Commerce/$metadata#Collection(Microsoft.Dynamics.Commerce.Runtime.DataModel.GlobalCustomer)",
  "value": [
    {
      "PartyNumber": "000003005",
      "RecordId": 68719533003,
      "IsAsyncCustomer": false,
      "AccountNumber": "004321",
      "FullName": "Karen berg",
      "FullAddress": "One Microsoft Way\nRedmond, WA 98052\nUSA",
      "Email": "karenb@contoso.com",
      "CustomerTypeValue": 1,
      "IsB2b": false,
      "Images": [],
      "ExtensionProperties": []
    }
  ]
}
```

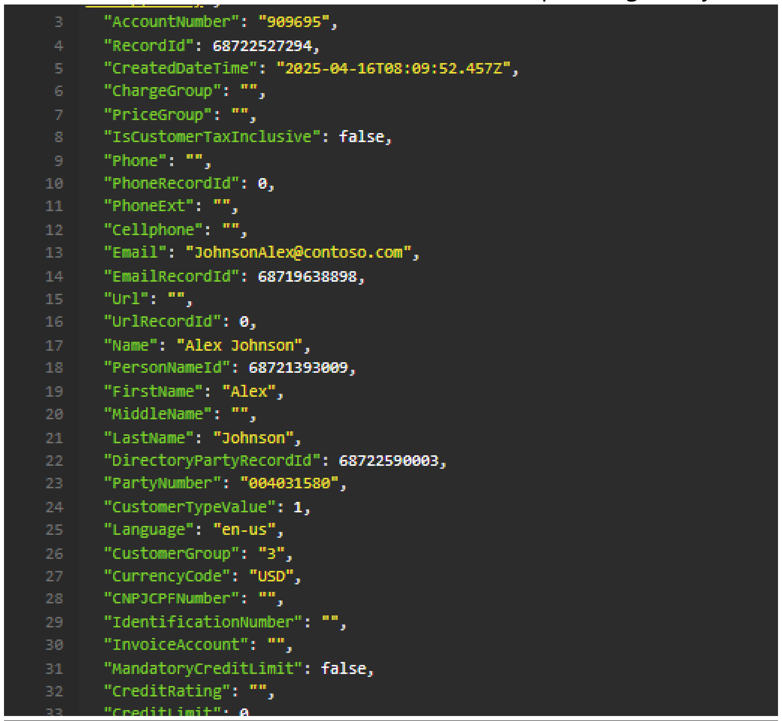
Exercise 4: Create customer

Execute the "Create customer" request to generate a new customer account based on the data provided in the request body, which includes customer and address details.

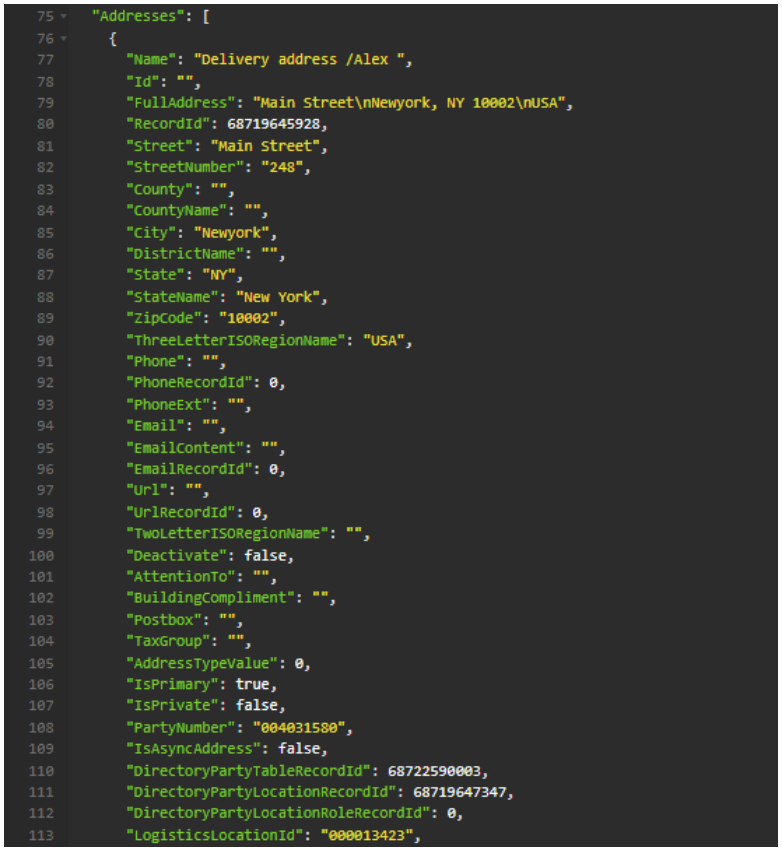


```
{
    "AccountNumber": "",
    "FirstName": "Alex",
    "Name": "",
    "MiddleName": "",
    "LastName": "Johnson",
    "CustomerTypeValue": 1,
    "Language": "en-us",
    "CustomerGroup": "3",
    "CurrencyCode": "USD",
    "IsAsyncCustomer": false,
    "TitleRecordId": 5637144576,
    "ReceiptEmail":"JohnsonAlex@contoso.com",
    "Email":"JohnsonAlex@contoso.com",
    "Addresses": [
      {
            "Name": "Delivery address /Alex ",
            "Id": "",
            "FullAddress": "",
            "RecordId": 34,
            "Street": "Main Street" ,
            "StreetNumber": "248",
            "County": "",
            "CountyName": "",
            "City": "Newyork",
            "DistrictName": "",
            "State": "NY",
            "StateName": "",
            "ZipCode": "10002",
            "ThreeLetterISORegionName": "USA",
            "Phone": "",
            "PhoneRecordId": 0,
            "PhoneExt": "",
            "Email": "",
            "EmailContent": "",
            "EmailRecordId": 0,
            "Url": "",
            "UrlRecordId": 0,
```

A customer account is created, and the corresponding entity is subsequently retrieved in response.



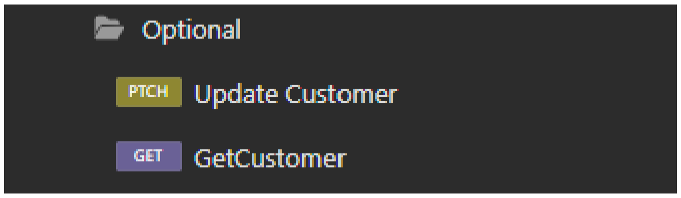
```
  "AccountNumber": "909695",
  "RecordId": 68722527294,
  "CreatedDateTime": "2025-04-16T08:09:52.457Z",
  "ChargeGroup": "",
  "PriceGroup": "",
  "IsCustomerTaxInclusive": false,
  "Phone": "",
  "PhoneRecordId": 0,
  "PhoneExt": "",
  "Cellphone": "",
  "Email": "JohnsonAlex@contoso.com",
  "EmailRecordId": 68719638898,
  "Url": "",
  "UrlRecordId": 0,
  "Name": "Alex Johnson",
  "PersonNameId": 68721393009,
  "FirstName": "Alex",
  "MiddleName": "",
  "LastName": "Johnson",
  "DirectoryPartyRecordId": 68722590003,
  "PartyNumber": "004031580",
  "CustomerTypeValue": 1,
  "Language": "en-us",
  "CustomerGroup": "3",
  "CurrencyCode": "USD",
  "CNPJCPFNumber": "",
  "IdentificationNumber": "",
  "InvoiceAccount": "",
  "MandatoryCreditLimit": false,
  "CreditRating": "",
  "CreditLimit": 0
```



```
"Addresses": [
  {
    "Name": "Delivery address /Alex ",
    "Id": "",
    "FullAddress": "Main Street\nNewyork, NY 10002\nUSA",
    "RecordId": 68719645928,
    "Street": "Main Street",
    "StreetNumber": "248",
    "County": "",
    "CountyName": "",
    "City": "Newyork",
    "DistrictName": "",
    "State": "NY",
    "StateName": "New York",
    "ZipCode": "10002",
    "ThreeLetterISORegionName": "USA",
    "Phone": "",
    "PhoneRecordId": 0,
    "PhoneExt": "",
    "Email": "",
    "EmailContent": "",
    "EmailRecordId": 0,
    "Url": "",
    "UrlRecordId": 0,
    "TwoLetterISORegionName": "",
    "Deactivate": false,
    "AttentionTo": "",
    "BuildingCompliment": "",
    "Postbox": "",
    "TaxGroup": "",
    "AddressTypeValue": 0,
    "IsPrimary": true,
    "IsPrivate": false,
    "PartyNumber": "004031580",
    "IsAsyncAddress": false,
    "DirectoryPartyTableRecordId": 68722590003,
    "DirectoryPartyLocationRecordId": 68719647347,
    "DirectoryPartyLocationRoleRecordId": 0,
    "LogisticsLocationId": "000013423",
```

> Customers are created either in the back office or Commerce Scale unit, depending on the asynchronous customer creation settings. https://learn.microsoft.com/en-us/dynamics365/commerce/async-customer-mode

Note: There are optional exercises: the first is to update customer details, and the second is to retrieve customer details based on the account number.



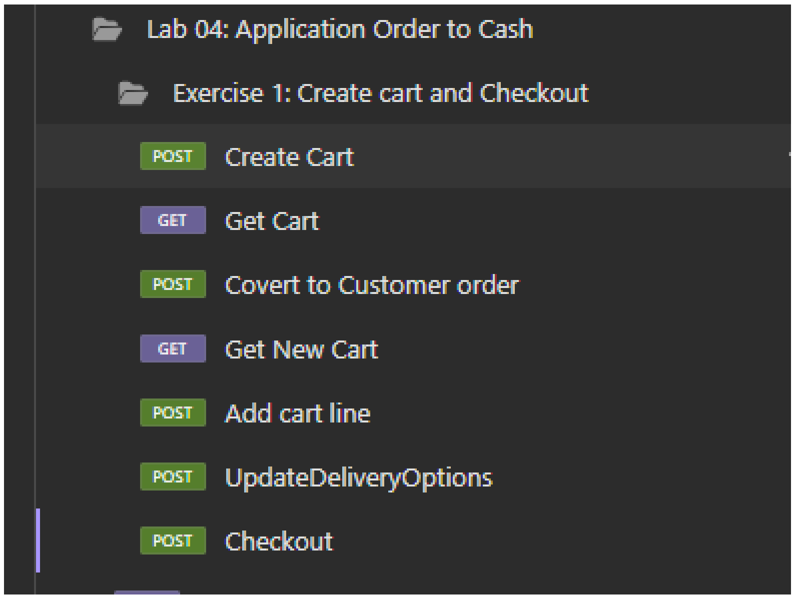
Lab 04: Application - role (Order to Cash)

> Ensure the OUN is set to "Online store" when processing the cart. Running the headless request for brick-and-mortar stores must include shift aspects when creating the cart.

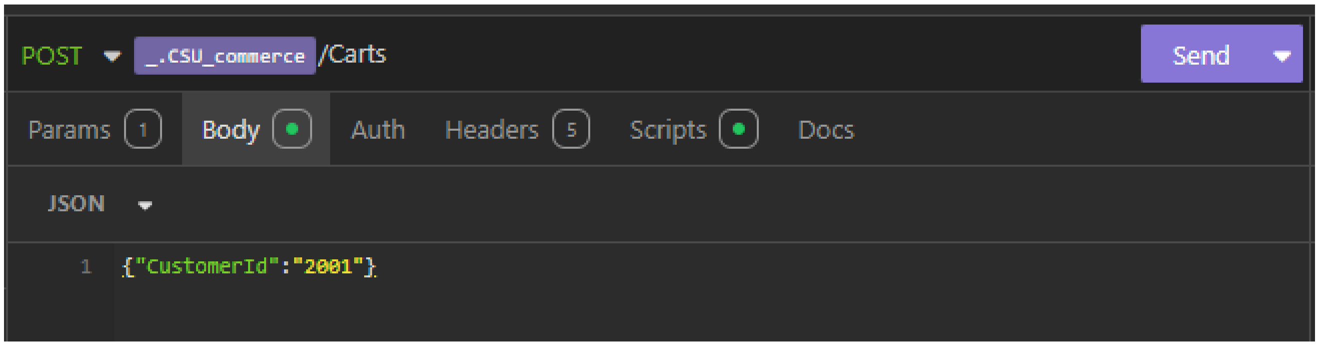
Exercise 1: Create cart and Checkout

This exercise simulates a synchronous checkout using headless commerce. The headless engine utilizes Commerce pricing (or Unified pricing) to calculate price/discounts and promotions, ensuring omni-channel pricing is respected through these calls.

Note: Execute each request to complete the checkout process.

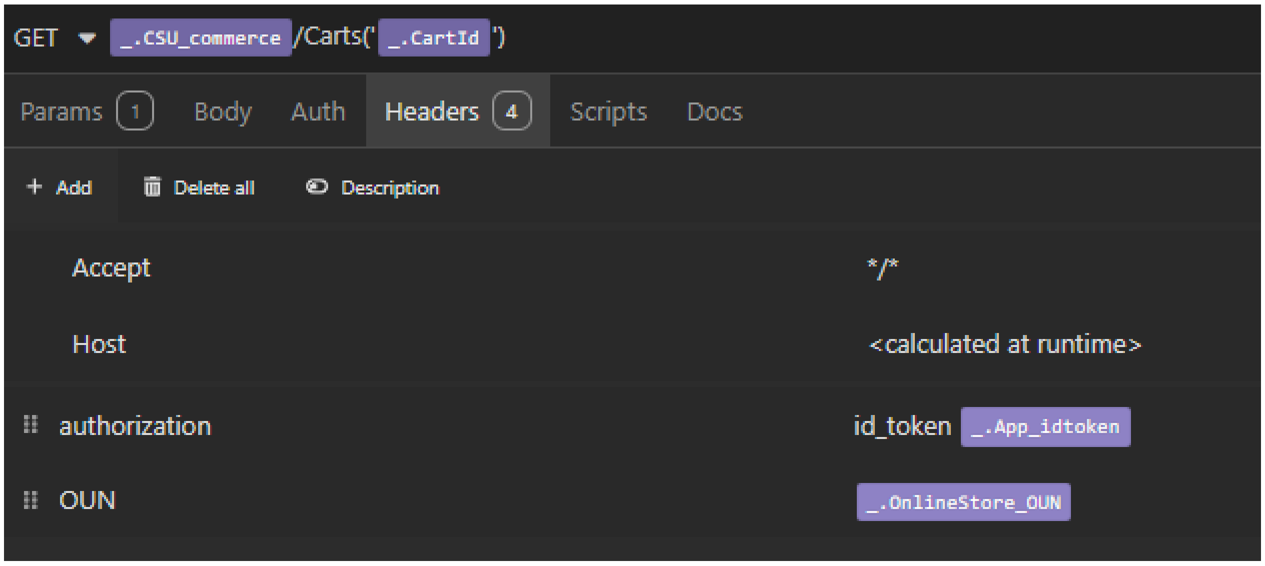

Lab 04: Application Order to Cash
Exercise 1: Create cart and Checkout
POST Create Cart
GET Get Cart
POST Covert to Customer order
GET Get New Cart
POST Add cart line
POST UpdateDeliveryOptions
POST Checkout

1. Create a cart



```
POST _.CSU_commerce/Carts
{"CustomerId":"2001"}
```

2. Get the cart details



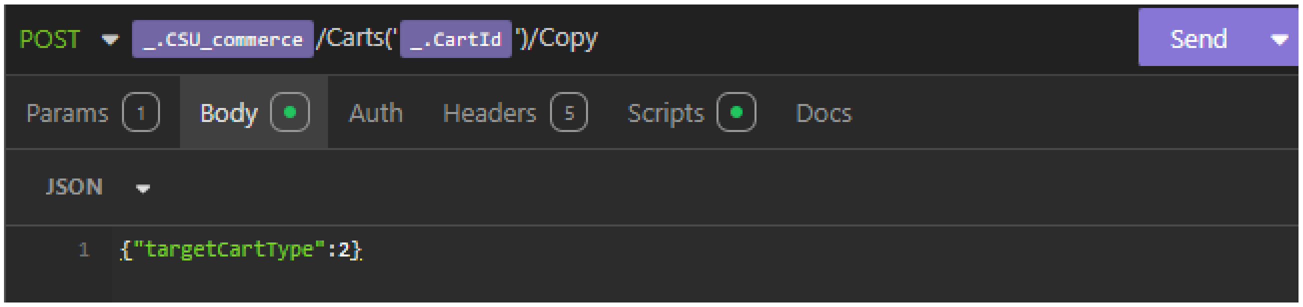
GET _.CSU_commerce/Carts('_.CartId')

Params 1 | Body | Auth | Headers 4 | Scripts | Docs

+ Add | Delete all | Description

Accept	*/*
Host	<calculated at runtime>
authorization	id_token _.App_idtoken
OUN	_.OnlineStore_OUN

3. Convert the cart to "Customer order" (by updating the cart, the cart change to customer order)



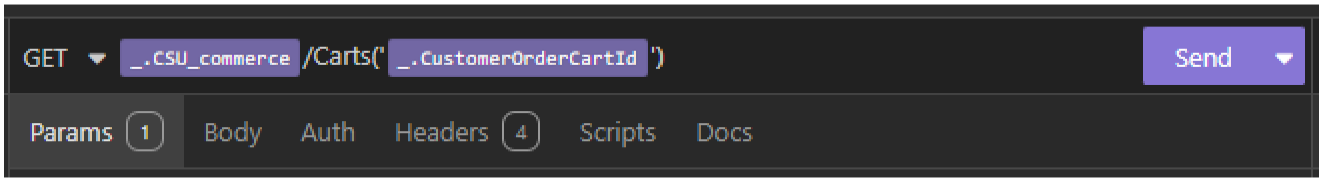
POST _.CSU_commerce/Carts('_.CartId')/Copy

Params 1 | Body ● | Auth | Headers 5 | Scripts ● | Docs

JSON

```
{"targetCartType":2}
```

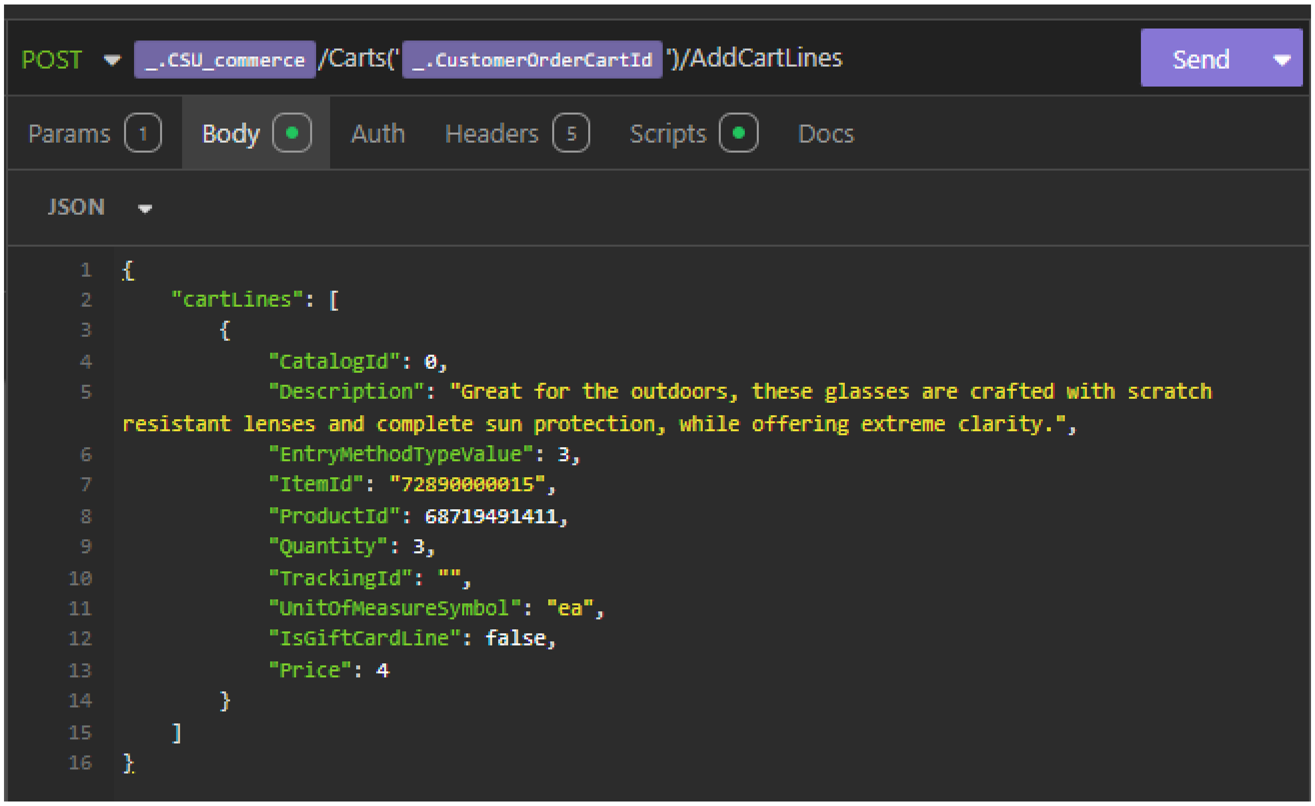
4. Obtain the newly created cart during the update to 'customer order'



GET _.CSU_commerce/Carts('_.CustomerOrderCartId')

Params 1 | Body | Auth | Headers 4 | Scripts | Docs

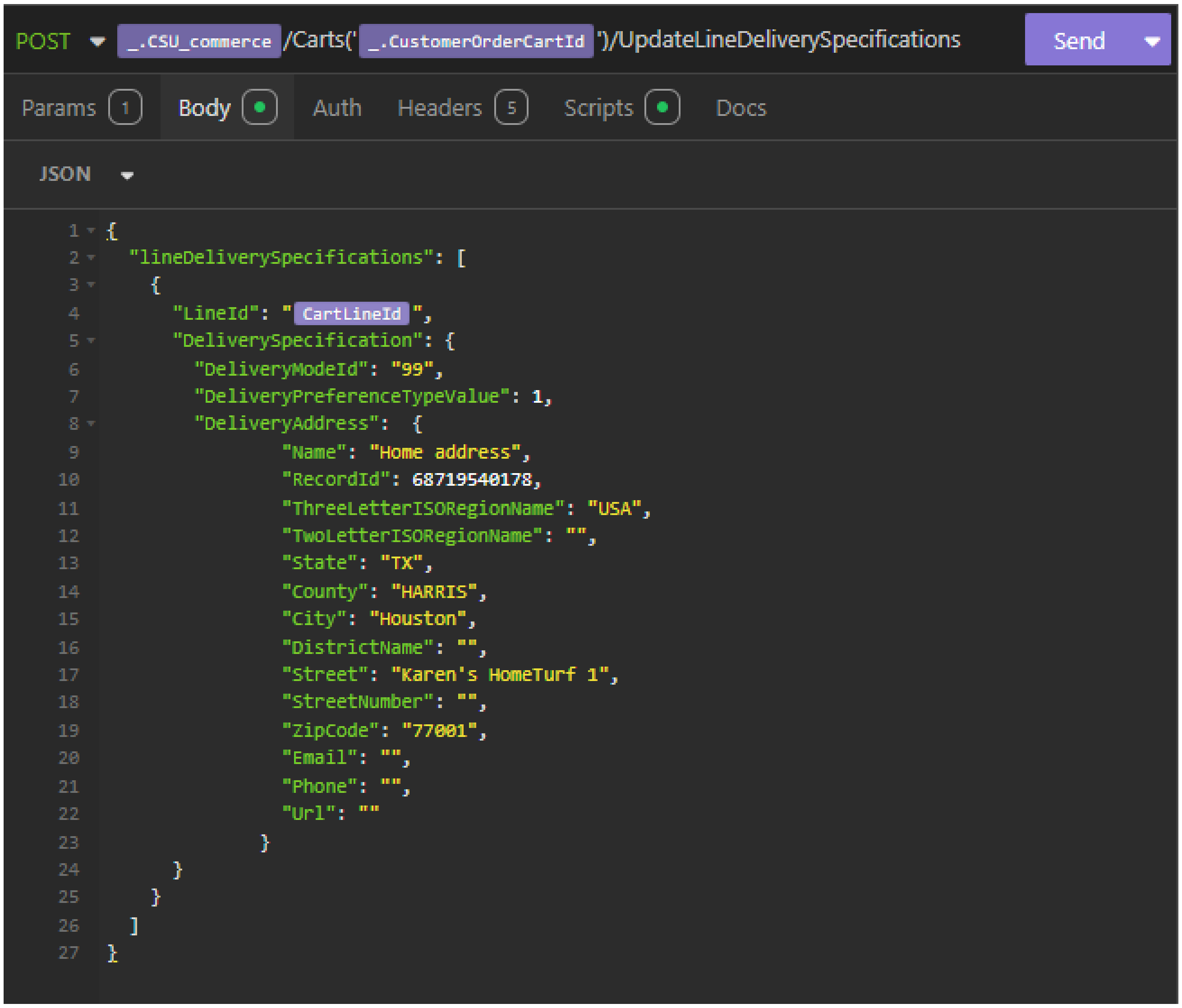
5. Add a cart line



```
_.CSU_commerce/Carts('_.CustomerOrderCartId')/AddCartLines
```

```
{
    "cartLines": [
        {
            "CatalogId": 0,
            "Description": "Great for the outdoors, these glasses are crafted with scratch resistant lenses and complete sun protection, while offering extreme clarity.",
            "EntryMethodTypeValue": 3,
            "ItemId": "72890000015",
            "ProductId": 68719491411,
            "Quantity": 3,
            "TrackingId": "",
            "UnitOfMeasureSymbol": "ea",
            "IsGiftCardLine": false,
            "Price": 4
        }
    ]
}
```

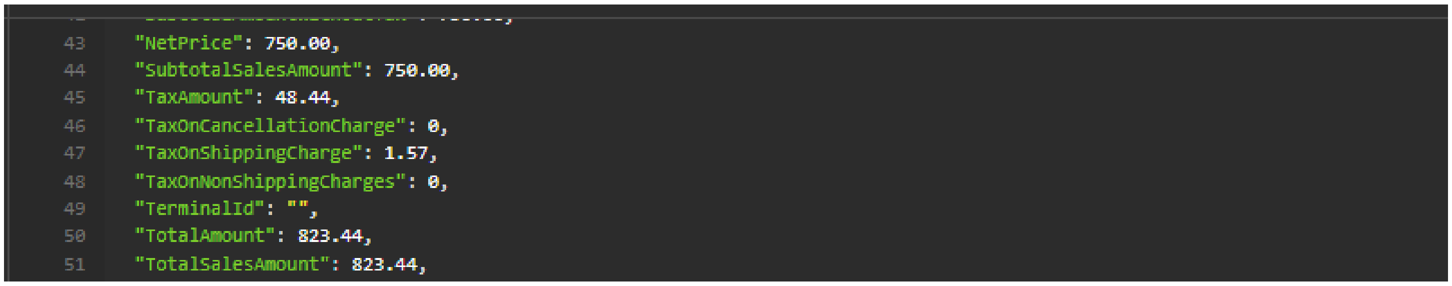
6. Update delivery mode and address in cart



```
POST _.CSU_commerce /Carts(' _.CustomerOrderCartId ')/UpdateLineDeliverySpecifications
```

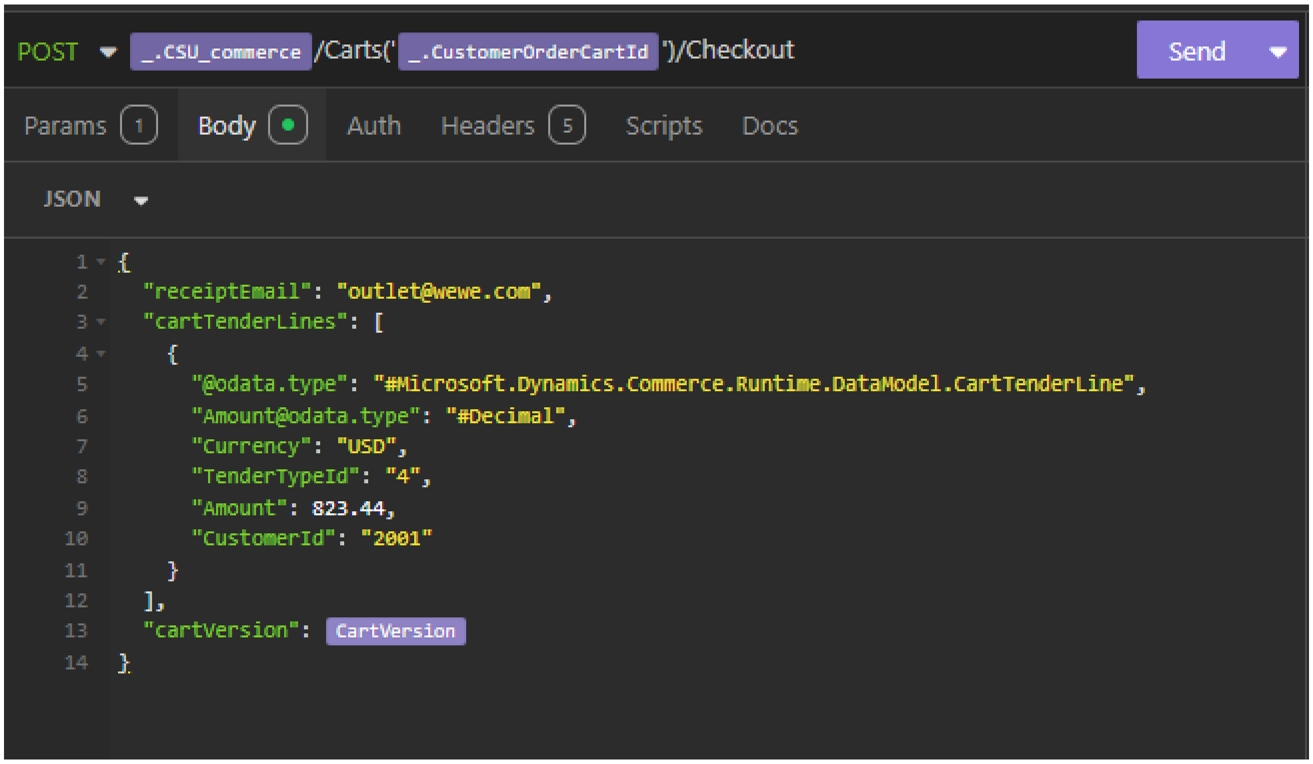
```
{
  "lineDeliverySpecifications": [
    {
      "LineId": " CartLineId ",
      "DeliverySpecification": {
        "DeliveryModeId": "99",
        "DeliveryPreferenceTypeValue": 1,
        "DeliveryAddress":  {
                "Name": "Home address",
                "RecordId": 68719540178,
                "ThreeLetterISORegionName": "USA",
                "TwoLetterISORegionName": "",
                "State": "TX",
                "County": "HARRIS",
                "City": "Houston",
                "DistrictName": "",
                "Street": "Karen's HomeTurf 1",
                "StreetNumber": "",
                "ZipCode": "77001",
                "Email": "",
                "Phone": "",
                "Url": ""
              }
      }
    }
  ]
}
```

- Once the delivery method is added, the total amount will be recalculated. Please note the final payable amount.



```
"NetPrice": 750.00,
"SubtotalSalesAmount": 750.00,
"TaxAmount": 48.44,
"TaxOnCancellationCharge": 0,
"TaxOnShippingCharge": 1.57,
"TaxOnNonShippingCharges": 0,
"TerminalId": "",
"TotalAmount": 823.44,
"TotalSalesAmount": 823.44,
```

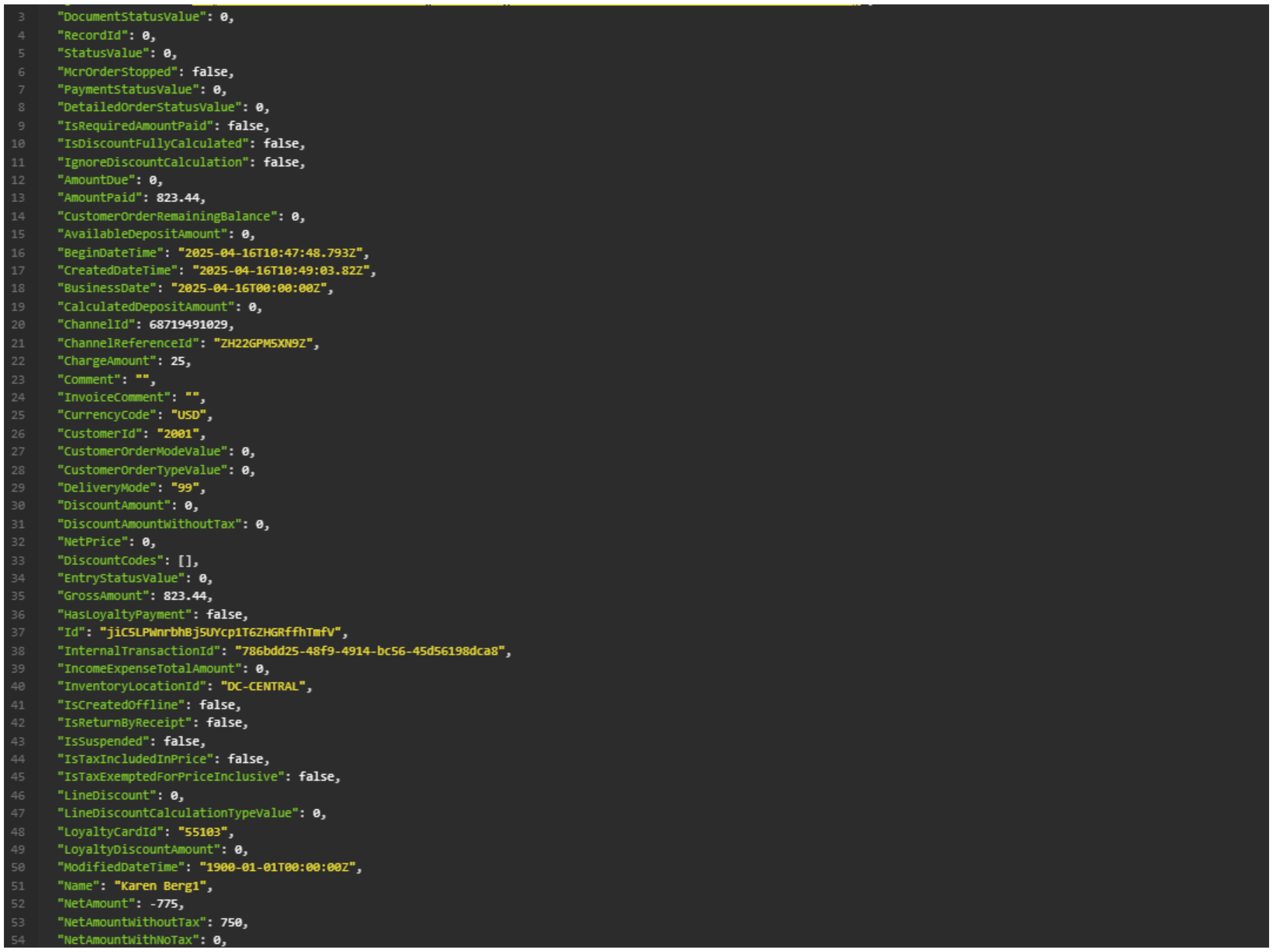
7. Check out the cart, make sure to complete payment with the correct payable amount from the above step.



```
_.CSU_commerce /Carts(' _.CustomerOrderCartId ')/Checkout
```

```
{
  "receiptEmail": "outlet@wewe.com",
  "cartTenderLines": [
    {
      "@odata.type": "#Microsoft.Dynamics.Commerce.Runtime.DataModel.CartTenderLine",
      "Amount@odata.type": "#Decimal",
      "Currency": "USD",
      "TenderTypeId": "4",
      "Amount": 823.44,
      "CustomerId": "2001"
    }
  ],
  "cartVersion": CartVersion
}
```

- Executing the checkout will commit the cart into the Commerce scale unit. Subsequently, once the P-Job is completed, the checked-out cart will be synchronized to the Retail transaction table in the back office.

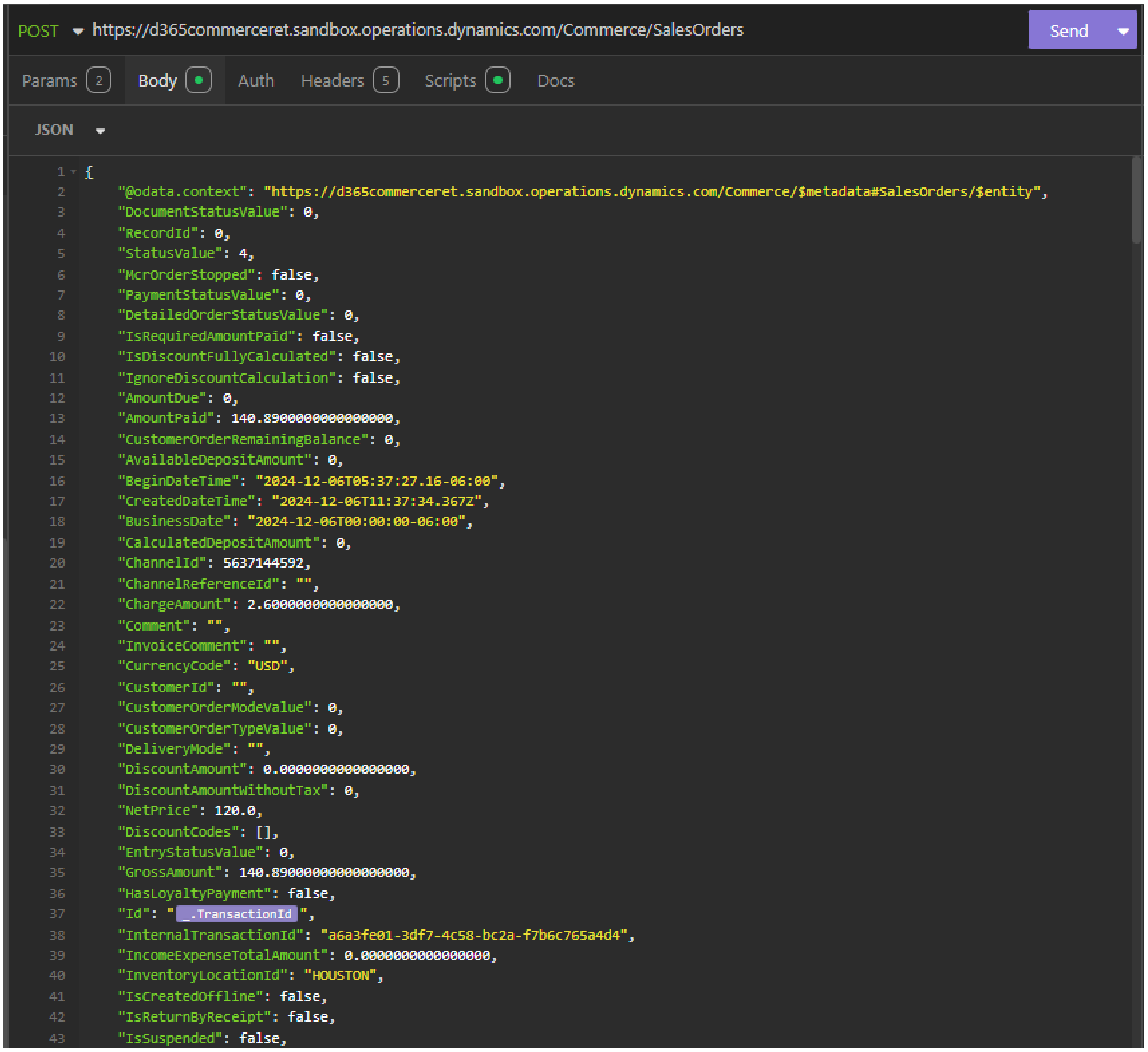


```
    "DocumentStatusValue": 0,
    "RecordId": 0,
    "StatusValue": 0,
    "McrOrderStopped": false,
    "PaymentStatusValue": 0,
    "DetailedOrderStatusValue": 0,
    "IsRequiredAmountPaid": false,
    "IsDiscountFullyCalculated": false,
    "IgnoreDiscountCalculation": false,
    "AmountDue": 0,
    "AmountPaid": 823.44,
    "CustomerOrderRemainingBalance": 0,
    "AvailableDepositAmount": 0,
    "BeginDateTime": "2025-04-16T10:47:48.793Z",
    "CreatedDateTime": "2025-04-16T10:49:03.82Z",
    "BusinessDate": "2025-04-16T00:00:00Z",
    "CalculatedDepositAmount": 0,
    "ChannelId": 68719491029,
    "ChannelReferenceId": "ZH22GPM5XN9Z",
    "ChargeAmount": 25,
    "Comment": "",
    "InvoiceComment": "",
    "CurrencyCode": "USD",
    "CustomerId": "2001",
    "CustomerOrderModeValue": 0,
    "CustomerOrderTypeValue": 0,
    "DeliveryMode": "99",
    "DiscountAmount": 0,
    "DiscountAmountWithoutTax": 0,
    "NetPrice": 0,
    "DiscountCodes": [],
    "EntryStatusValue": 0,
    "GrossAmount": 823.44,
    "HasLoyaltyPayment": false,
    "Id": "jiC5LPWnrbhBj5UYcp1T6ZHGRffhTmfV",
    "InternalTransactionId": "786bdd25-48f9-4914-bc56-45d56198dca8",
    "IncomeExpenseTotalAmount": 0,
    "InventoryLocationId": "DC-CENTRAL",
    "IsCreatedOffline": false,
    "IsReturnByReceipt": false,
    "IsSuspended": false,
    "IsTaxIncludedInPrice": false,
    "IsTaxExemptedForPriceInclusive": false,
    "LineDiscount": 0,
    "LineDiscountCalculationTypeValue": 0,
    "LoyaltyCardId": "55103",
    "LoyaltyDiscountAmount": 0,
    "ModifiedDateTime": "1900-01-01T00:00:00Z",
    "Name": "Karen Berg1",
    "NetAmount": -775,
    "NetAmountWithoutTax": 750,
    "NetAmountWithNoTax": 0,
```

Exercise 2: Upload Sales Order

This exercise simulates asynchronous order integration with headless commerce. The order will appear directly in Backoffice retail transactions tables. It will be synchronized to Sales order by statement posting for brick-and-mortar stores or by synchronizing the online transaction for online stores.

Note: This API will not validate the transaction, so run the 'validate retail transaction job' to validate the uploaded transaction.

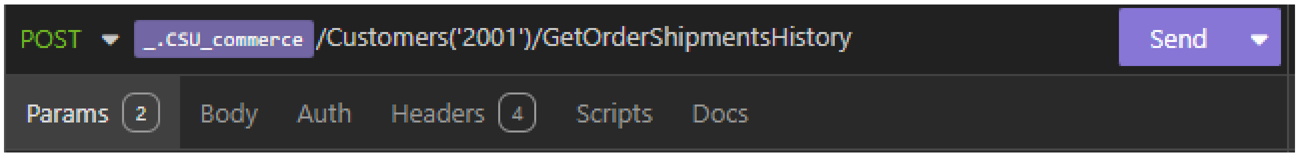

POST
https://d365commerceret.sandbox.operations.dynamics.com/Commerce/SalesOrders
Send
Params 2
Body
Auth
Headers 5
Scripts
Docs
JSON

```
{
    "@odata.context": "https://d365commerceret.sandbox.operations.dynamics.com/Commerce/$metadata#SalesOrders/$entity",
    "DocumentStatusValue": 0,
    "RecordId": 0,
    "StatusValue": 4,
    "McrOrderStopped": false,
    "PaymentStatusValue": 0,
    "DetailedOrderStatusValue": 0,
    "IsRequiredAmountPaid": false,
    "IsDiscountFullyCalculated": false,
    "IgnoreDiscountCalculation": false,
    "AmountDue": 0,
    "AmountPaid": 140.8900000000000000,
    "CustomerOrderRemainingBalance": 0,
    "AvailableDepositAmount": 0,
    "BeginDateTime": "2024-12-06T05:37:27.16-06:00",
    "CreatedDateTime": "2024-12-06T11:37:34.367Z",
    "BusinessDate": "2024-12-06T00:00:00-06:00",
    "CalculatedDepositAmount": 0,
    "ChannelId": 5637144592,
    "ChannelReferenceId": "",
    "ChargeAmount": 2.6000000000000000,
    "Comment": "",
    "InvoiceComment": "",
    "CurrencyCode": "USD",
    "CustomerId": "",
    "CustomerOrderModeValue": 0,
    "CustomerOrderTypeValue": 0,
    "DeliveryMode": "",
    "DiscountAmount": 0.0000000000000000,
    "DiscountAmountWithoutTax": 0,
    "NetPrice": 120.0,
    "DiscountCodes": [],
    "EntryStatusValue": 0,
    "GrossAmount": 140.89000000000000000,
    "HasLoyaltyPayment": false,
    "Id": " _.TransactionId ",
    "InternalTransactionId": "a6a3fe01-3df7-4c58-bc2a-f7b6c765a4d4",
    "IncomeExpenseTotalAmount": 0.0000000000000000,
    "InventoryLocationId": "HOUSTON",
    "IsCreatedOffline": false,
    "IsReturnByReceipt": false,
    "IsSuspended": false,
```

Lab 05: Application - role (Order history)

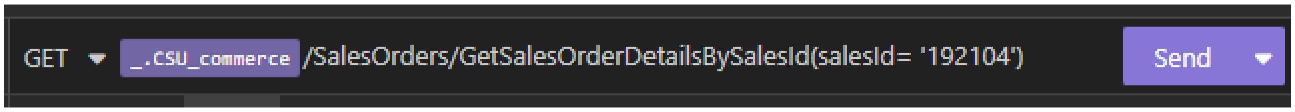
Exercise 1: Order history

- This exercise demonstrates how to retrieve the sales order history for a customer by passing the account number in the URL.


POST _.CSU_commerce /Customers('2001')/GetOrderShipmentsHistory

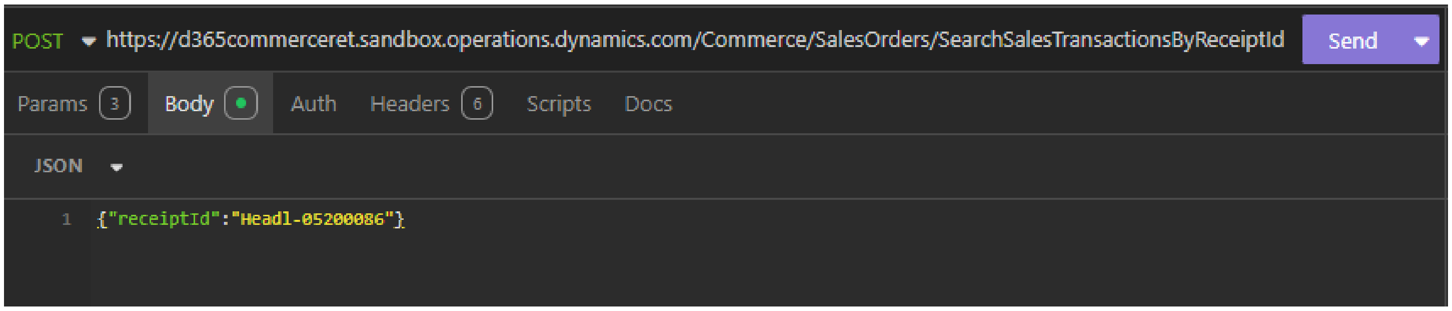
Exercise 2: Sales Order details

- This exercise illustrates the method for retrieving sales order details using the sales ID.


GET _.CSU_commerce /SalesOrders/GetSalesOrderDetailsBySalesId(salesId= '192104')

Exercise 3: Search by Receipt Id

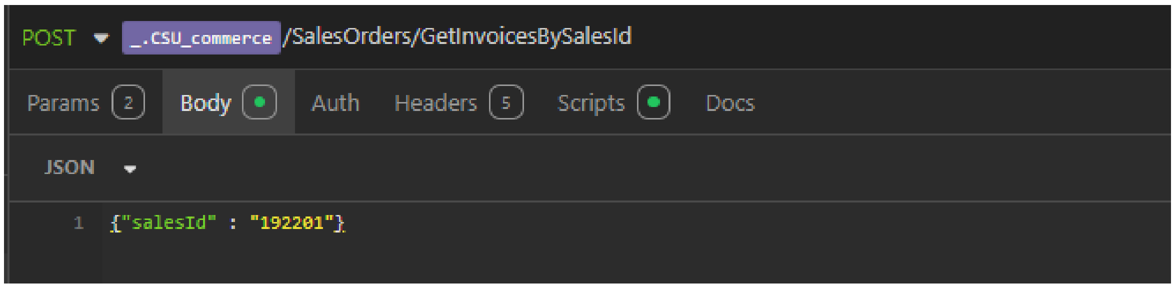
- This exercise shows how to retrieve sales transaction details with the receipt ID.


POST https://d365commerceret.sandbox.operations.dynamics.com/Commerce/SalesOrders/SearchSalesTransactionsByReceiptId

```
{"receiptId":"Headl-05200086"}
```

Exercise 4: Get Invoices by Sales Id

- This exercise shows how to get invoice details for a sales order.


POST _.CSU_commerce /SalesOrders/GetInvoicesBySalesId

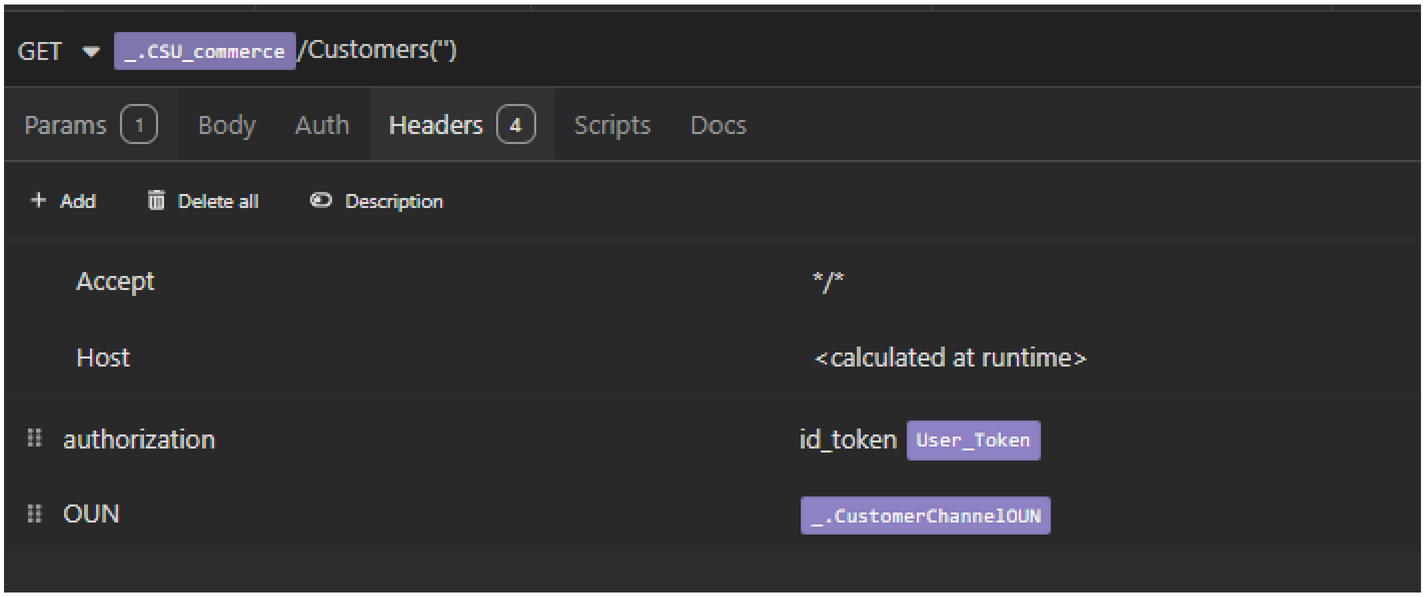
```
{"salesId" : "192201"}
```

Lab 06: Customer - role

Pro tip: To enable the customer role, ensure the API header's ID token is updates with user id token, which is extracted from a logged-in user after completing customer authentication configuration as described here: Set up a B2C tenant in Commerce - Commerce | Dynamics 365 | Microsoft Learn

Exercise 1: Customer login

This exercise simulates the customer login scenario where the API is accessed using a user ID token. The headless commerce engine then resolves the user Id-token to the appropriate customer account and retrieves the customer details.



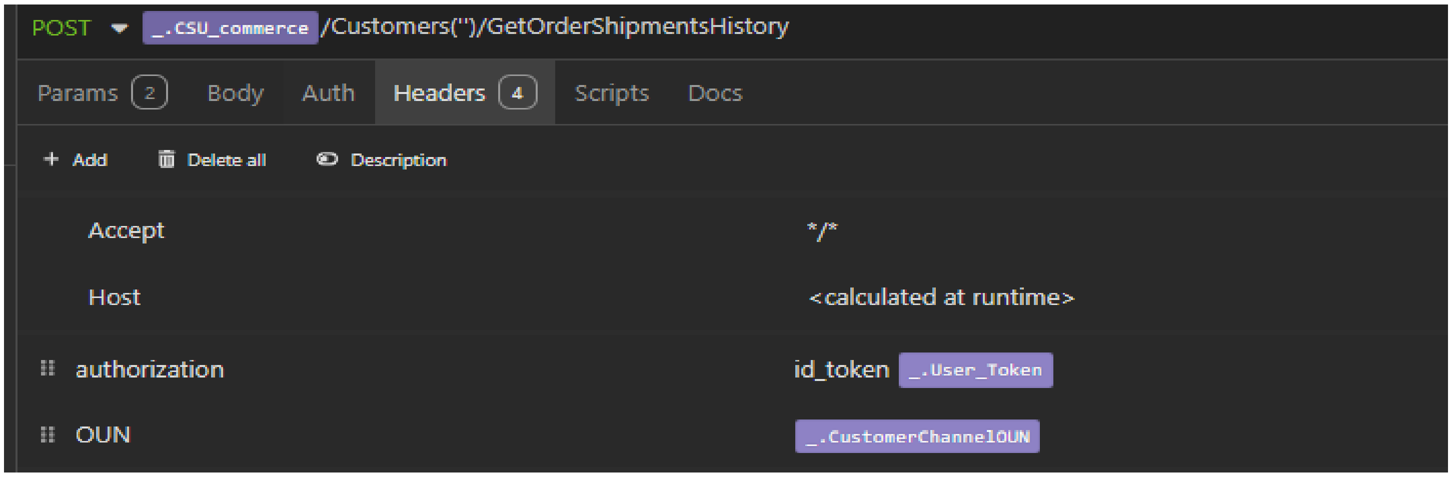
GET _.CSU_commerce/Customers('')

Accept	*/*
Host	<calculated at runtime>
authorization	id_token User_Token
OUN	_.CustomerChannelOUN

Exercise 2: Customer Order history

- This exercise simulates retrieving the order history for a customer.

The account number should be retrieved based on the User ID token, as no customer account is passed on in the URL.



POST _.CSU_commerce/Customers('')/GetOrderShipmentsHistory

Accept	*/*
Host	<calculated at runtime>
authorization	id_token _.User_Token
OUN	_.CustomerChannelOUN

Lab 07: Knowledge Test

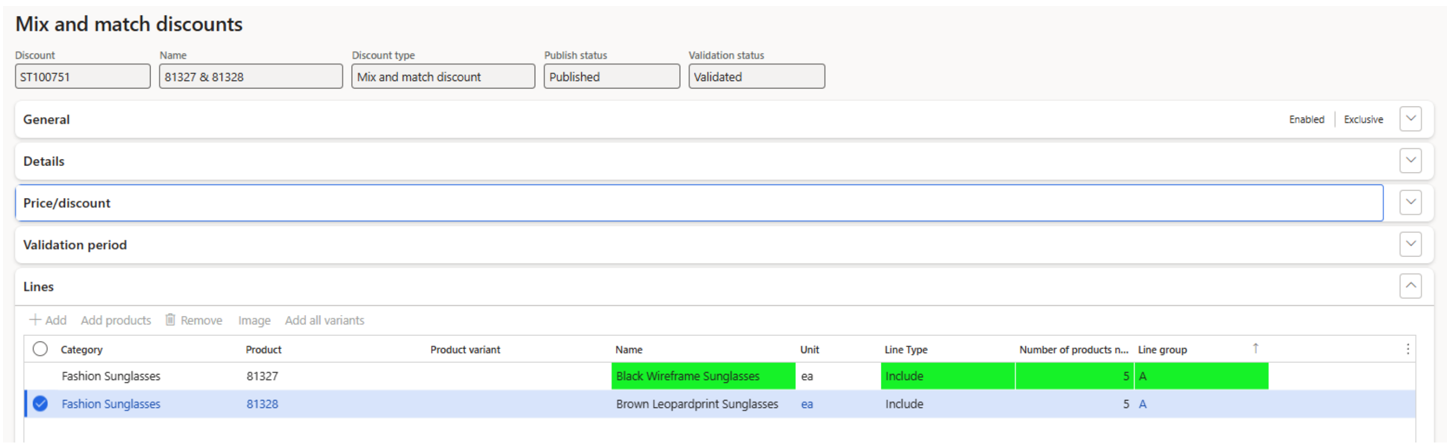
Exercise 1: Find the deal price?

Please retrieve the mix and match discount from Headless commerce based on the input parameters provided.

Set up the Mix and Match deal price, then verify it in POS as a prerequisite.

Mix and match setup:

- Item ID: 81327, 81328
- Quantity: 5
- Find the deal price?



Mix and match discounts

Discount	Name	Discount type	Publish status	Validation status
ST100751	81327 & 81328	Mix and match discount	Published	Validated

General — Enabled | Exclusive

Details

Price/discount

Validation period

Lines

Category	Product	Product variant	Name	Unit	Line Type	Number of products n...	Line group
Fashion Sunglasses	81327		Black Wireframe Sunglasses	ea	Include	5	A
Fashion Sunglasses	81328		Brown Leopardprint Sunglasses	ea	Include	5	A

Pro tip: The "*CalculateSalesDocument*" API determines prices and discounts for grouped product quantities without creating the cart. Commerce pricing APIs - Commerce | Dynamics 365 | Microsoft Learn