Adobe Experience Platform Bootcamp Deep Dive Edition

DOCUMENTATION AND LABS

CONTENTS

1. LAB OVERVIEW

Defining schemas and understanding how data is described by those schemas is one of the first things a customer will do on Adobe Experience Platform. This is the first step to ingesting data to the platform.

This lab will introduce you to the Adobe Experience Platform schema design user experience. You will learn how to create your own schemas, as well as browse existing components.

2. LEARNING OBJECTIVES

What should you walk away with after taking this Lab?

Create various AEP/XDM components using API Calls. Examples are:

1. Class
2. Field Group
3. Schema
4. Descriptors
    a. Identities (including Primary)
    b. Relationship
    c. Reference

Adding some Schema configurations through Schema UI interface. Browse AEP/XDM Components through API Calls Extending the Event Types via API Calls Validating the created Schemas through UI.

3. PRE REQUISITES

Learners should have gone through the following concepts/courses/modules

- Basics of XDM and Schema Composition
- Profile concepts and the important XDM components for Profile SID Methodology – Should have applied the SID methodology to the **Customer Source Data Model** and should have the expected **Customer Data - AEP XDM ERD**
- Should be aware of the Source to XDM Mapping Document (**Orders Schema - Customer to XDM Mapping.xlsx**). This will be the source of all the Schemas created in this Lab.
- As part of your POST setup class, you would have POSTMAN setup and you should know how to set up your **Environment Configuration,** You should also know how to generate your access token and import a collection.
- Please make sure you are aware of your tenant name space. This lab provides an example using sample tenant name space **"dxp"**. You will have to use/replace your own tenant name space while performing the LAB steps.

4. LAB GUIDE

This guide will walk you through step by step process to create Orders Schema and its corresponding relationships to other Lookup/Dimensional Schemas via API Calls.

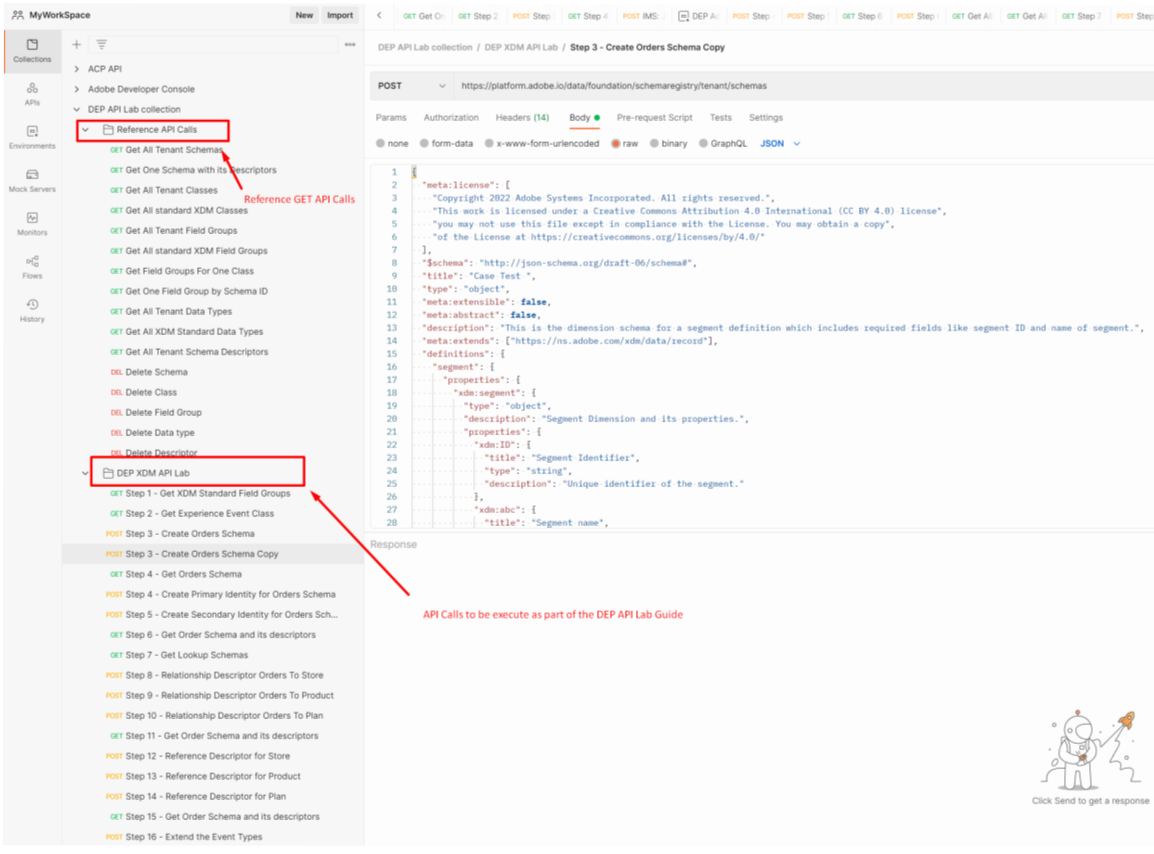
4.1. Import the following collection into your POSTMAN. This collection has the following folders: DEP API Lab collection

1. DEP XDM API Lab

a. Contains all the API calls to create Order related XDM components like Schema, Field Groups, Primary Identities, Secondary Identities, Relationships and Reference Identities.

2. Reference API Calls

a. Contains some reference **"GET"** API calls to help you navigate the Schema registry
b. You can use these API calls to browse the XDM components you created and also to get identifiers of various XDM components which you would need as you go through the API Lab. It would be good to have at least have a glance at these
c. Please note that you will have to **create/POST** the XDM components first to expect results in the GET API calls.
d. Below is a sample screen shot of the folders in your **POSTMAN** collection.

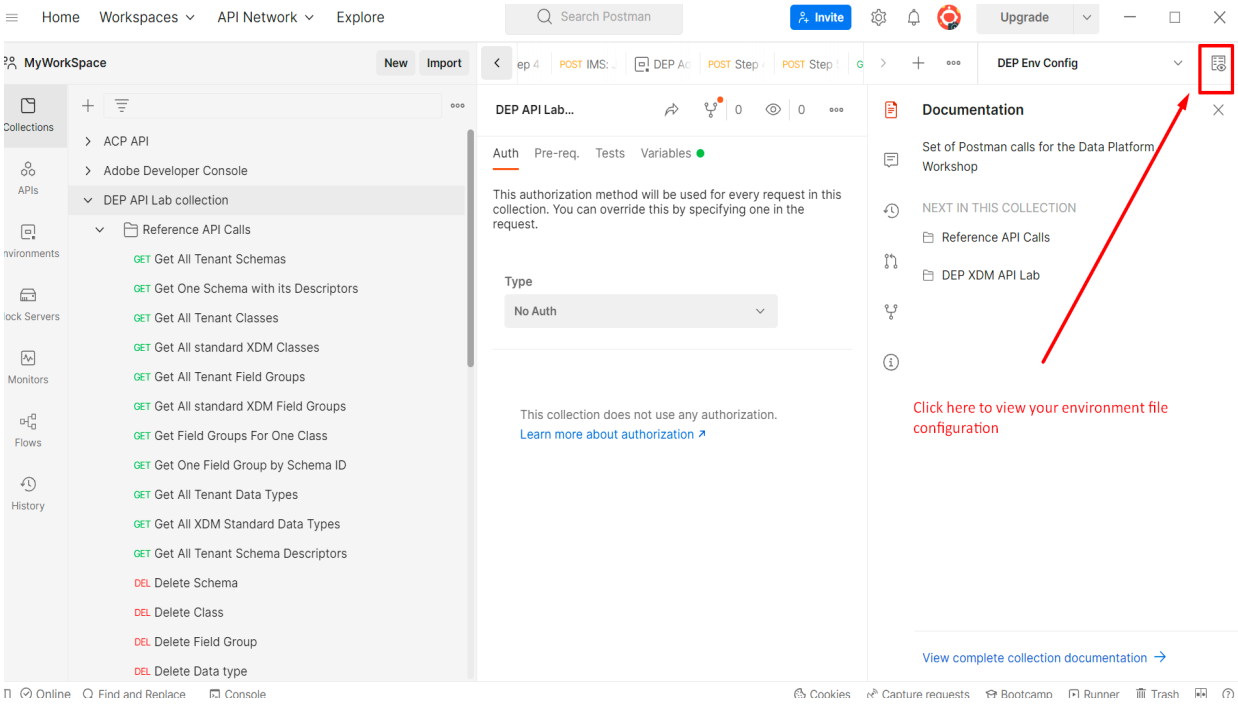


Glossary

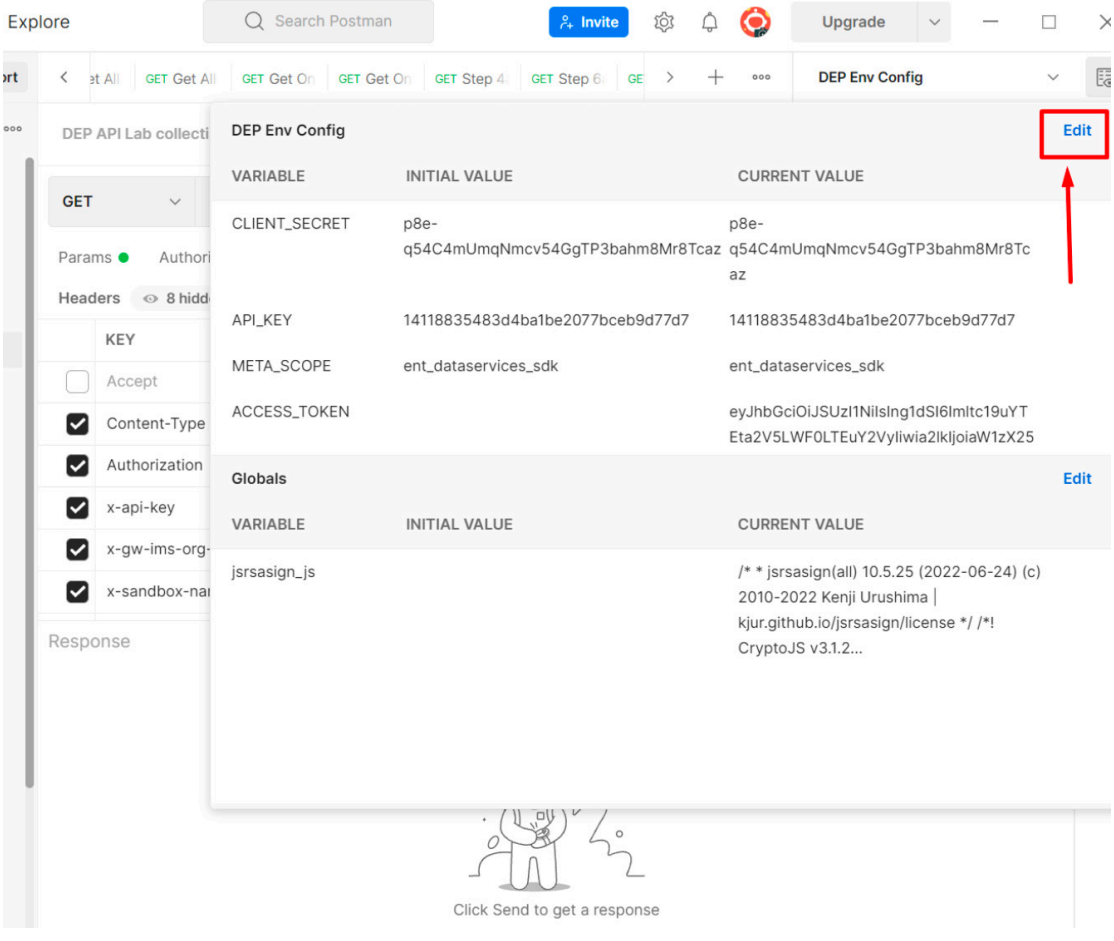
Term	Description/Definition
_dxp	Sample tenant namespace
https://platform.adobe.io/data/foundation/schemaregistry/tenant	Schema Registry end point to access tenant XDM components
https://platform.adobe.io/data/foundation/schemaregistry/global	Schema Registry end point to access standard XDM components
https://platform.adobe.io/data/foundation/schemaregistry/global/schemas	Schema Registry end point for standard XDM Schemas
https://platform.adobe.io/data/foundation/schemaregistry/tenant/schemas	Schema Registry end point for tenant XDM Schemas
https://platform.adobe.io/data/foundation/schemaregistry/global/mixins	Schema Registry end point for standard XDM Field Groups
https://platform.adobe.io/data/foundation/schemaregistry/tenant/schemas	Schema Registry end point for tenant XDM Field Groups
https://platform.adobe.io/data/foundation/schemaregistry/global/descriptors	Schema Registry end point for standard XDM Schema Descriptors
https://platform.adobe.io/data/foundation/schemaregistry/tenant/descriptors	Schema Registry end point for tenant XDM Schema Descriptors
GET	API call to fetch an XDM component
POST	API call to create an XDM component
PATCH	API call to update or add to an existing XDM component

4.2. Add the "SANDBOX_NAME" variable into your environment file.

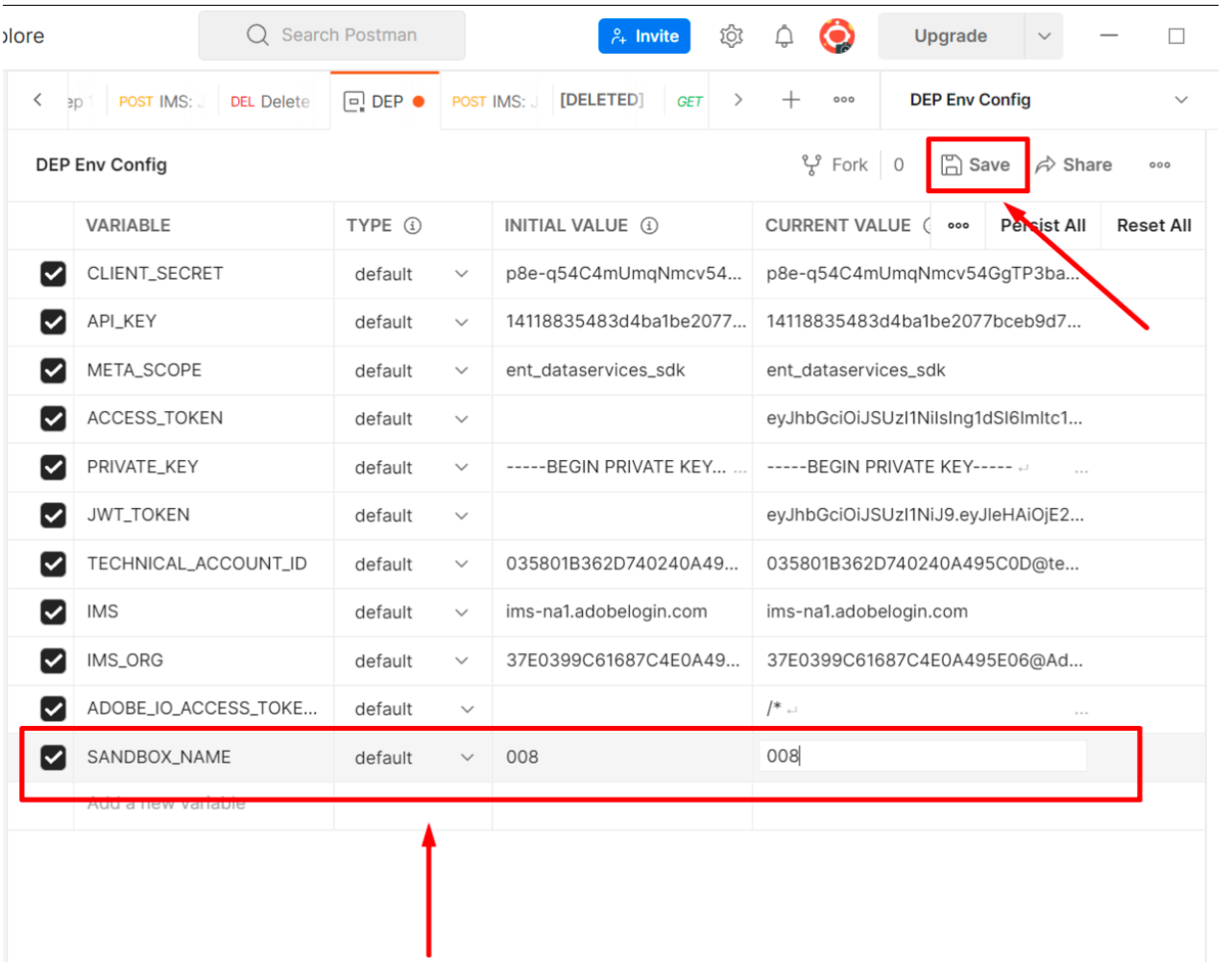
1. Your Sandbox name should be provided to you when you start the API Lab.
2. Click at the **"Environment Quick Look"** button next to your Environment FIle



3. Click **"Edit"** button to add the variable for Sandbox as shown below:



5. Add the variable **"SANDBOX_NAME"** in the environment file as shown below:
6. Make sure you hit the "Save" button after you add the variable and the values in both the **"INITIAL_VALUE"** and **"CURRENT_VALUE"** Fields. example: *attendee-001*



4.3. Get the Schema ID for the XDM standard "Order Details" Field Group

"Field Group" was referred as a **"mixin"** previously so these terms might be used inter-changeably

1. **Click** on **"Step 1 - Get XDM Standard Field Groups"** API call in the "DEP XDM API Lab" Folder. **Do not execute it yet.**
2. **Please note the following points:**
    a. Note the use of **"global"** in **https://platform.adobe.io/data/foundation/schemaregistry/global/mixins**. **"global"** is used to get the XDM provided out of the box standard XDM components (Field group/mixin in this case).
        i. Remember we have two types of owners in AEP (Adobe/Standard and Tenant/custom). The path to these objects is either through:

            1. Adobe/Standard objects will have global in the path e.g. **https://platform.adobe.io/data/foundation/schemaregistry/global/mixins**

2. Tenant/Custom objects will have [tenant] in the path e.g. **https://platform.adobe.io/data/foundation/schemaregistry/tenant/mixins**

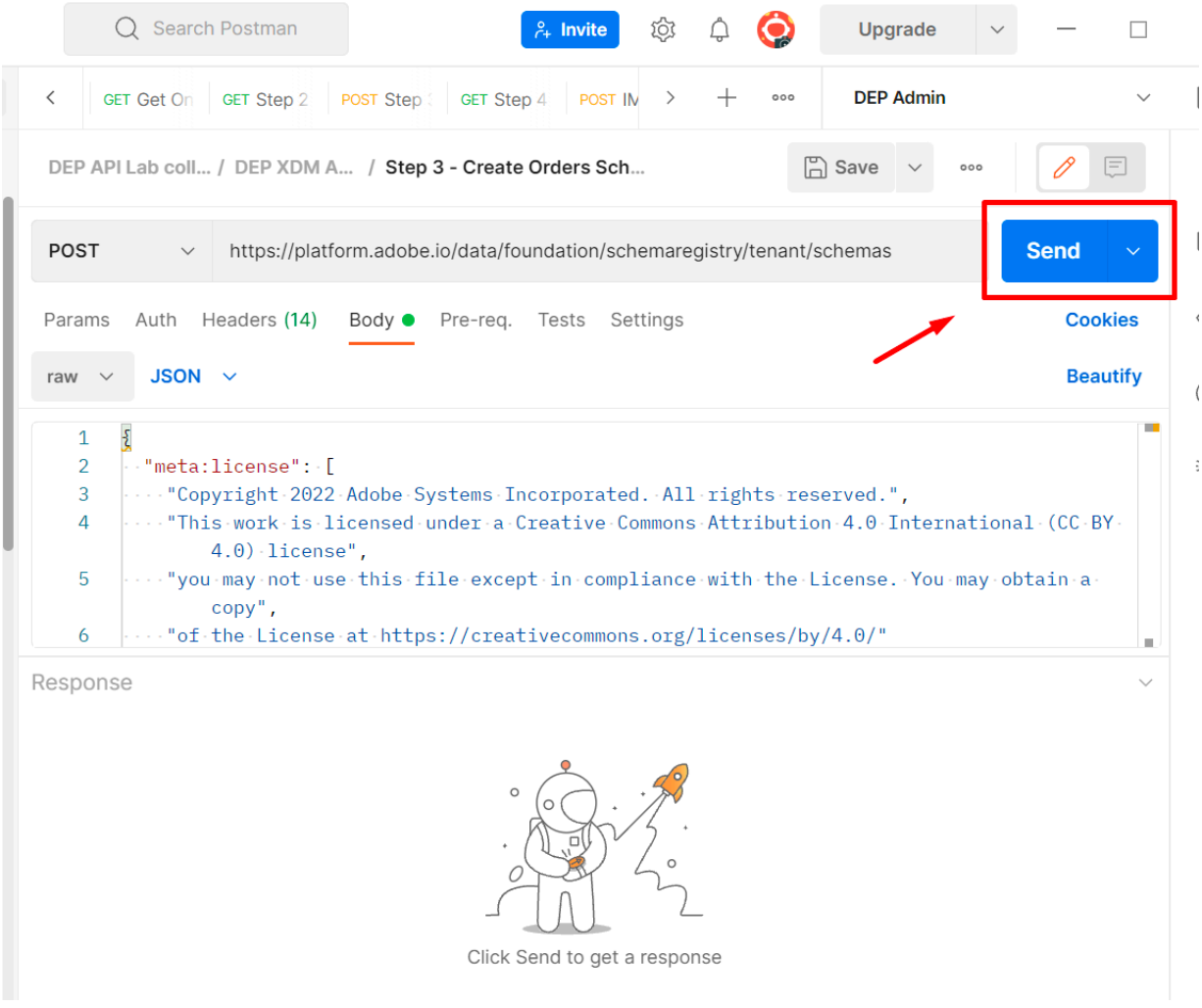
b. This API call gives you all the standard XDM field Groups

3. **Execute "Step 1 - Get XDM Standard Field Groups"** by clicking the **"Send"** button

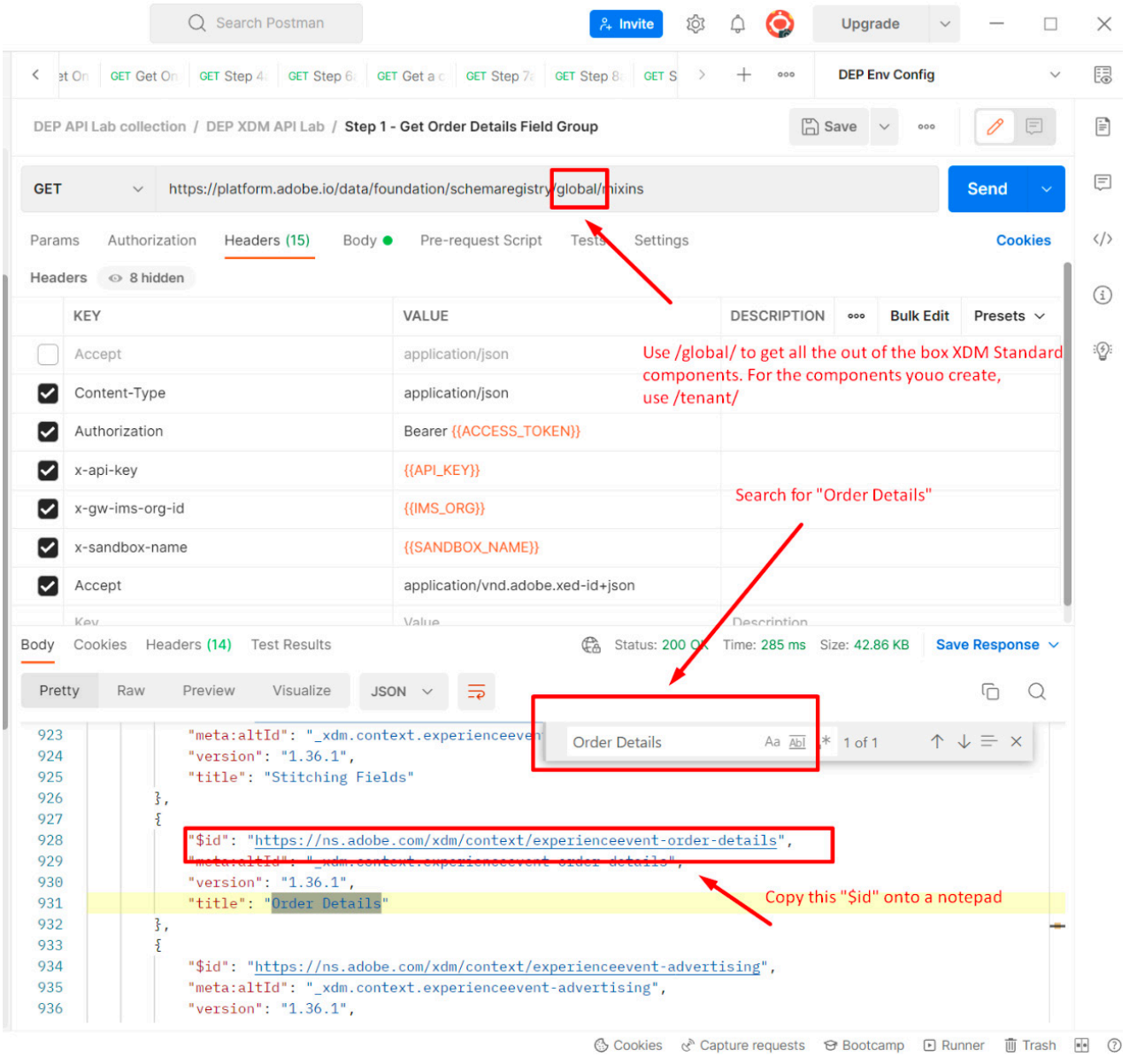
a. After you execute the GET API call, search for **"Order Details"** in the Response

b. Copy the **"$id"** of the **"Order Details"** Field Group and have it handy with you

4. A Sample **"GET"** Schema call looks like below.



5. A Sample Response of a **"GET"** Schema call looks like this.

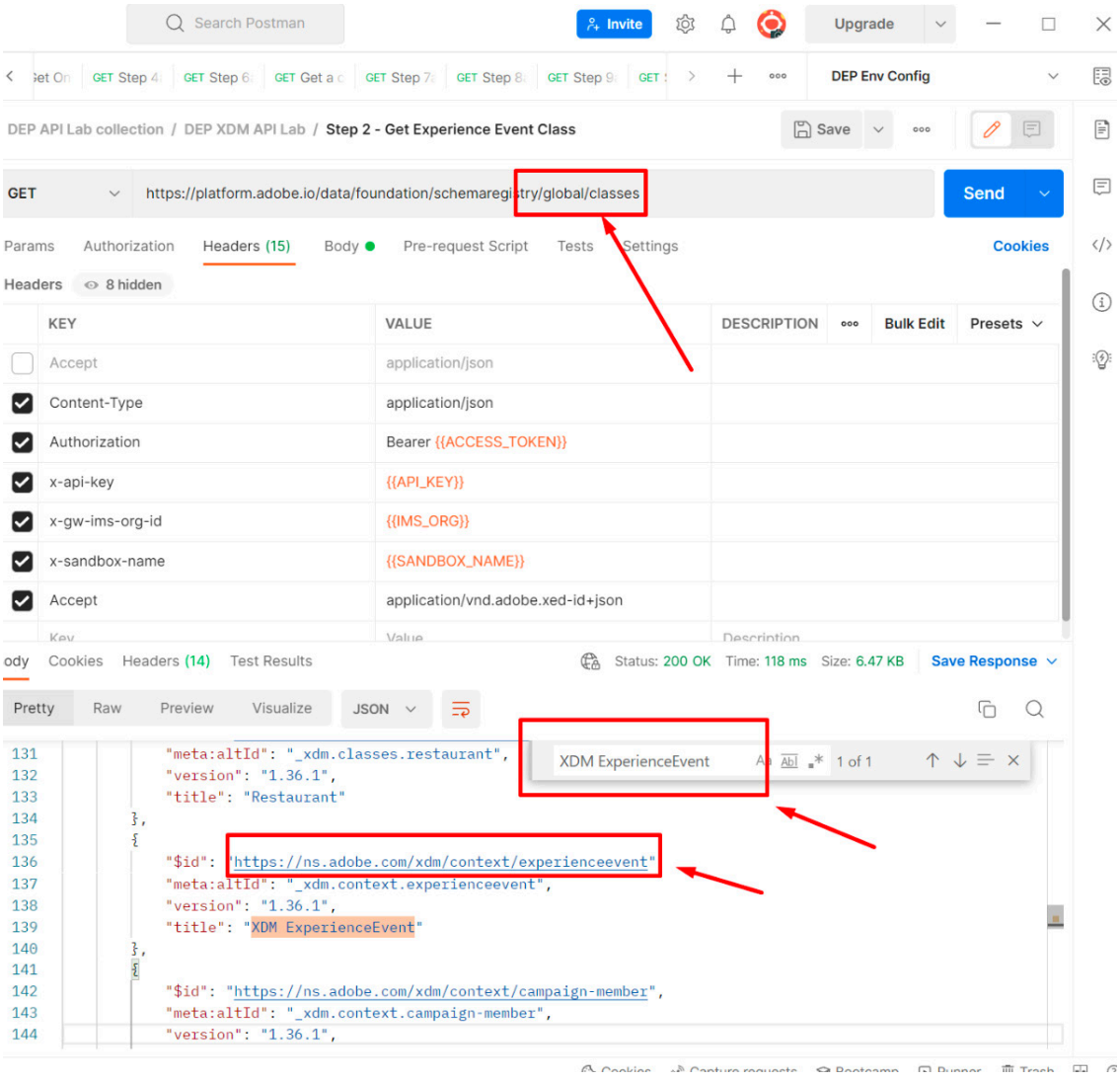


4.4. Get the Schema ID for the XDM standard "Experience Event" Class.

1. **Click on "Step 2 - Get Experience Event Class"** API call in the **"DEP XDM API Lab"** Folder. **Do not execute it yet.**

2. **Please note the following points:**

   a. In this step, we are looking for **"classes"** and not **Field Groups (Mixins)**
   b. This API call gives you all the standard XDM Classes
   c. Execute **"Step 2 - Get Experience Event Class"** by clicking the "Send" button
   d. After you execute the GET API call, search for **"XDM ExperienceEvent"** in the Response
   e. Copy the **"$id"** of the **"XDM ExperienceEvent"** Field Group and have it handy with you.

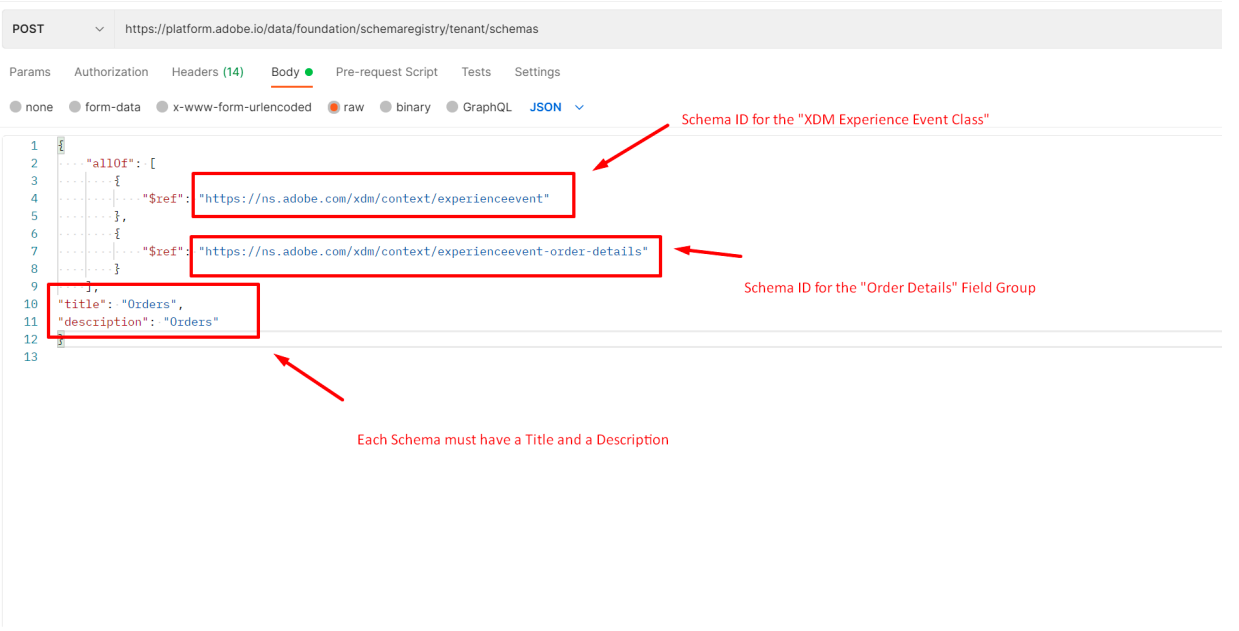


4.5. Create the "Orders" Schema.

1. Click on **"Step 3 - Create Orders Schema"** API call in the "DEP XDM API Lab" Folder. **Do not execute it yet**

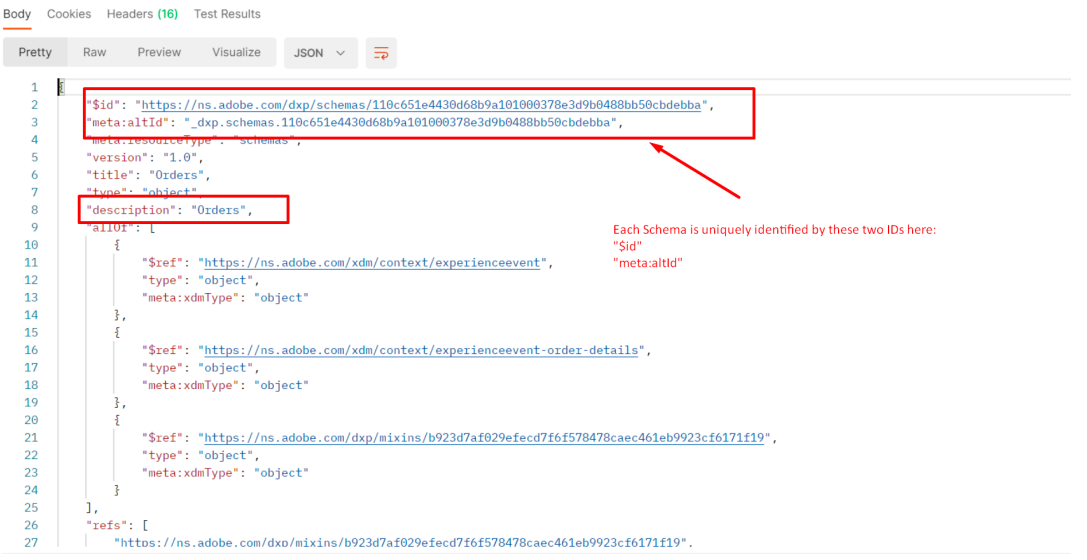
2. Please note the following points:
   a. A Schema must include a **"Class"** and a possible **"Field Group".** This schema would be created by combining
      i. **"XDM Experience Event"** Class
      ii. **"Order Details"** Field Group
   b. Each Schema must have a **"title"** and a **"description".**

3. A Sample payload for creating a Schema looks like this.

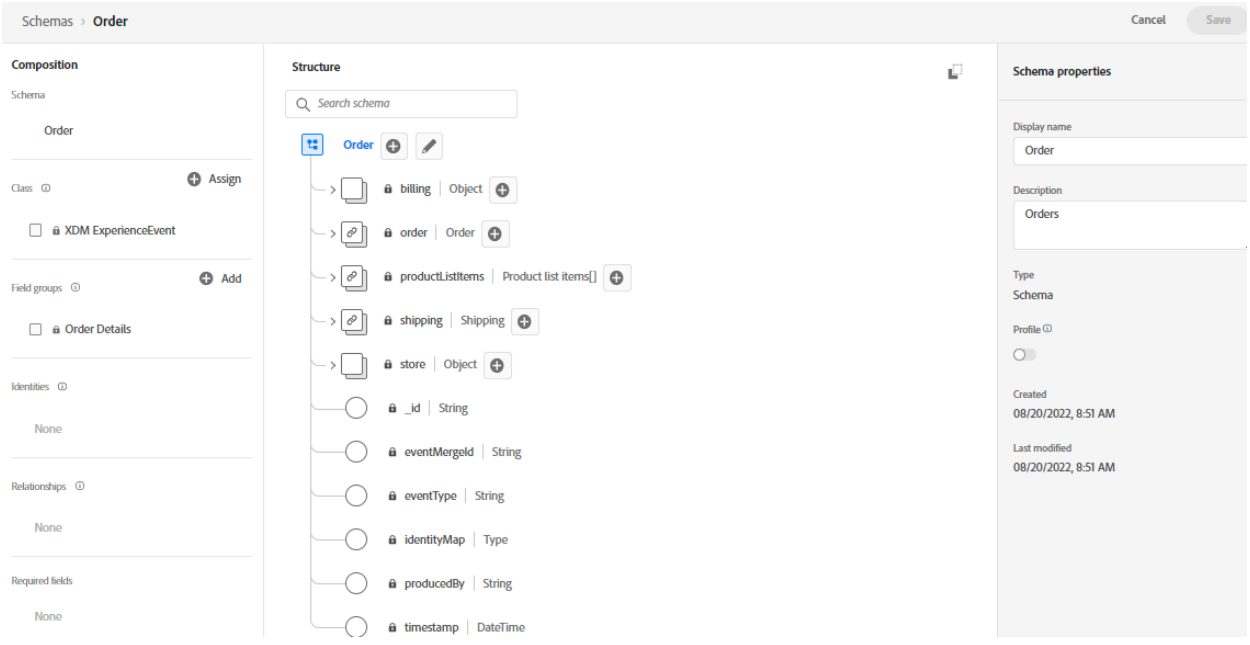


a. Execute **"Step 3 - Create Orders Schema"** by clicking the **"Send"** button
b. Copy the **"$id"** and **"meta:altId"** to use for future references to "Orders" Schema. Keep this copied IDs handy in a notepad as they will be required for future reference in further steps.

4. A Sample Response of a **"POST"** Schema call looks like this.
   a. Note how the Schema URL are structured for Adobe standard XDM components as compared to the Customer created ones. See the use of **"dxp"** in the API response below which is the tenant namespace assigned to an IMS Org.



4.6. Validate and verify your Schema through the UI



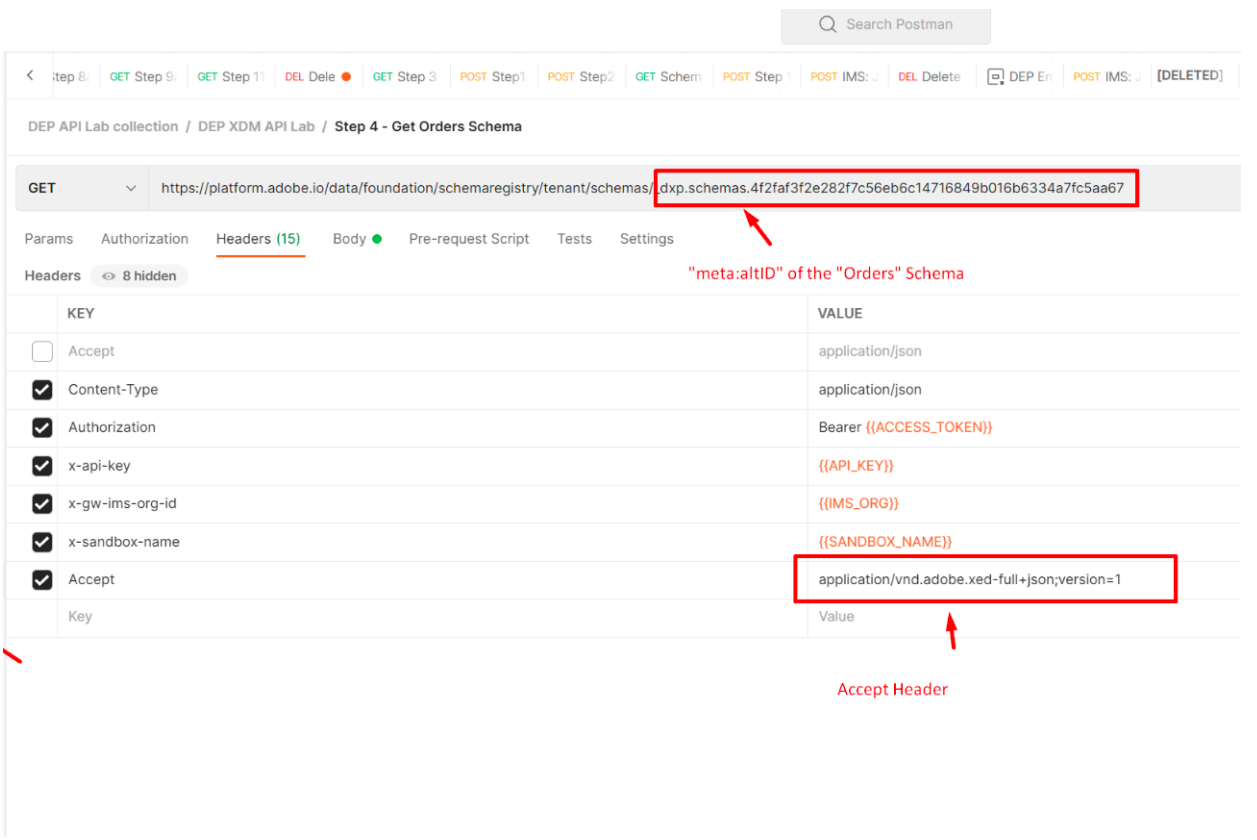
4.7. Add Custom Properties to Orders Schema.

1. Go to the Schema UI editor and add the following properties to the "Orders" Schema
2. These additional properties are also in the Source to XDM Mapping document here : **Orders Schema - Customer to XDM Mapping.xlsx**

```
_dxp {
  acqSource
  customerID
  personalEmail
},
order {
  _dxp {
    plan {
      planID
      name
    }
  }
},
productListItems[
  _dxp {
    make
    model
  }
]
```

4.8. Browse the "Orders" Schema you created above

1. **Click on "Step 4 - Get Orders Schema"** API call in the **"DEP XDM API Lab"** Folder. **Do not execute it yet.**
2. **Please note the following points:**
   b. Note the use of **"tenant"** in **https://platform.adobe.io/data/foundation/schemaregistry/tenant/mixins.**
   c. Observe the use of "meta:altId" to fetch the Schema from the registry. Replace the meta:altID in your API from the step 4.5 above.
   d. Also observe the "Accept" header which tells the schema registry what to return back in the API Response. You can see other "Accept" header options here **Schema Registry APIs**
   e. Execute **"Step 4 - Get Orders Schema"** by clicking the "Send" button
   f. With the pre-loaded Accept Header in the postman collection, you will see the fully exploded hierarchical view of the Orders Schema where every single property belonging to a class or a field group will be exposed.
   g. All the object properties will be exploded all the way to their leaf nodes.
3. A Sample **"GET"** Schema Descriptor call looks like below.



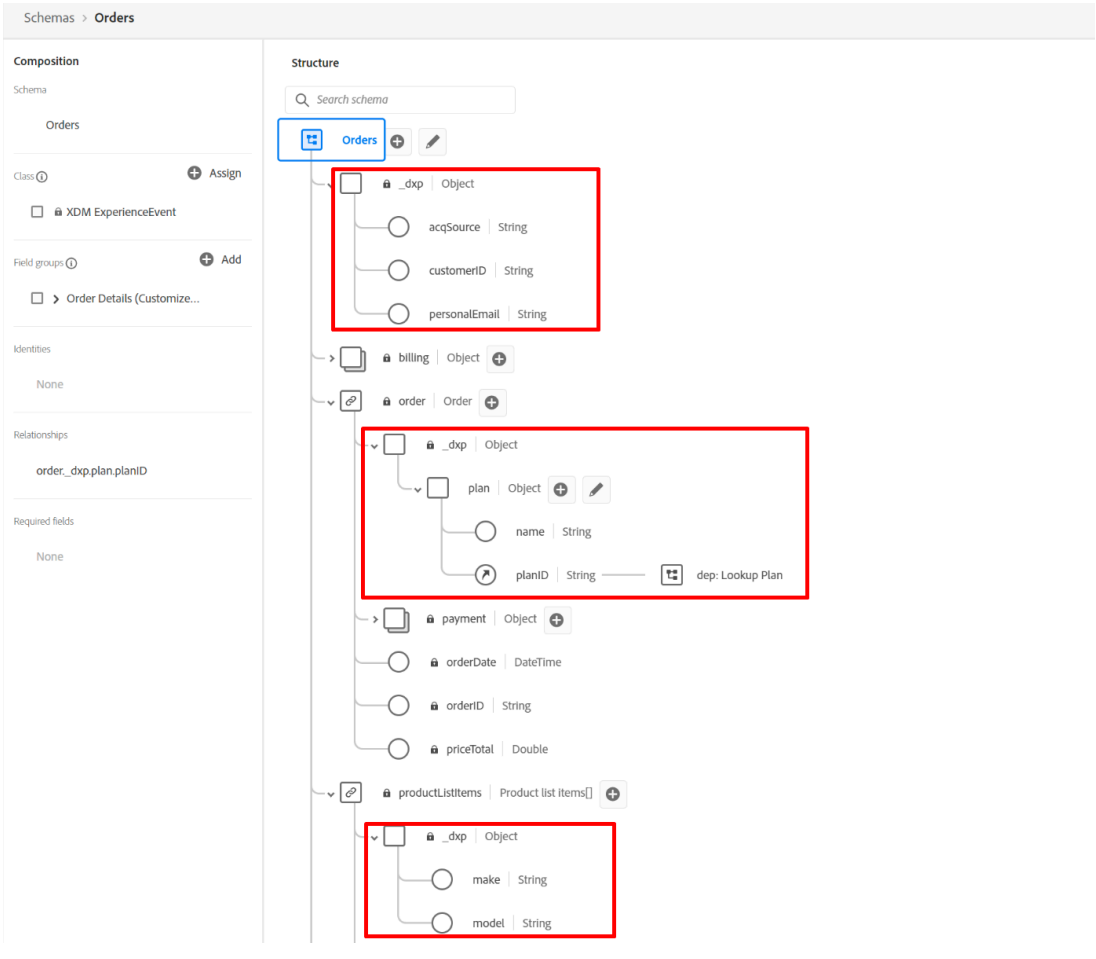
4. A Sample API **"RESPONSE"** looks like below:

Body Cookies Headers (15) Test Results

Pretty Raw Preview Visualize JSON

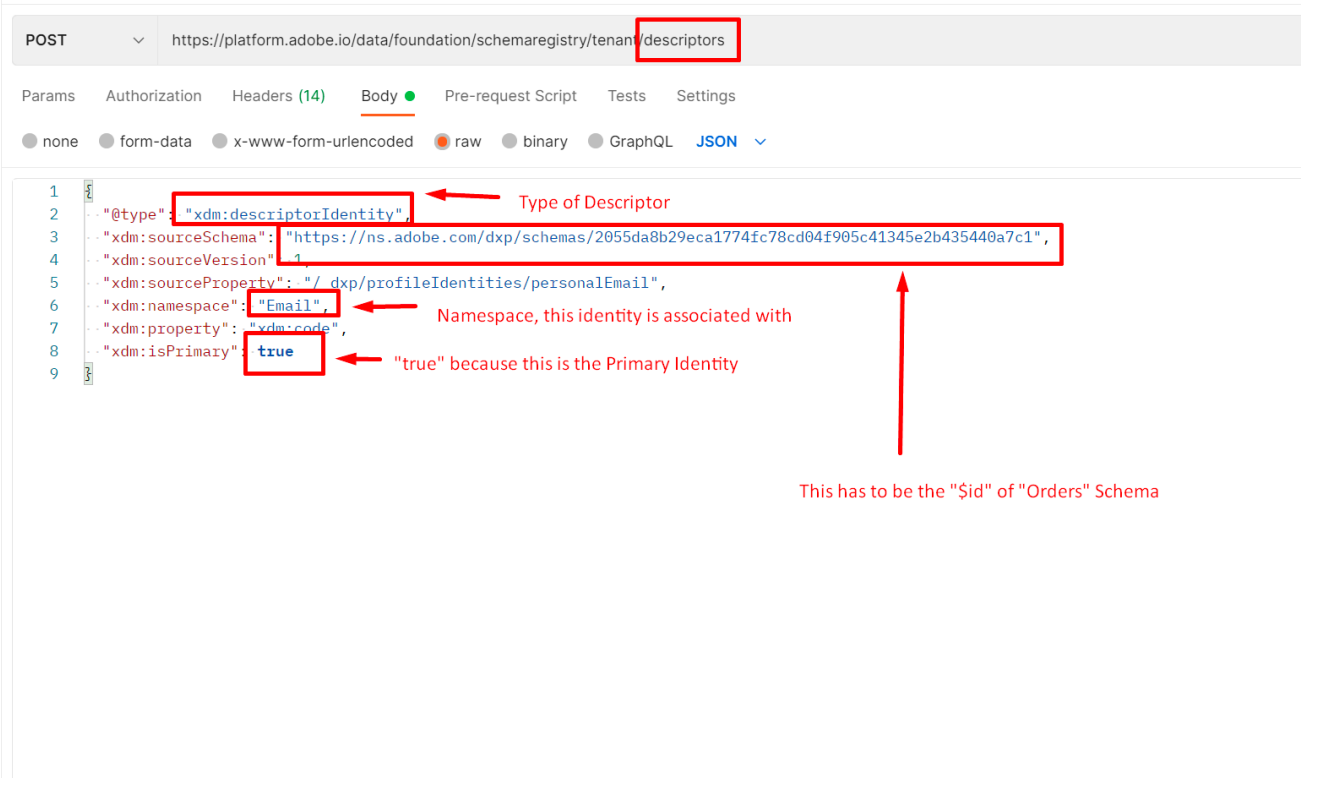

```
{
    "$id": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
    "meta:altId": "_dxp.schemas.4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
    "meta:resourceType": "schemas",
    "version": "1.0",
    "title": "Orders 2",
    "type": "object",
    "description": "Orders 2",
    "properties": {
        "_id": {
            "title": "Identifier",
            "type": "string",
            "format": "uri-reference",
            "description": "A unique identifier for the time-series event.",
            "meta:xdmType": "string",
            "meta:xdmField": "@id"
        },
        "billing": {
            "title": "Billing Details",
            "type": "object",
            "description": "Billing related information.",
            "properties": {
                "address": {
                    "title": "Billing Address",
                    "description": "Billing Address.",
                    "type": "object",
                    "meta:xdmType": "object",
                    "properties": {
```


4.9. Validate and verify your Schema through the UI

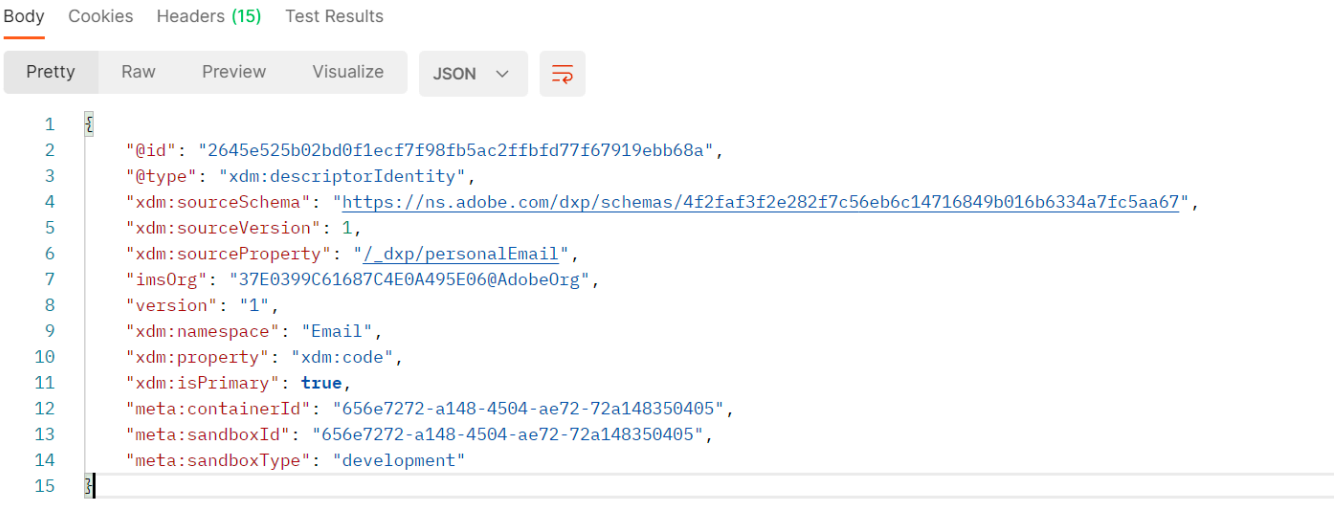


4.10. Create "Primary Identity" Descriptor for "Orders" Schema

1. **Click on "Step 5.1 - Create Primary Identity for Orders Schema"** API call in the **"DEP XDM API Lab"** Folder. **Do not execute it yet.**
2. **Please note the following points:**
   a. You would have identified the **"Primary"** Identities for "Orders" Schema in the SID lab.
   a. Note the use of **"descriptors"** in **https://platform.adobe.io/data/foundation/schemaregistry/tenant/descriptors.**
   b. Schema Properties:
      i. Type - Descriptor type. Its **"xdm:descriptorIdentity"** in this case. You would have gone through the **Schema Descriptors** in the **SID class**.
      ii. Source Schema - The **"$id"** of the **"Orders"** Schema since we are defining an identity for the "Orders" Schema.
      iii. Source Property - Full **"path"** for the Schema property which needs to be marked as an Identity (**"/_dxp/personalEmail"** in this case).
      iv. Name Space - The **"namespace"** to which this identity is associated with. (**"Email"** in this case).
      v. Is Primary - This marked as **"true"** when the identity is Primary. For all other identities, this must be "false" as a schema can only have 1 Primary Identity.
3. A Sample **"POST"** Schema Descriptor call looks like below.



4. A Sample API **"RESPONSE"** looks like below:

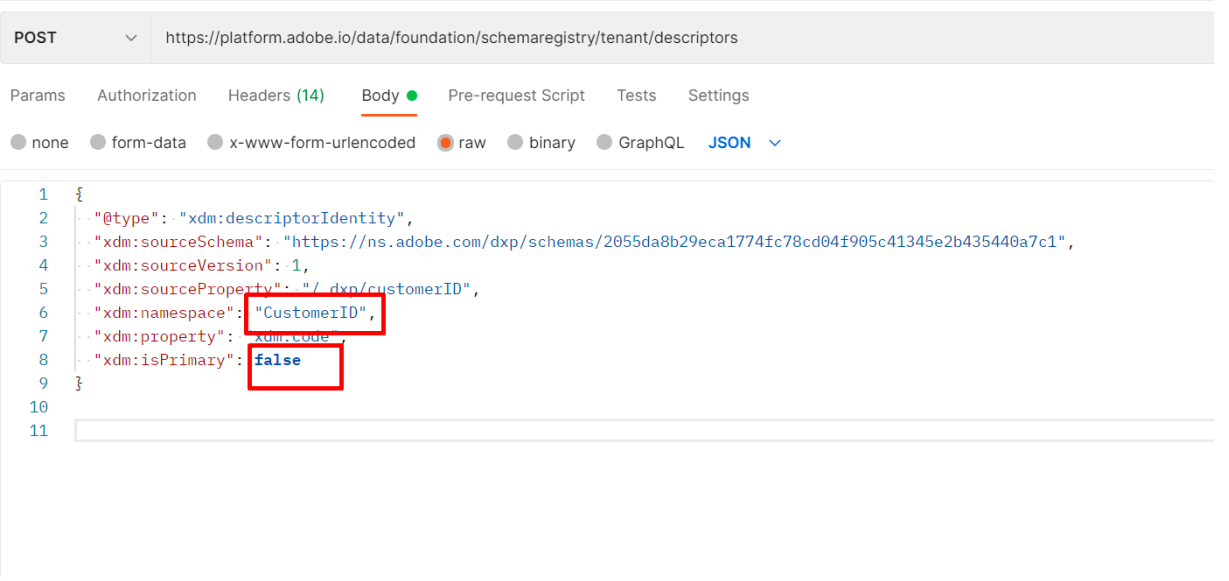


5. Execute **"Step 5.1 - Create Primary Identity for Orders Schema"** by clicking the "Send" button

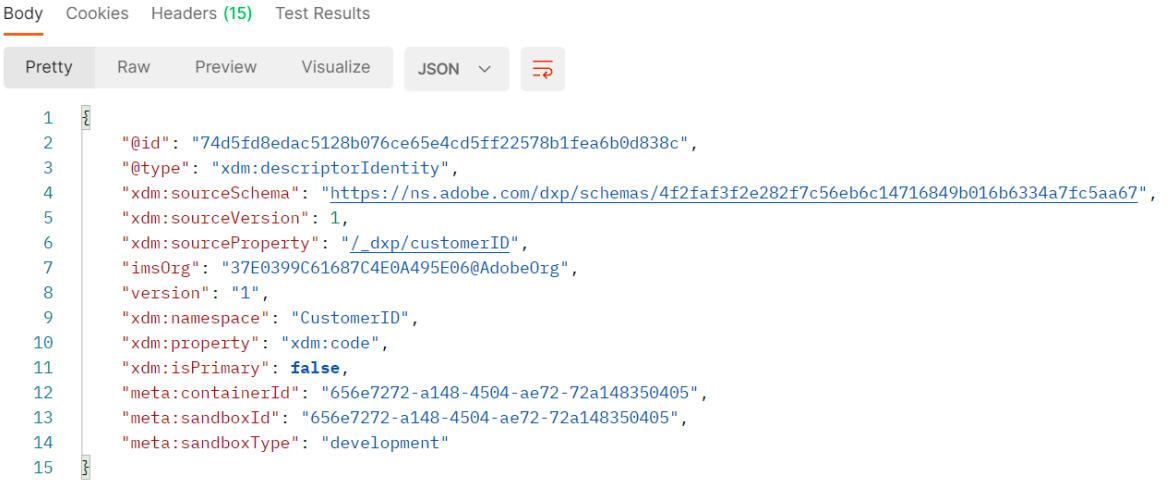
4.11. Create Remaining Identity Descriptor(s) for "Orders" Schema

1. **Click on "Step 5.2 - Create CustomerID Identity for Orders Schema"** API call in the **"DEP XDM API Lab"** Folder. **Do not execute it yet.**
2. **Please note the following points:**
   a. You would have identified the **"Secondary"** Identities for **"Orders"** Schema in the SID lab.
   b. Descriptor Properties:
      i. Type - Descriptor type. Its **"xdm:descriptorIdentity"** in this case
      ii. Source Schema - The **"$id"** of the **"Orders"** Schema since we are defining an identity for the **"Orders"** Schema.
      iii. Source Property - Full "path" for the Schema property which needs to be marked as an Identity (**"/_dxp/customerID"** in this case).
      iv. Source Version - Version number of the Source schema. Always defined as 1
      v. Name Space - The **"namespace"** to which this identity is associated with (**"customerID"** in this case).
      vi. Is Primary - This marked as **"false"** when the identity is non-Primary.
      vii. You can have more than 1 secondary identities for a Schema as long as they are valid and do not create collisions across different individuals.

3. A Sample **"POST"** Schema Descriptor call looks like below.



4. A Sample API **"RESPONSE"** looks like below:



5. Execute **"Step 5.2 - Create CustomerID Identity for Orders Schema"** by clicking the **"Send"** button

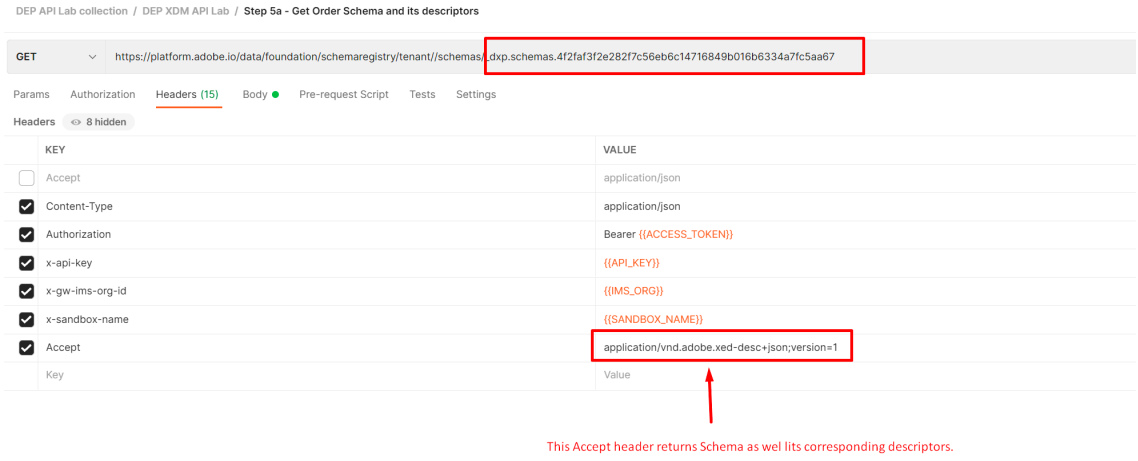
4.12. Browse the "Orders" Schema to see Identities.

1. Click on **"Step 6 - Get Order Schema and its descriptors"** API call in the **"DEP XDM API Lab"** Folder. **Do not execute it yet.**
2. Please note the following points:
   a. A Sample **"GET "** Schema call looks like below.
   b. Observe the use of **"meta:altId"** to fetch the Schema from the registry.
   c. Also observe the changed "Accept" header in this case. This accept header will return the Schemas along with its associated descriptors in the API response. This tells the schema registry what to return back in the API Response. You can see other **"Accept"** header options here **Schema Registry APIs**
   d. With the pre-loaded Accept Header in the postman collection, you will see the fully exploded hierarchical view of the Orders Schema where every single

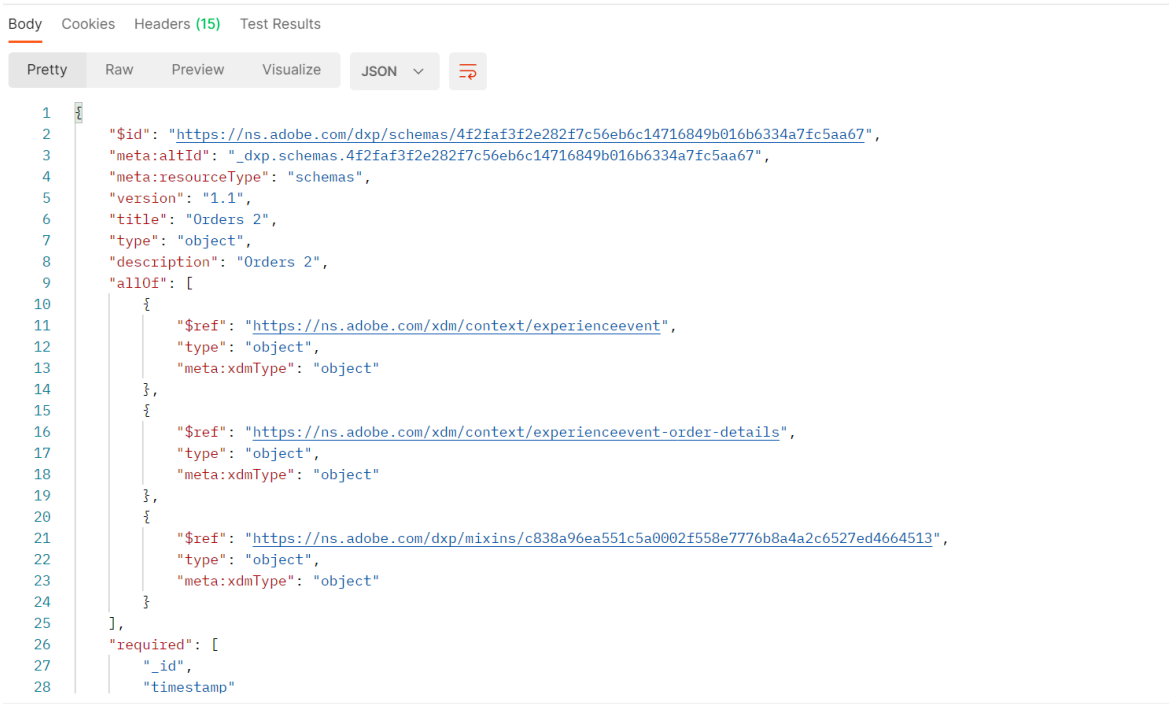
property belonging to a class or a field group will be exposed.

e. All the object properties will be exploded all the way to their leaf nodes.

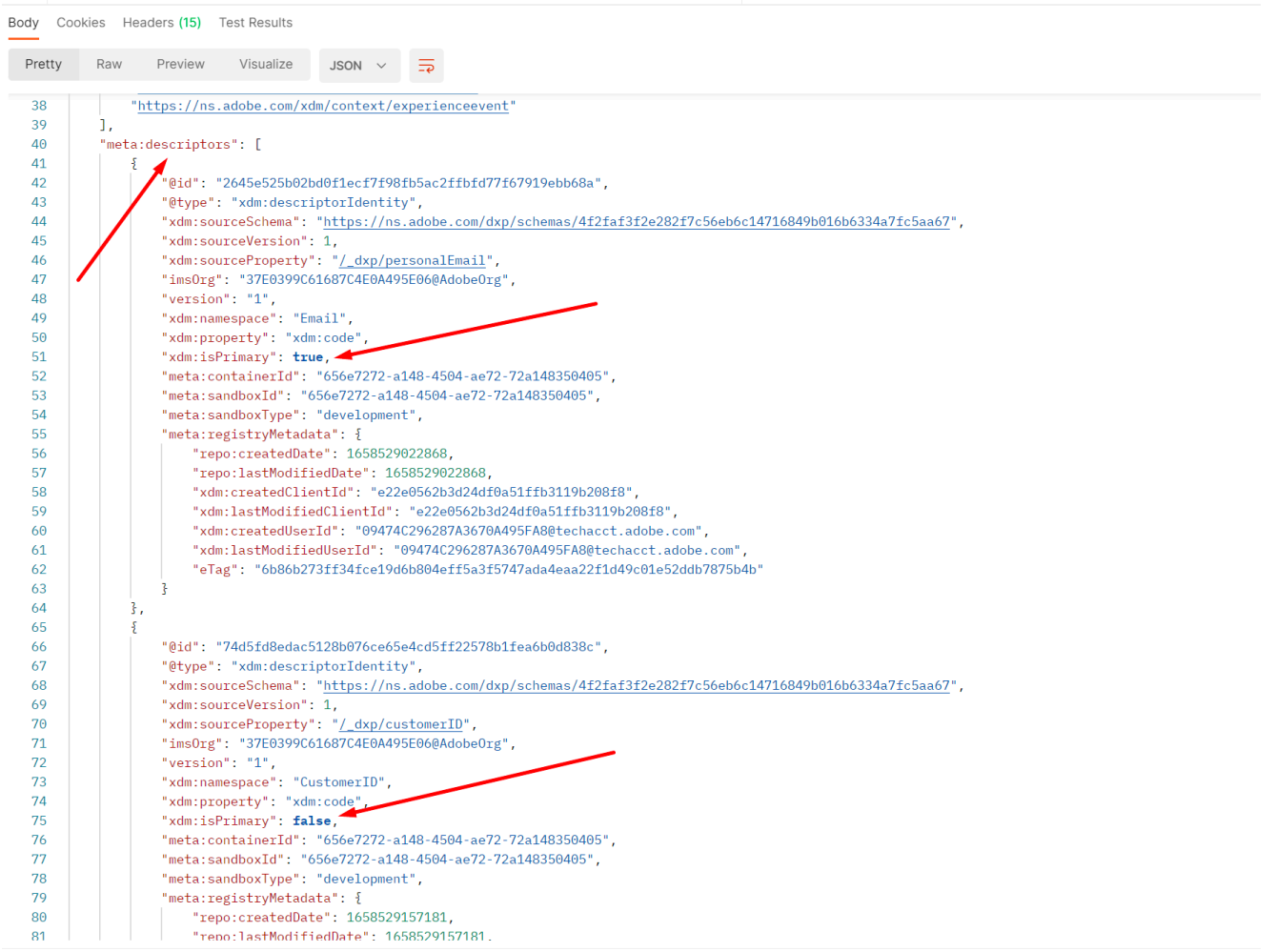
3. A Sample **"GET"** Schema Descriptor call looks like below.



4. A Sample API **"RESPONSE"** looks like below:

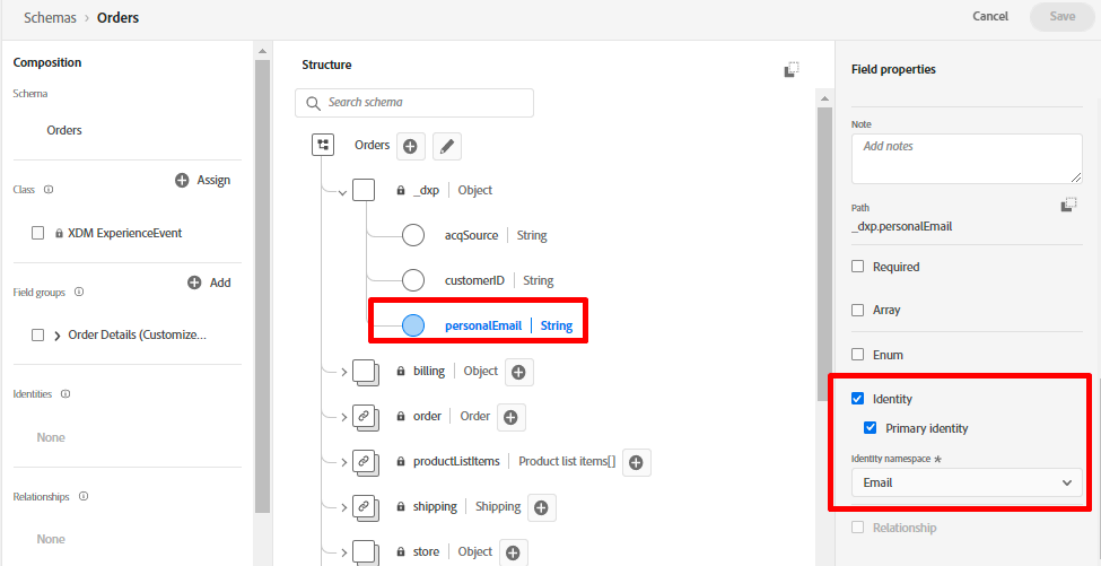


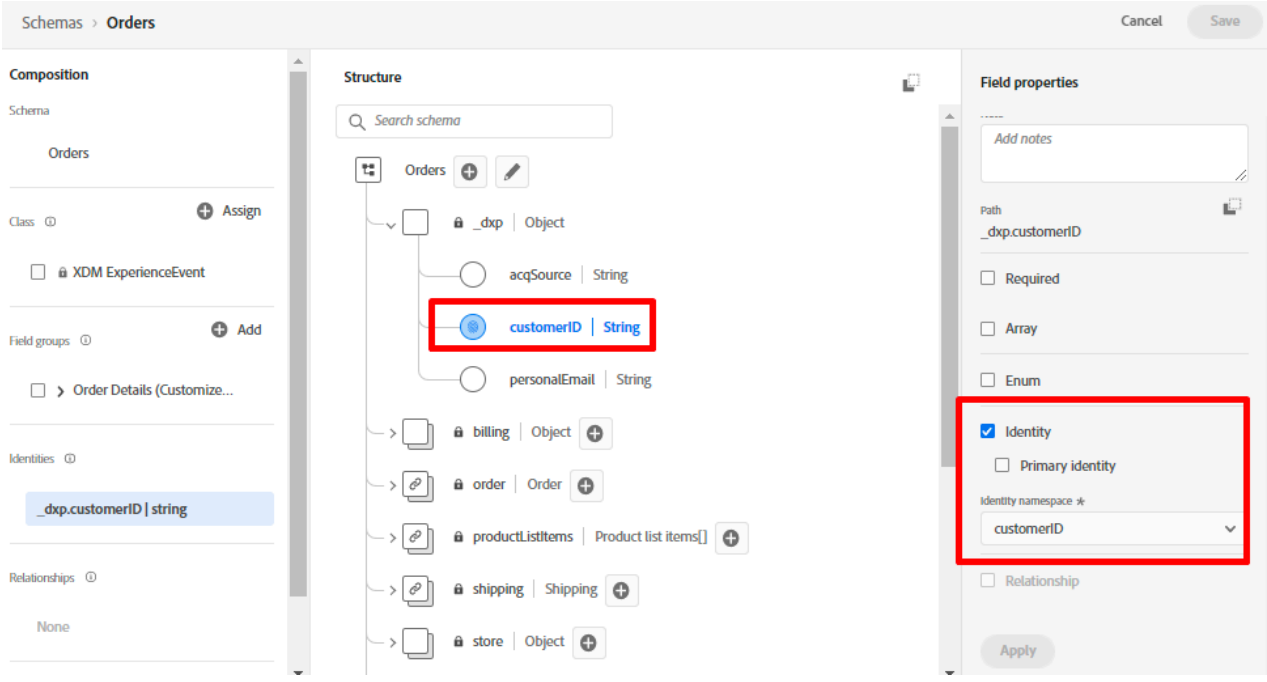
5. A Sample API **"RESPONSE"** looks like below (Browse down in the response to look for Identity Descriptors for this Schema):



6. Execute **"Step 6 - Get Order Schema and its descriptors"** by clicking the "Send" button

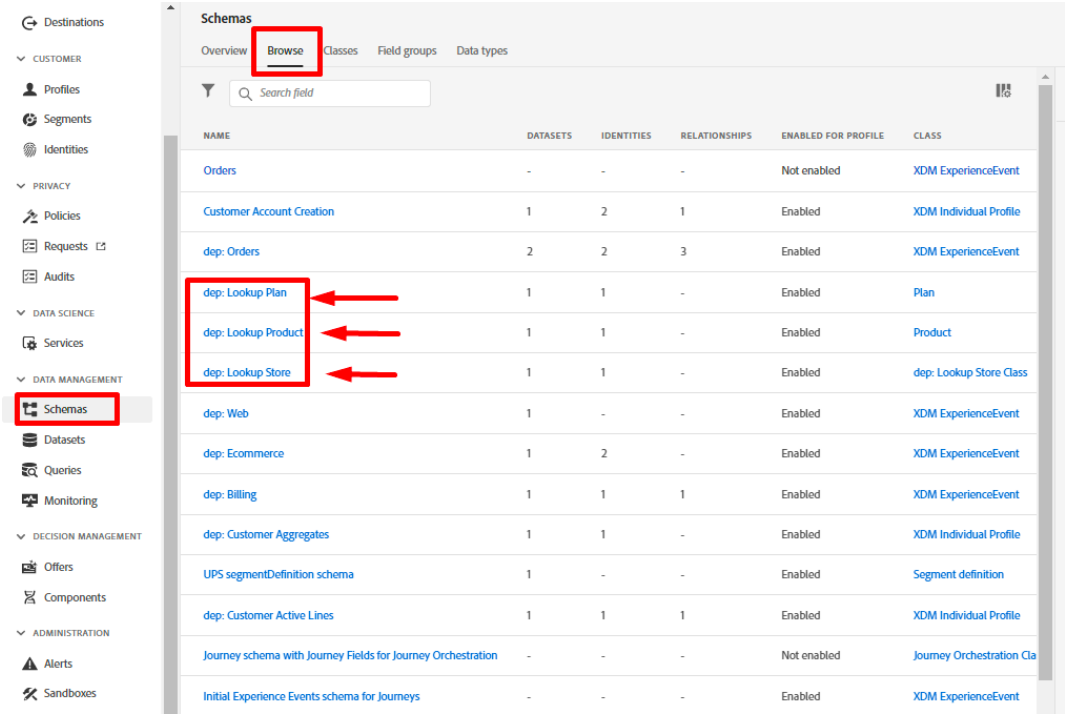
4.13. Validate and verify your Schema through the UI





4.14. Get the Schema ID for the Lookup Schemas

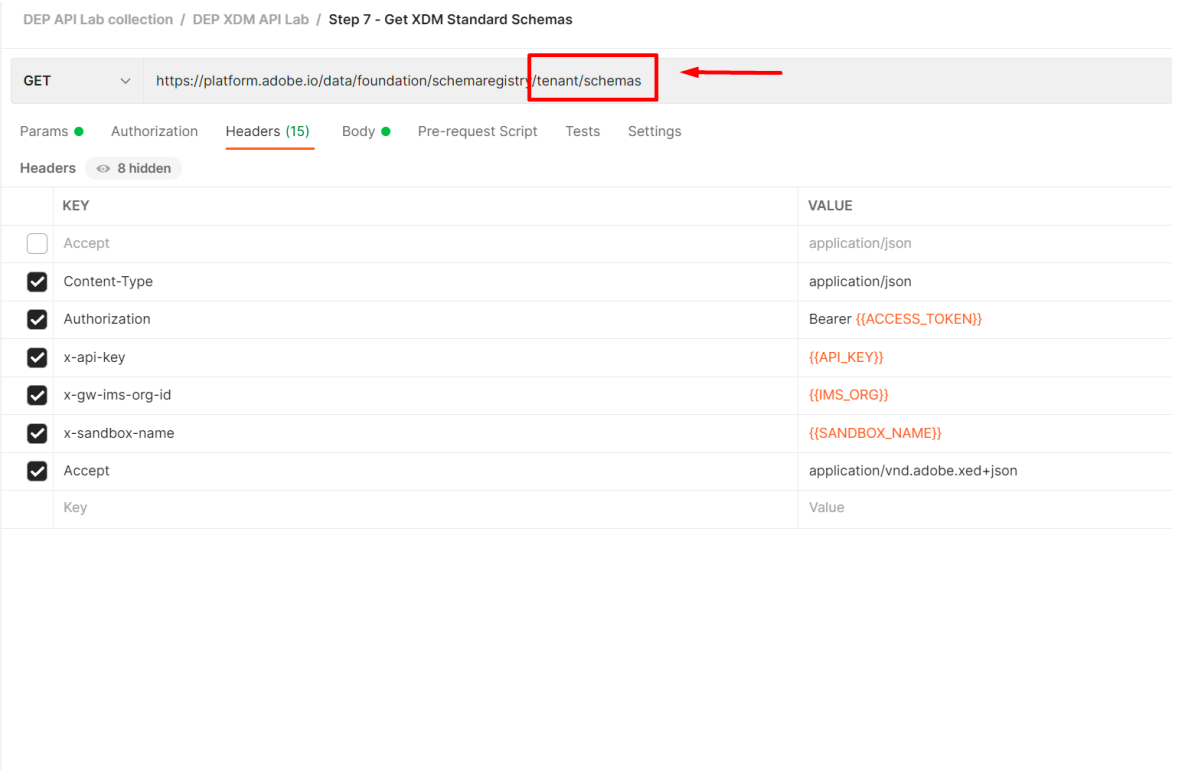
1. **Execute the "Step 7 - Get Lookup Schemas"** API call in the **"DEP XDM API Lab"** Folder.
2. Within your response from **Step 7** above
    a. Look for the Product Schema (Search keyword "**dep: Lookup Product**")
    b. Look for the Store Schema (Search keyword "**dep: Lookup Store**")
    c. Look for the Plan Schema (Search keyword "**dep: Lookup Plan**")
    d. You can also get the Schema names from UI by browsing the Schemas



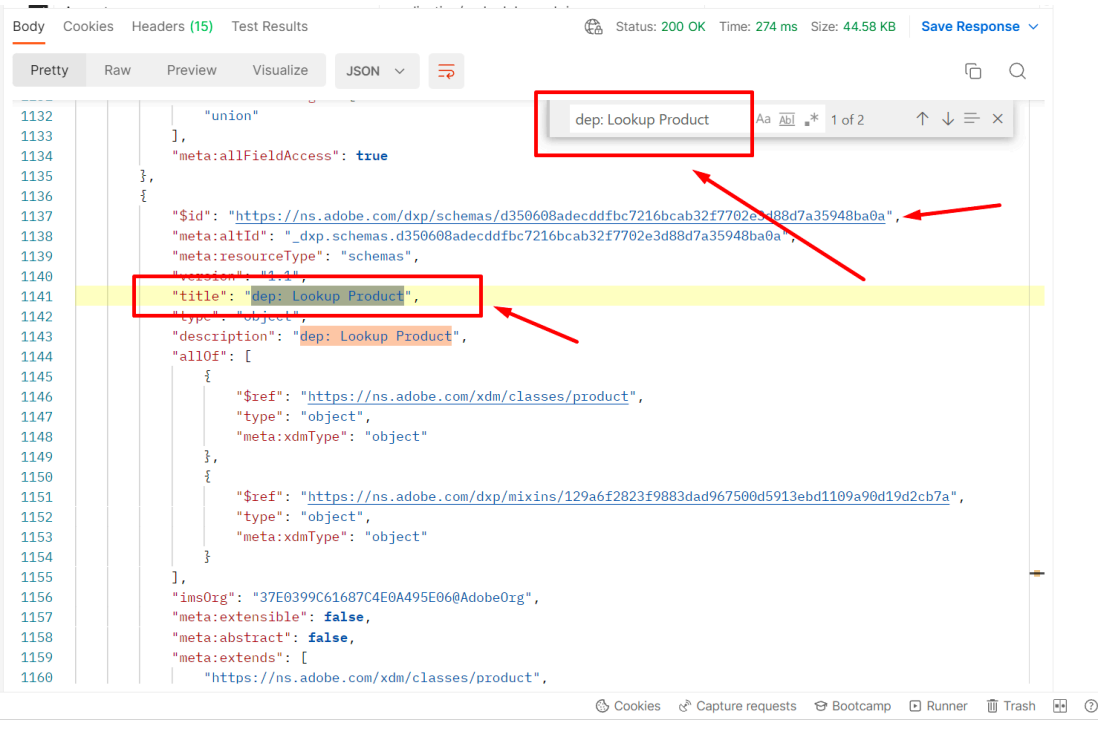
Please note the following points:

i. These three schemas should be pre-deployed on your Sandbox.
ii. Relationship Descriptors will be created from Orders schema to each of these three lookup schemas.
iii. Copy the **"$id"** of these three schemas and have them handy with you. They will be required in CREATE/POST Relationship descriptor API calls.

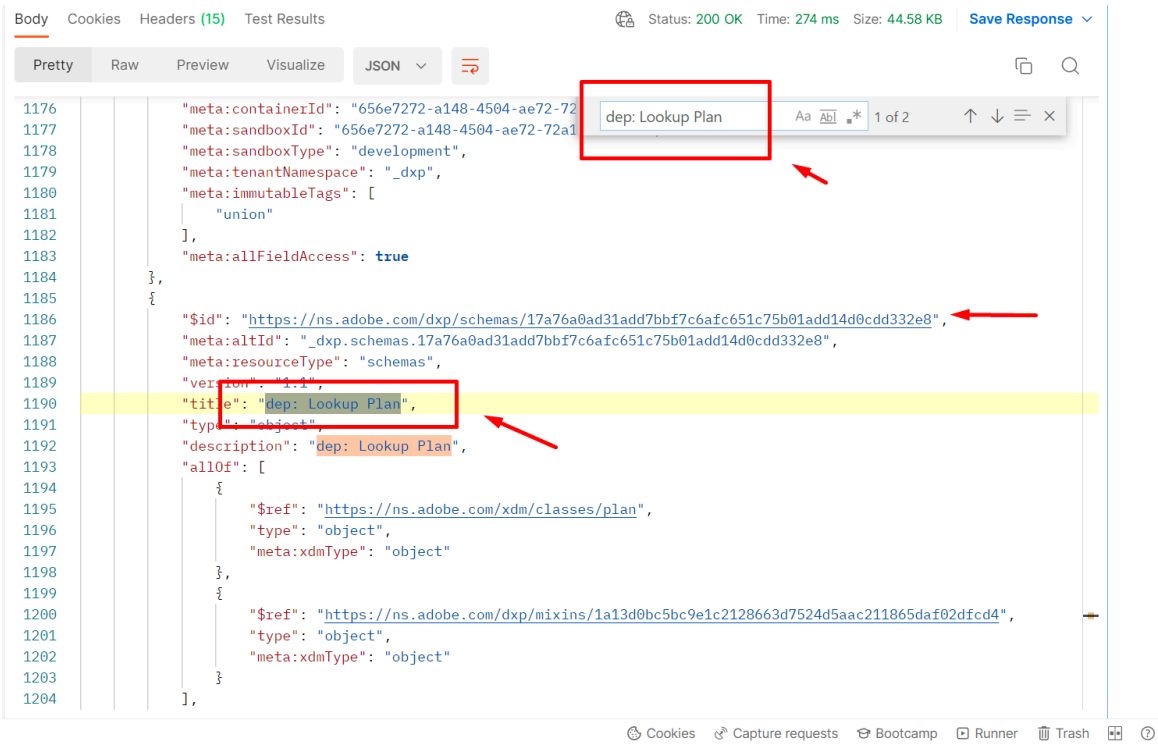
3. A Sample **"GET"** Schema call looks like below.



4. A Sample API **"RESPONSE"** looks like below (Search for Product Schema):

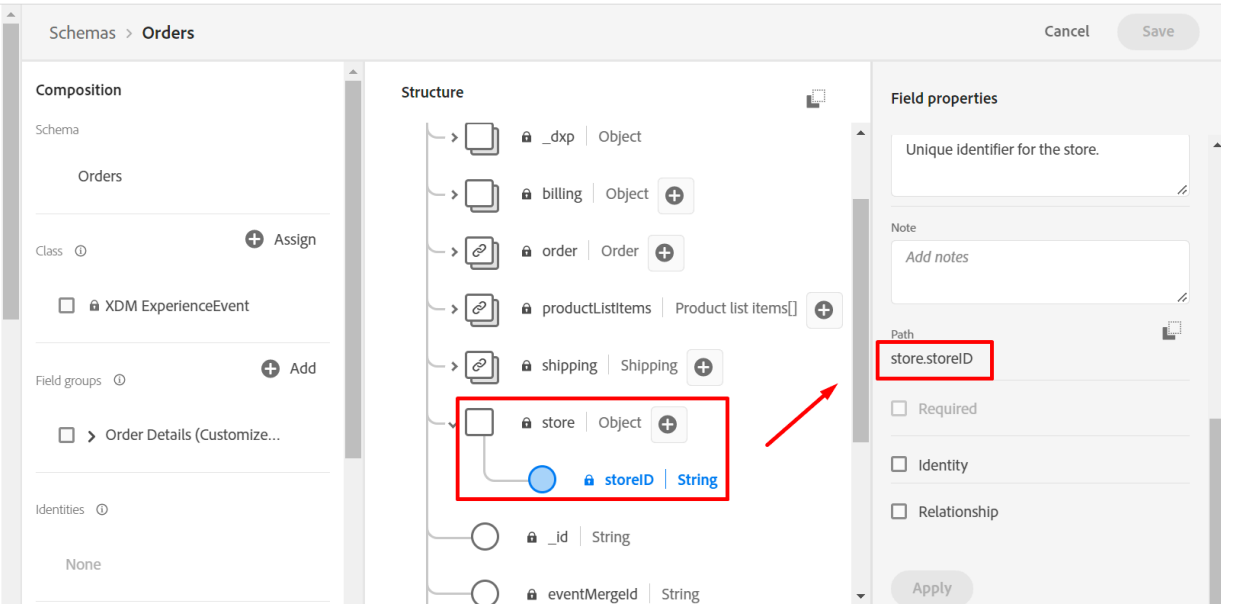


5. A Sample API **"RESPONSE"** looks like below (Search for Plan Schema):



4.15. Create "Orders to Store" Relationship Descriptor for "Orders" Schema

1. **Click on "Step 8 - Relationship Descriptor Orders To Store"** API call in the **"DEP XDM API Lab"** Folder. **Do not execute it yet.**
2. **Please note the following points:**
   a. You would have identified the **"relationships"** from **"Orders"** Schema in the **SID lab**.
   b. "Orders : Store" relationship cardinality is "**M:1**" .
   c. A relationship descriptor is always defined from either the **"Profile"** or the **"Experience Event"** class based schemas (Experience Event class in this case).
   d. It's worth noting that AEP only supports **1 hop join** from Profile or EE schemas i.e. we can only create **1 level lookup** relationships.
   e. You should have the **"$id"** of the Store Schema.
   f. Descriptor Properties:
      i. Type - Descriptor type. Its **"xdm:descriptorOneToOne"** in this case
         1. Even though the descriptor type refers to **OneToOne** , we use this to define relationships for **M:1 cardinality** as well.
      ii. Source Schema - The "**$id**" of the schema, the relationship is originating from (**"Orders"** Schema in this case). Please note that you can create a descriptor from either side of the relationship but Profile services will only look for relationships originating from either the "profile" class based schemas or "Experience Event" Class based schemas.
      iii. Source Property - Full "**path**" for the Schema property which needs to be marked as a Relationship (**"/store/storeID"** in this case). You could get the fully qualified path of a schema property like below



iv. Source Version - Version number of the Source schema. Always defined as **1**

v. Destination Schema - The **"$id"** of the schema, the relationship is referring to (**"Store"** Schema in this case).

vi. Destination Property - This is an optional parameter. A relationship descriptor assumes that the source property , the relation is defined on is marked as a primary identity in the Destination Schema.

vii. Destination Version - Version number of the destination schema. Always defined as **1**

3. A Sample **"POST"** Relationship Descriptor call looks like below.

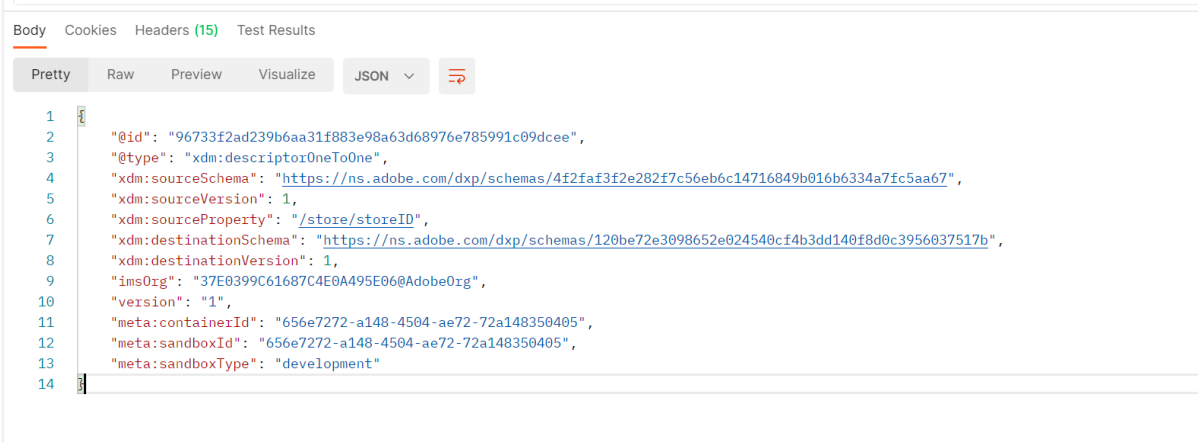
POST https://platform.adobe.io/data/foundation/schemaregistry/tenant/descriptors

Params Authorization Headers (14) Body Pre-request Script Tests Settings

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

```
{
  "@type": "xdm:descriptorOneToOne",
  "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
  "xdm:sourceVersion": 1,
  "xdm:sourceProperty": "/store/storeID",
  "xdm:destinationSchema": "https://ns.adobe.com/dxp/schemas/120be72e3098652e024540cf4b3dd140f8d0c3956037517b",
  "xdm:destinationVersion": 1
}
```

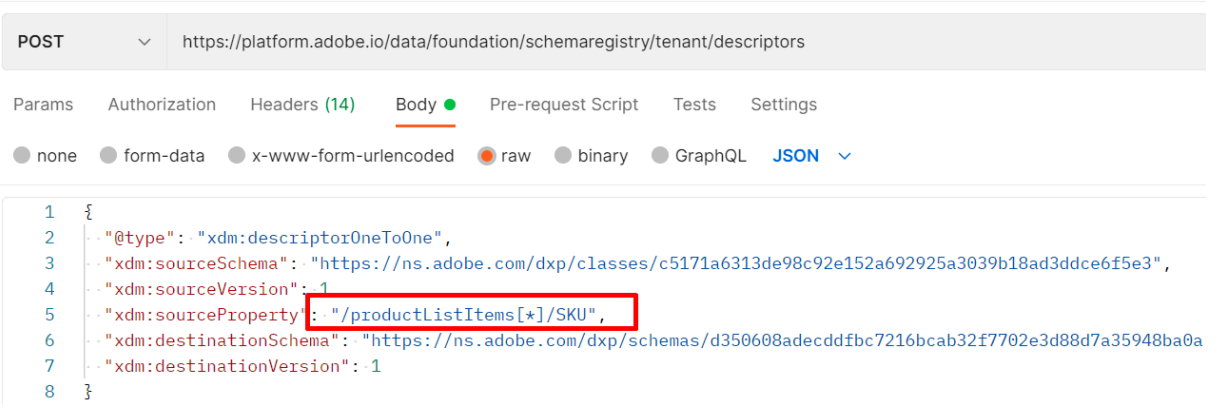
4. A Sample API **"RESPONSE"** looks like below:



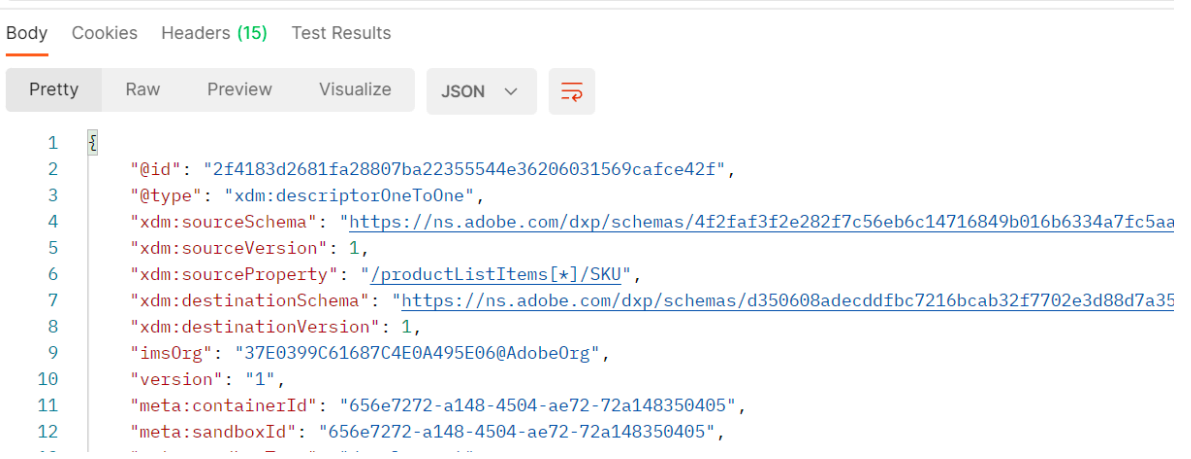
5. Execute **"Step 8 - Relationship Descriptor Orders To Store"** by clicking the "Send" button

4.16. Create "Orders to Product" Relationship Descriptor for "Orders" Schema

1. **Click on "Step 9 - Relationship Descriptor Orders To Product"** API call in the **"DEP XDM API Lab"** Folder. **Do not execute it yet.**
2. **Please note the following points:**
    g. Descriptor Properties:
        i. Type - Descriptor type. Its **"xdm:descriptorOneToOne"** in this case
        ii. Source Schema - The **"$id"** of the schema, the relationship is originating from (**"Orders"** Schema in this case).
        iii. Source Property - Full **"path"** for the Schema property which needs to be marked as a Relationship (**"/productListItems[*]/SKU**" in this case).
            1. Observe the property path here when the property belongs to an array of objects. The array is referenced as **ArrayNa me[*]**.
        iv. Source Version - Version number of the Source schema. Always defined as **1**
        v. Destination Schema - The "**$id**" of the schema, the relationship is referring to (**"Product"** Schema in this case).
        vi. Destination Property - This is an optional parameter. A relationship descriptor assumes that the source property , the relation is defined on is marked as a primary identity in the Destination Schema.
        vii. Destination Version - Version number of the destination schema. Always defined as **1**
3. A Sample **"POST"** Relationship Descriptor call looks like below.



4. A Sample API **"RESPONSE"** looks like below:



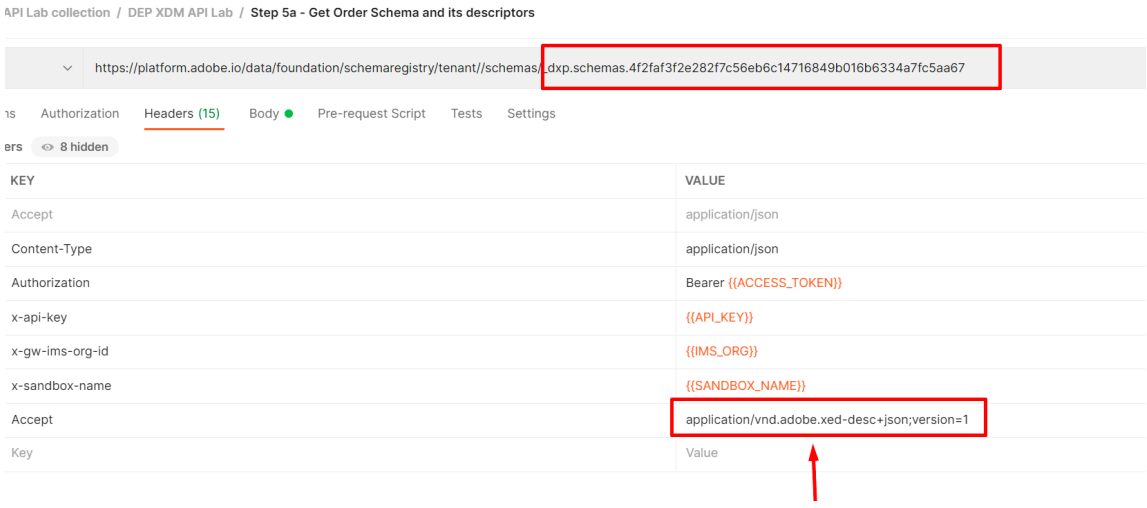
5. Execute **"Step 9 - Relationship Descriptor Orders To Product"** by clicking the "Send" button

4.17. Create "Orders to Plan" Relationship Descriptor for "Orders" Schema

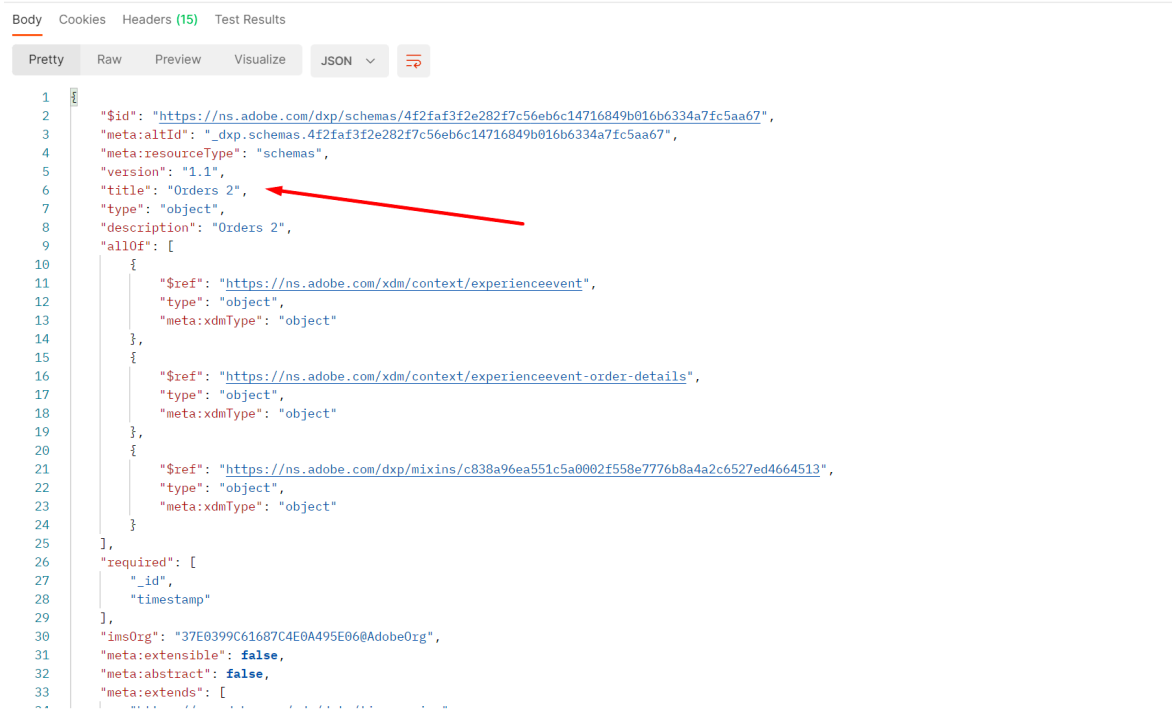
1. Execute **"Step 10 - Relationship Descriptor Orders To Plan"** by clicking the "Send" button

4.18. Browse the "Orders" Schema to see your Relationship Descriptors.

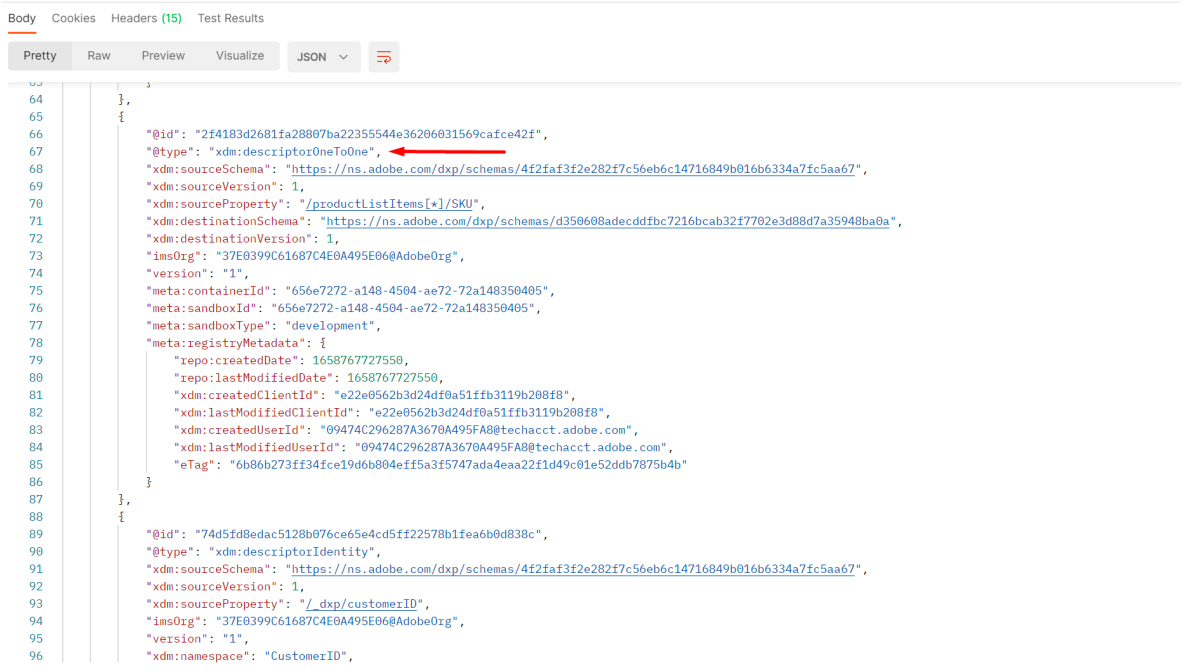
1. Execute the **"Step 11 - Get Order Schema and its descriptors"** by clicking the **"Send"** button
2. A Sample **"GET"** Schema Descriptor call looks like below.



3. A Sample API **"RESPONSE"** looks like below:



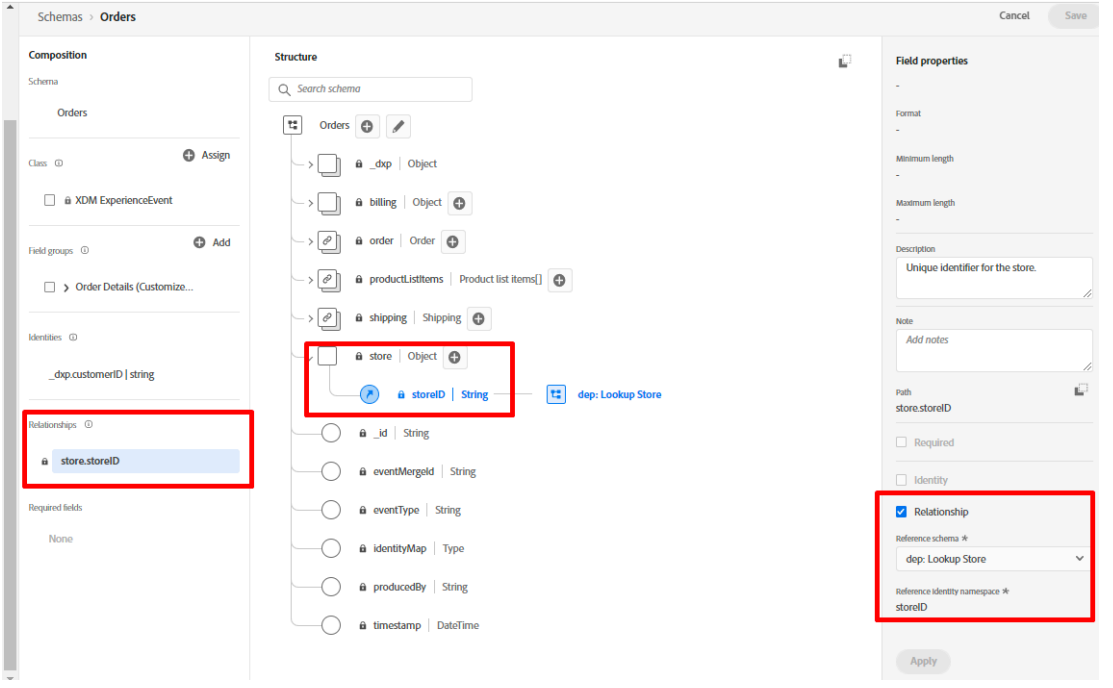
4. A Sample API **"RESPONSE"** looks like below (Browse down in the response to look for Relationship Descriptors for this

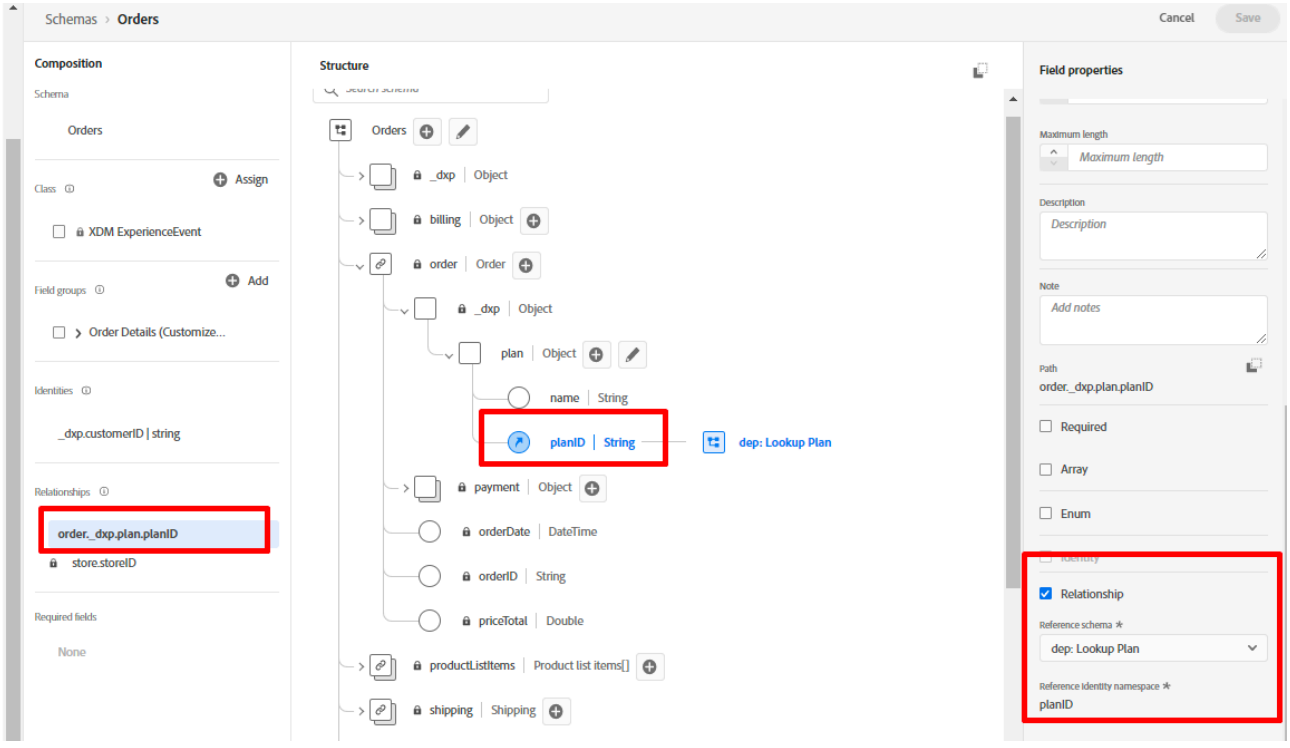


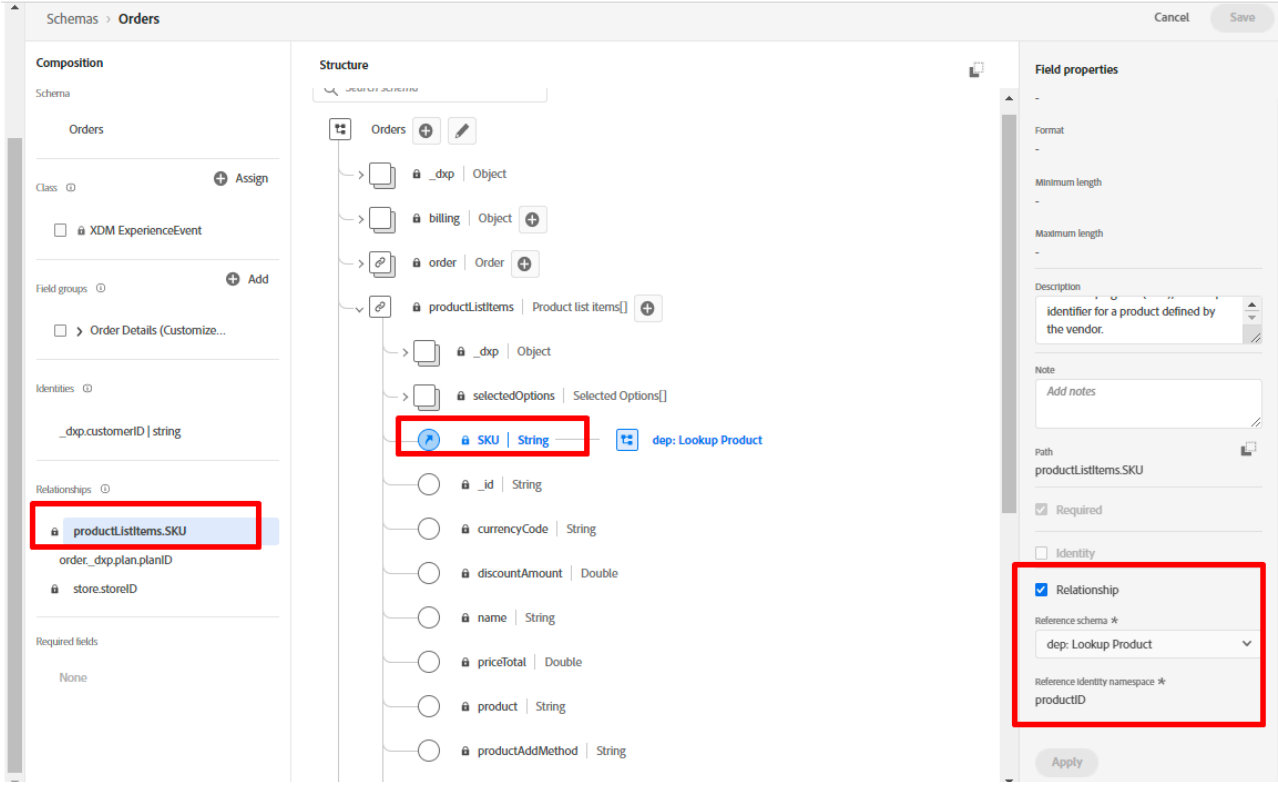

```
    },
    {
        "@id": "2f4183d2681fa28807ba22355544e36206031569cafce42f",
        "@type": "xdm:descriptorOneToOne",
        "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
        "xdm:sourceVersion": 1,
        "xdm:sourceProperty": "/productListItems[*]/SKU",
        "xdm:destinationSchema": "https://ns.adobe.com/dxp/schemas/d35060badecddfbc7216bcab32f7702e3d88d7a35948ba0a",
        "xdm:destinationVersion": 1,
        "imsOrg": "37E0399C61687C4E0A495E06@AdobeOrg",
        "version": "1",
        "meta:containerId": "656e7272-a148-4504-ae72-72a148350405",
        "meta:sandboxId": "656e7272-a148-4504-ae72-72a148350405",
        "meta:sandboxType": "development",
        "meta:registryMetadata": {
            "repo:createdDate": 1658767727550,
            "repo:lastModifiedDate": 1658767727550,
            "xdm:createdClientId": "e22e0562b3d24df0a51ffb3119b208f8",
            "xdm:lastModifiedClientId": "e22e0562b3d24df0a51ffb3119b208f8",
            "xdm:createdUserId": "09474C296287A3670A495FA8@techacct.adobe.com",
            "xdm:lastModifiedUserId": "09474C296287A3670A495FA8@techacct.adobe.com",
            "eTag": "6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b"
        }
    },
    {
        "@id": "74d5fd8edac5128b076ce65e4cd5ff22578b1fea6b0d838c",
        "@type": "xdm:descriptorIdentity",
        "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
        "xdm:sourceVersion": 1,
        "xdm:sourceProperty": "/_dxp/customerID",
        "imsOrg": "37E0399C61687C4E0A495E06@AdobeOrg",
        "version": "1",
        "xdm:namespace": "CustomerID",
```


5. Execute **"Step 6 - Get Order Schema and its descriptors"** by clicking the "Send" button

4.19. Validate and verify your schema through the UI







4.20. Create "Store" Reference Identity Descriptor for "Orders" Schema

1. **Click on "Step 12 - Reference Descriptor for Store"** API call in the **"DEP XDM API Lab"** Folder. **Do not execute it yet.**
2. **Please note the following points:**
   a. You would have identified the **"relationships"** from **"Orders"** Schema in the **SID lab.**
   b. Reference Identity descriptors are created automatically from the backend when you create relationships from Schema UI. You have to create them explicitly when

you create schema using API calls.

c. A reference descriptor is required to be defined for the **"Source Property"** in each **"Relationship descriptor".**

d. Descriptor Properties:

i. Type - Descriptor type. Its **"xdm:descriptorReferenceIdentity"** in this case

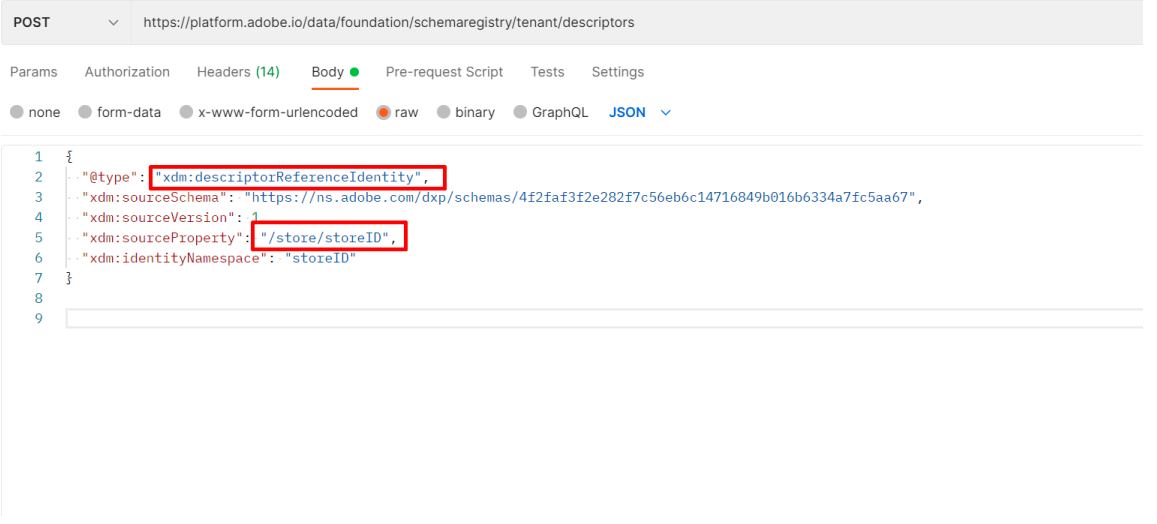
ii. Source Schema - The **"$id"** of the schema, the relationship is originating from (**"Orders"** Schema in this case).

iii. Source Property - Full **"path"** for the Schema property which needs to be marked as a reference identity (**"/store/storeID"** in this case).

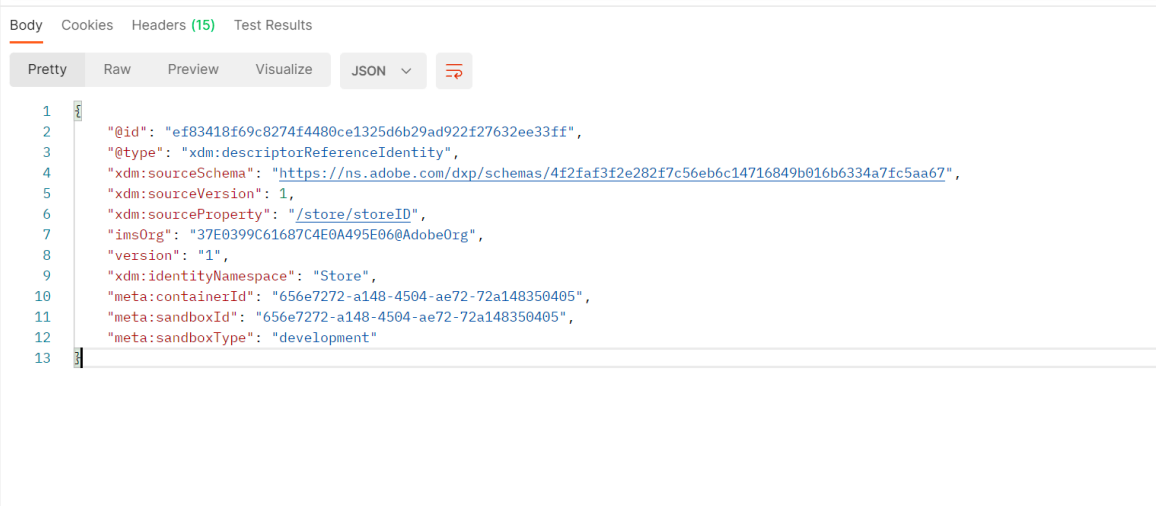
iv. Source Version - Version number of the Source schema. Always defined as 1

v. Identity Namespace- The namespace for the Source Schema Property (**"storeID"** in this case)

3. A Sample **"POST"** Relationship Descriptor call looks like below.



4. A Sample API **"RESPONSE"** looks like below:



5. Execute the **"Step 12 - Reference Descriptor for Store"** by clicking the "Send" button

4.21. Create "Product" Reference Identity Descriptor for "Orders" Schema

1. **Please note the following points:**

    a. Descriptor Properties:
        i. Type - Descriptor type. Its **"xdm:descriptorReferenceIdentity"** in this case
        ii. Source Schema - The **"$id"** of the schema, the relationship is originating from (**"Orders"** Schema in this case).
        iii. Source Property - Full **"path"** for the Schema property which needs to be marked as a reference identity(**"/productListItems[*]/SKU"** in this case).
        iv. Source Version - Version number of the Source schema. Always defined as **1**
        v. Identity Namespace- The namespace for the Source Schema Property (**"productID"** in this case)

2. Execute the **"Step 13 - Reference Descriptor for Product"** by clicking the "Send" button

4.22. Create "Plan" Reference Identity Descriptor for "Orders" Schema

1. **Please note the following points:**
    a. Descriptor Properties:
        i. Type - Descriptor type. Its **"xdm:descriptorReferenceIdentity"** in this case
        ii. Source Schema - The **"$id"** of the schema, the relationship is originating from ("Orders" Schema in this case).
        iii. Source Property - Full **"path"** for the Schema property which needs to be marked as a reference identity(**"/order/_dxp/plan/planID"** in this case).
        iv. Source Version - Version number of the Source schema. Always defined as **1**
        v. Identity Namespace- The namespace for the Source Schema Property (**"planID"** in this case)
2. Execute the **"Step 14 - Reference Descriptor for Plan"** by clicking the **"Send"** button

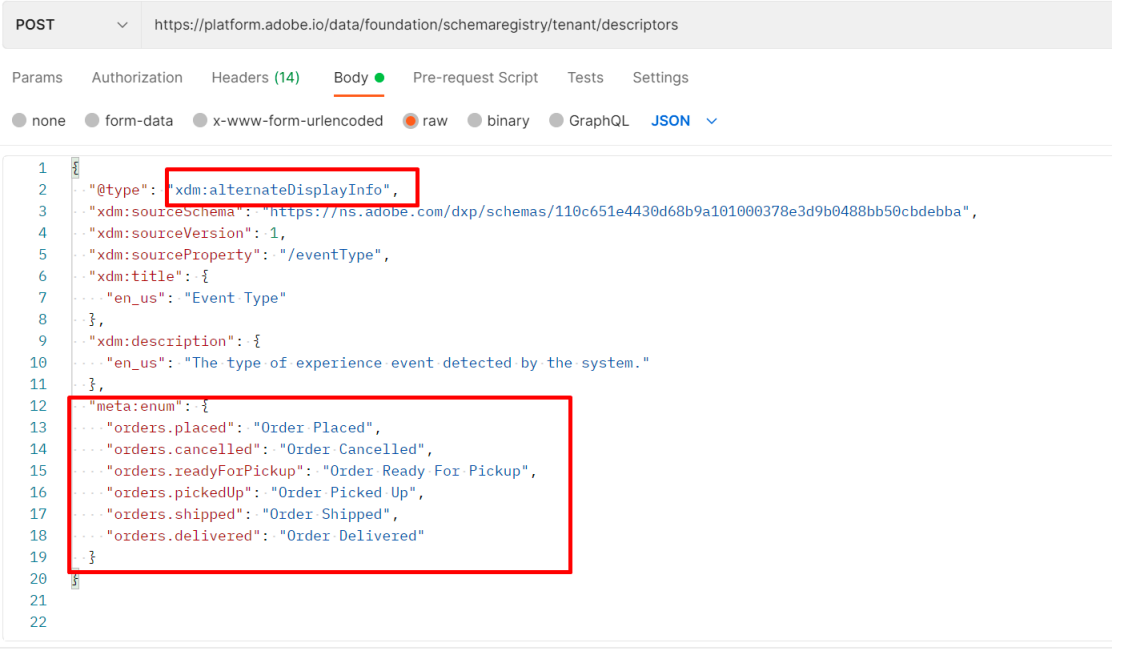
4.23. Browse the "Orders" Schema to see your Reference Descriptors.

1. Execute the **"Step 15 - Get Order Schema and its descriptors"** by clicking the **"Send"** button
2. You should be able to browse the following Schema descriptors:
    b. Identities (Primary/Secondary)
    c. Relationships
    d. Reference Identities

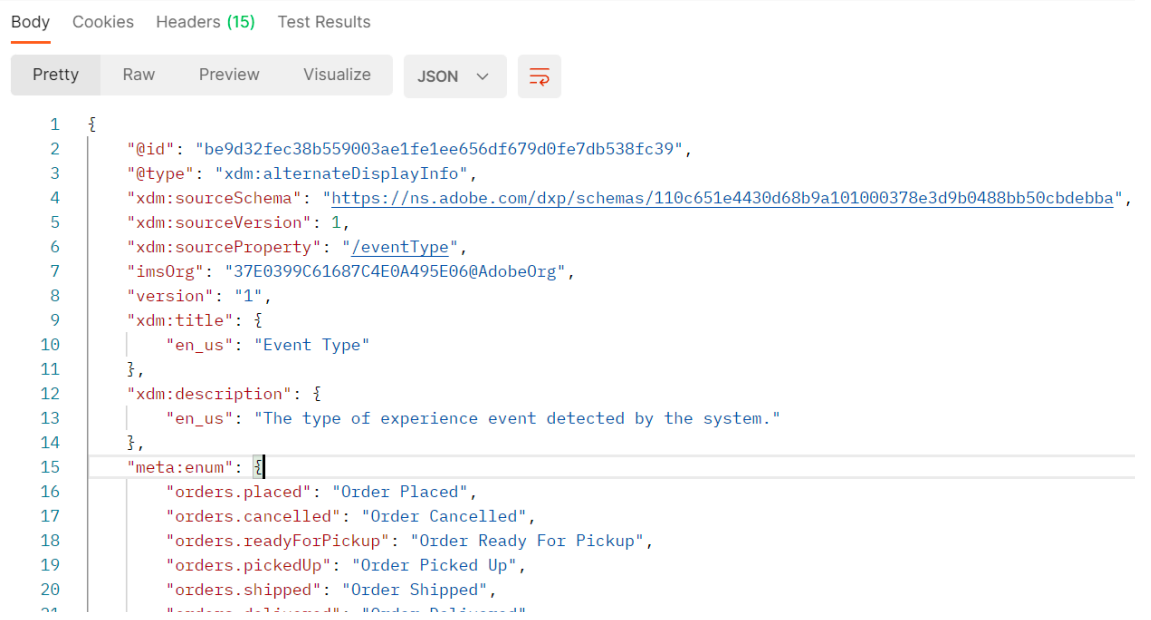
4.24. Extend Event Types for Order Schema

1. **Click on "Step 16 - Extend the Event Types"** API call in the **"DEP XDM API Lab"** Folder. **Do not execute it yet.**

2. **Please note the following points:**
    a. Standard XDM Event Types are provided by the Experience Event Class
    b. If the customers need to add more event types as per their use cases, they can do that using the API calls (UI enablement is in progress to do the same).
    c. "Orders" data will be sending the following event types to the platform:
        i. Order Placed
        ii. Order Cancelled
        iii. Order Ready For Pickup
        iv. Order Picked Up
        v. Order Shipped
        vi. Order Delivered
3. A Sample **"POST"** Descriptor call to add event types looks like below.

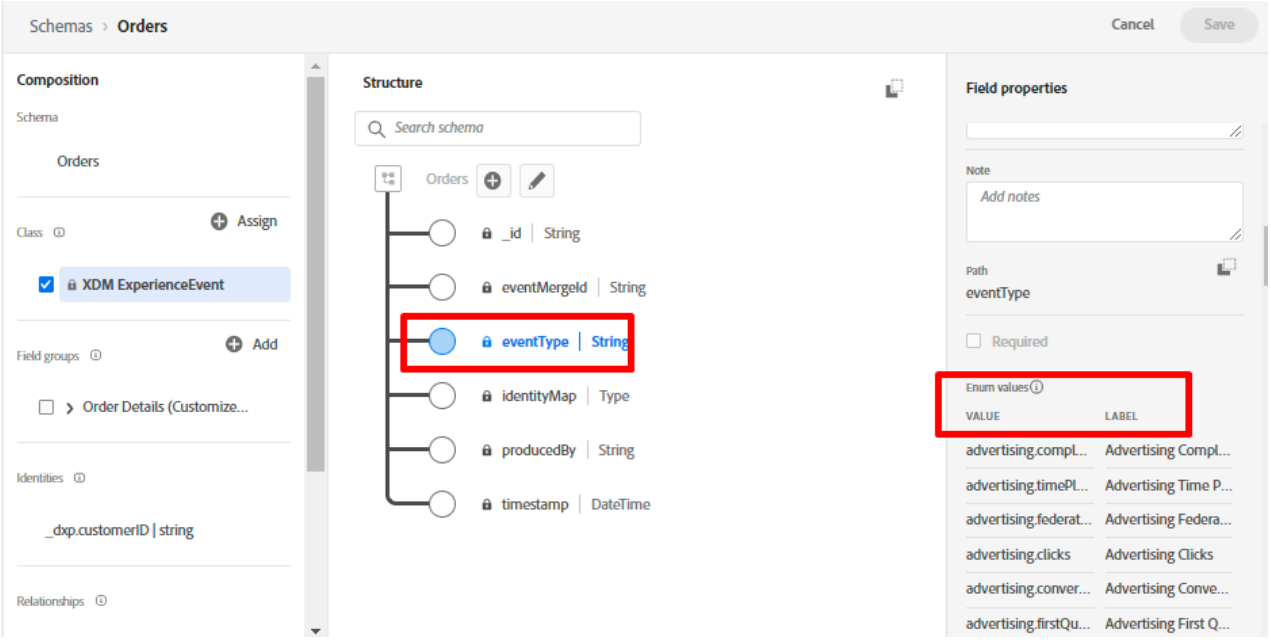


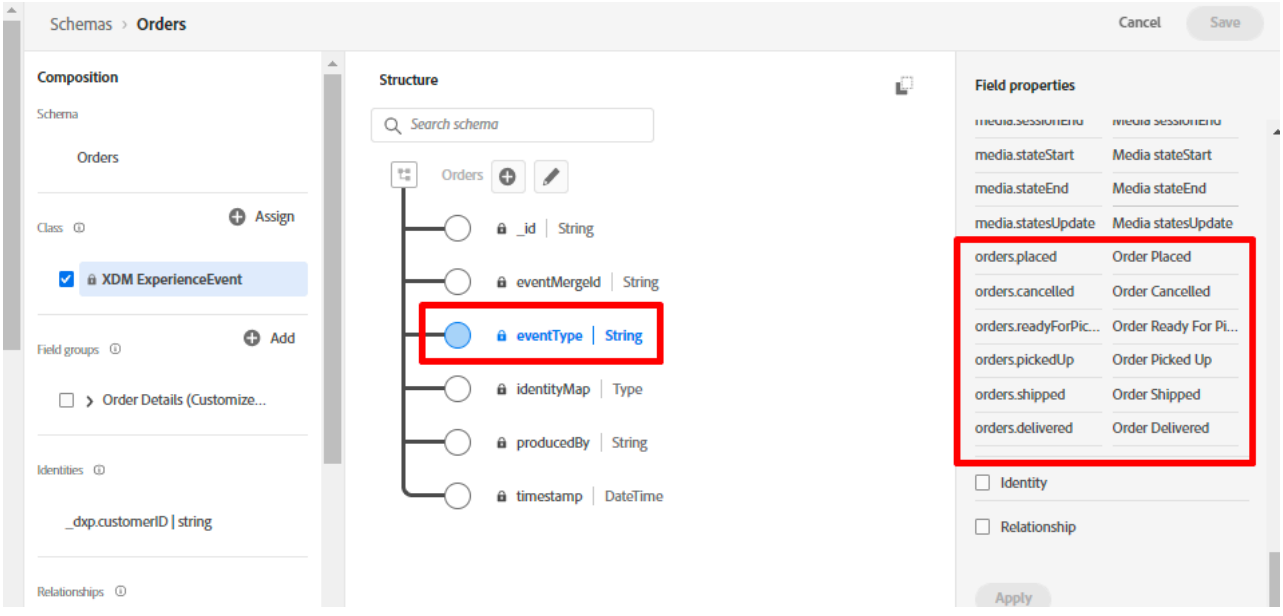
4. A Sample API **"RESPONSE"** looks like below:



5. Execute the **"Step 16 - Extend the Event Types"** by clicking the "Send" button

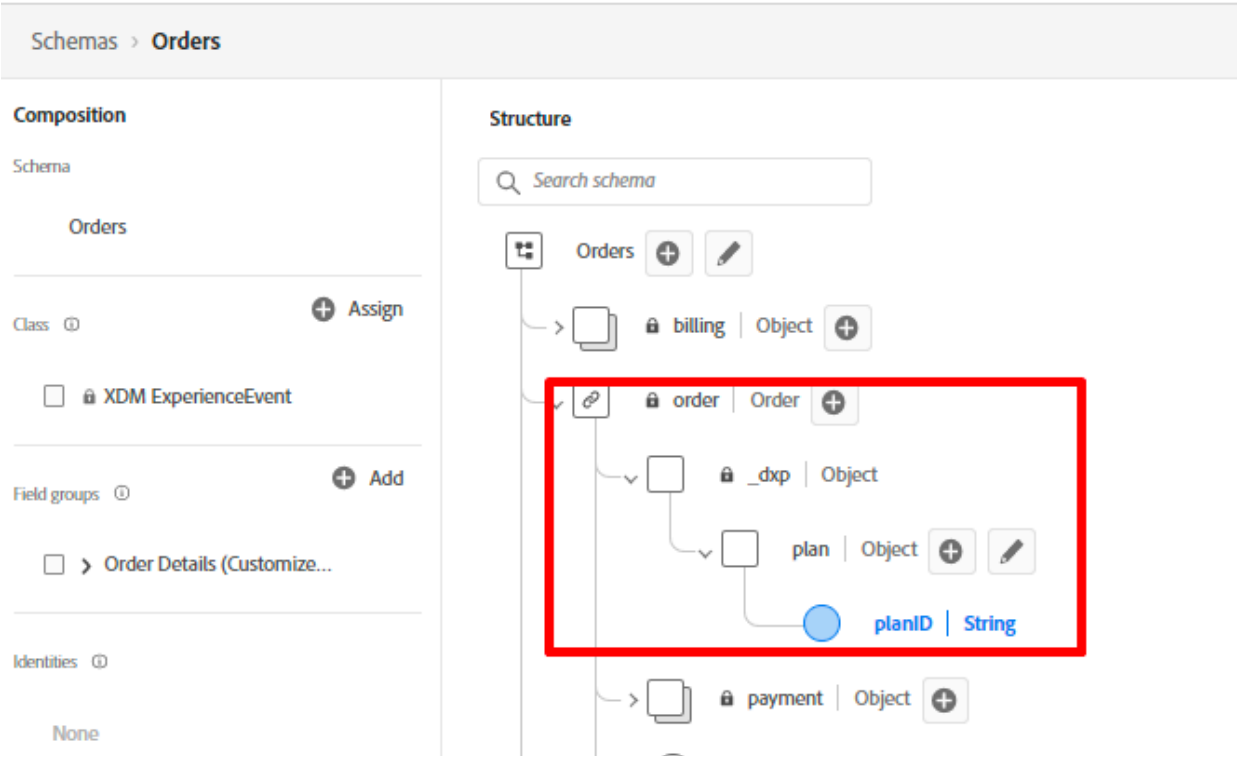
4.25. Verify and validate your schema through UI



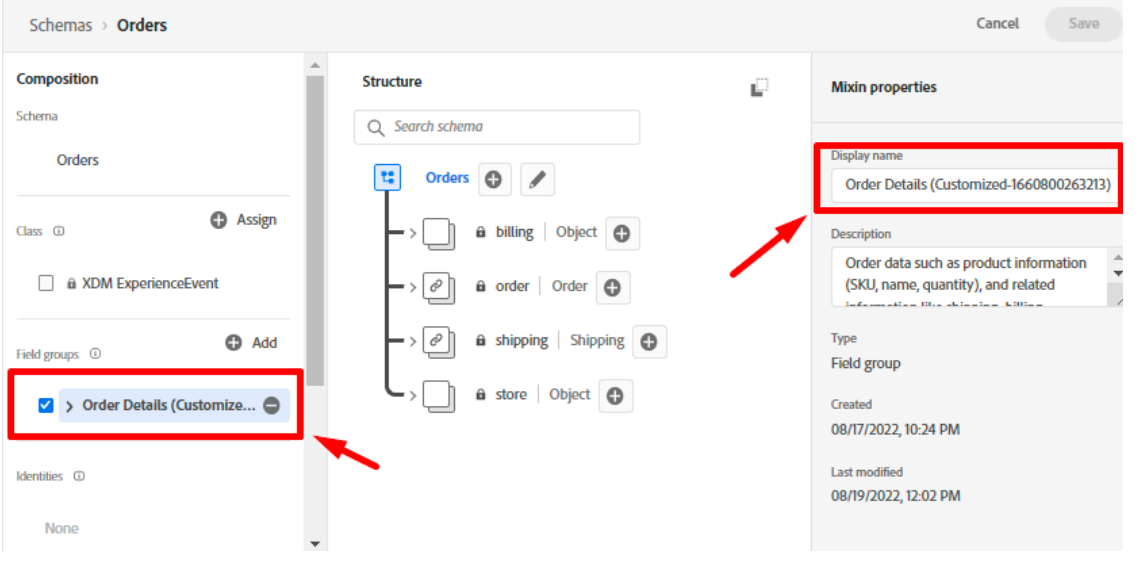


4.26. JSON Patch - Modify Order schema to add a missing property

1. Assume we forgot to add a custom field **"name"** under **order==>_dxp==>plan** object in Orders Schema. UI Schema screen shot below shows that.

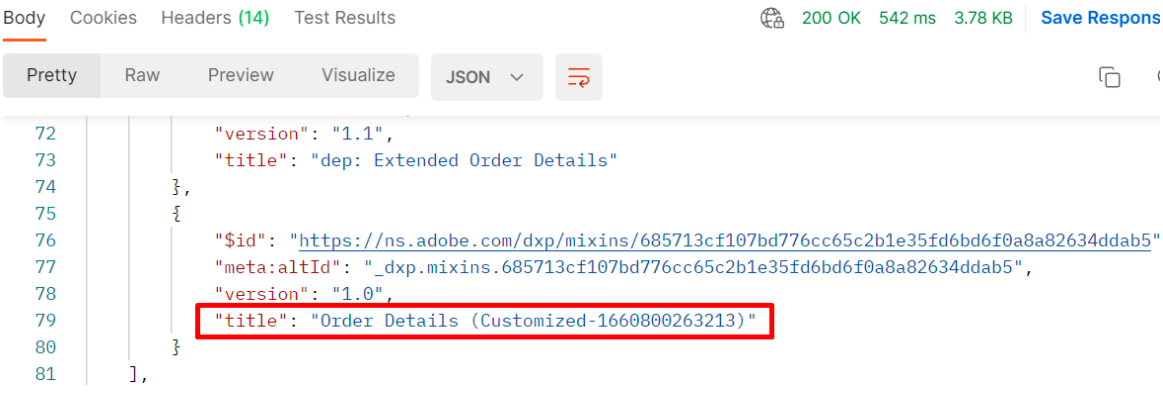


2. While using APIs, we can use a JSON PATCH call to update a portion of the schema. More information about JSON patch is here
    a. **http://jsonpatch.com/**
    b. **https://experienceleague.adobe.com/docs/experience-platform/xdm/api/schemas.html#patch**
3. **Please note the following points:**
    a. Remember,
        i. In AEP, a schema is composed of a class and a set of 1 or more field groups.
        ii. We can not add properties directly to a schema without adding it first to a field group. This helps in re-usability of a field group across schemas.
    b. Adding a new property in a schema would need the following steps
        i. Identify the field group where you would like to add the field
        ii. Open the **"Orders"** schema in UI and get the display name of the field group which you need to update

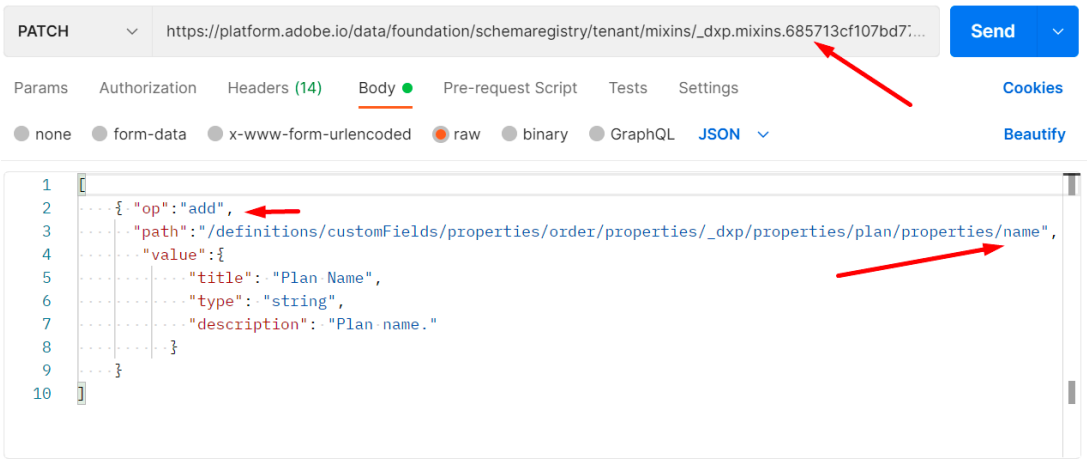


iii. **Execute the "Step 17 - Get Custom Order Field group"** by clicking the **"Send"** button

iv. Search for the schema ID for the custom field group using the display name from UI. Copy the **"meta:altId"** . This will be used in the next step to execute the PATCH call against this field group.



4. **Click on "Step 18 - Modify Custom Order details Field group"** API call in the **"DEP XDM API Lab"** Folder. **Do not execute it yet.**
5. A Sample **"PATCH"** call looks like below.
    a. We are using the **"add"** operation here as we are adding a new property
    b. Make sure that the fully qualified path is correctly specified.
    c. the API request should be having the exact schema ID of the field group we are updating.



6. A Sample API **"RESPONSE"** looks like below:
    d. First screen shot shows that the "$id" of the schema does not change as we are updating an existing schema
    e. Second screen shot below validates that the new property "name" is added to the schema.