



Adobe Experience Platform Bootcamp Deep Dive Edition

DOCUMENTATION AND LABS



CONTENTS

1. Lab Overview	3
2. Learning Objectives	3
3. Pre Requisites	3
4. Lab Guide	3
4.1. Import the following collection into your POSTMAN. This collection has the following folders: DEP API Lab collection	4
4.2. Add the "SANDBOX_NAME" variable into your environment file.	6
4.3. Get the Schema ID for the XDM standard "Order Details" Field Group	7
4.4. Get the Schema ID for the XDM standard "Experience Event" Class.	9
4.5. Create the "Orders" Schema.	10
4.6. Validate and verify your Schema through the UI	12
4.7. Add Custom Properties to Orders Schema.	12
4.8. Browse the "Orders" Schema you created above	13
4.9. Validate and verify your Schema through the UI	14
4.10. Create "Primary Identity" Descriptor for "Orders" Schema	15
4.11. Create Remaining Identity Descriptor(s) for "Orders" Schema	16
4.12. Browse the "Orders" Schema to see Identities.	17
4.13. Validate and verify your Schema through the UI	19
4.14. Get the Schema ID for the Lookup Schemas	20
4.15. Create "Orders to Store" Relationship Descriptor for "Orders" Schema	22
4.16. Create "Orders to Product" Relationship Descriptor for "Orders" Schema	24
4.17. Create "Orders to Plan" Relationship Descriptor for "Orders" Schema	25
4.18. Browse the "Orders" Schema to see your Relationship Descriptors.	25
4.19. Validate and verify your schema through the UI	27
4.20. Create "Store" Reference Identity Descriptor for "Orders" Schema	28
4.21. Create "Product" Reference Identity Descriptor for "Orders" Schema	30
4.22. Create "Plan" Reference Identity Descriptor for "Orders" Schema	30
4.23. Browse the "Orders" Schema to see your Reference Descriptors.	30
4.24. Extend Event Types for Order Schema	30
4.25. Verify and validate your schema through UI	32
4.26. JSON Patch - Modify Order schema to add a missing property	32



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

- 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

- Contains some reference “**GET**” API calls to help you navigate the Schema registry
- 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
- Please note that you will have to **create/POST** the XDM components first to expect results in the GET API calls.
- Below is a sample screen shot of the folders in your **POSTMAN** collection.

Reference GET API Calls

DEP Step 1 - Get XDM Standard Field Groups

DEP Step 2 - Get Experience Event Class

DEP Step 3 - Create Orders Schema

DEP Step 4 - Create Primary Identity for Orders Schema

DEP Step 5 - Create Secondary Identity for Orders Sch...

DEP Step 6 - Get Order Schema and its descriptors

DEP Step 7 - Get Lookup Schemas

DEP Step 8 - Relationship Descriptor Orders To Store

DEP Step 9 - Relationship Descriptor Orders To Product

DEP Step 10 - Relationship Descriptor Orders To Plan

DEP Step 11 - Get Order Schema and its descriptors

DEP Step 12 - Reference Descriptor for Store

DEP Step 13 - Reference Descriptor for Product

DEP Step 14 - Reference Descriptor for Plan

DEP Step 15 - Get Order Schema and its descriptors

DEP Step 16 - Extend the Event Types

API Calls to be execute as part of the DEP API Lab Guide

Click Send to get a response

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/mixins	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

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** Collections, APIs, Environments, Lock Servers, Monitors, Flows, History.
- Current Collection:** DEP API Lab collection.
- Environment Configuration:** DEP Env Config
- Documentation:** Set of Postman calls for the Data Platform Workshop.
- Links:** NEXT IN THIS COLLECTION, Reference API Calls, DEP XDM API Lab.
- Note:** Click here to view your environment file configuration.
- Bottom Buttons:** View complete collection documentation →, Cookies, Capture requests, Bootcamp, Runner, Trash.

A red arrow points to the "Edit" button in the top right corner of the environment configuration panel.

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

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** Explore, Collections, APIs, Environments, Lock Servers, Monitors, Flows, History.
- Current Collection:** DEP API Lab collection.
- Environment Configuration:** DEP Env Config
- Table:** Shows environment variables and their values.
- Buttons:** Edit (highlighted with a red box and arrow).
- Bottom Buttons:** Response, Click Send to get a response.

A red arrow points to the "Edit" button in the top right corner of the environment configuration panel.

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*

The screenshot shows the Postman interface with the "DEP Env Config" tab selected. A red box highlights the "Save" button in the top right corner of the main content area. Another red box highlights the row for the "SANDBOX_NAME" variable, which has its "CURRENT VALUE" set to "008". A red arrow points upwards from the bottom of the "SANDBOX_NAME" row towards the "Save" button.

VARIABLE	TYPE	INITIAL VALUE	CURRENT VALUE
CLIENT_SECRET	default	p8e-q54C4mUmqNmcv54...	p8e-q54C4mUmqNmcv54GgTP3ba...
API_KEY	default	14118835483d4ba1be2077...	14118835483d4ba1be2077bceb9d7...
META_SCOPE	default	ent_dataservices_sdk	ent_dataservices_sdk
ACCESS_TOKEN	default		eyJhbGciOiJSUzI1NiLSIn1dSI6Imtc1...
PRIVATE_KEY	default	-----BEGIN PRIVATE KEY----- ...	-----BEGIN PRIVATE KEY----- ...
JWT_TOKEN	default		eyJhbGciOiJSUzI1NiJ9eyJleHAiOjE2...
TECHNICAL_ACCOUNT_ID	default	035801B362D740240A49...	035801B362D740240A495C0D@te...
IMS	default	ims-na1.adobelogin.com	ims-na1.adobelogin.com
IMS_ORG	default	37E0399C61687C4E0A49...	37E0399C61687C4E0A495E06@Ad...
ADOBE_IO_ACCESS_TOKE...	default		/* ... */
SANDBOX_NAME	default	008	008

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 interchangeably

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>

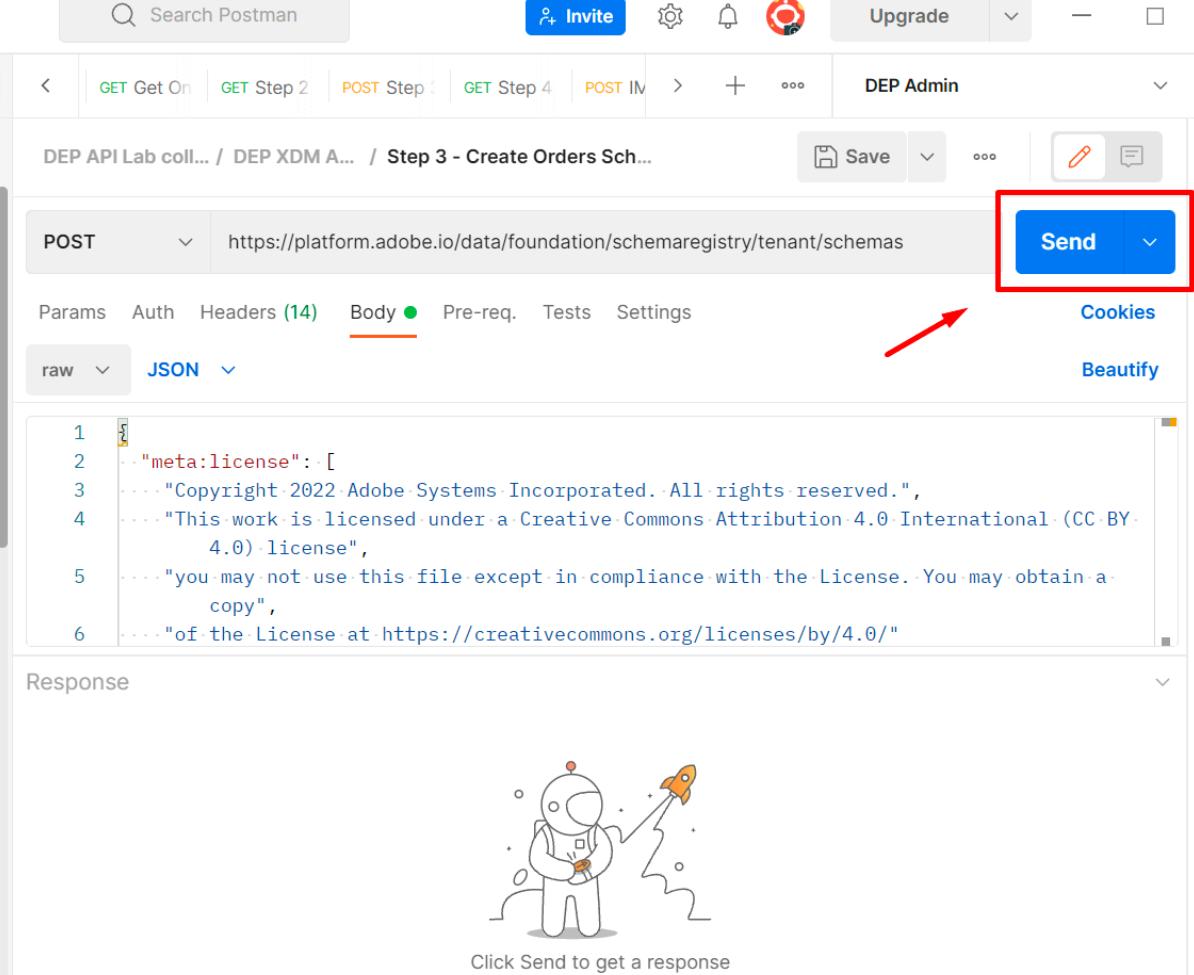
b. This API call gives you all the standard XDM field Groups

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.



The screenshot shows the Postman interface with a POST request to `https://platform.adobe.io/data/foundation/schemaregistry/tenant/schemas`. The 'Body' tab is selected, showing a JSON payload:

```
1 {"meta:license": [2 .... "Copyright 2022 Adobe Systems Incorporated. All rights reserved.",3 .... "This work is licensed under a Creative Commons Attribution 4.0 International (CC BY-4.0) license",4 .... "you may not use this file except in compliance with the License. You may obtain a copy",5 .... "of the License at https://creativecommons.org/licenses/by/4.0/"6 ]}
```

The 'Send' button is highlighted with a red box and an arrow points to it from the left. Below the request, there is a response placeholder with a cartoon character and the text "Click Send to get a response".

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

DEP API Lab collection / DEP XDM API Lab / Step 1 - Get Order Details Field Group

GET https://platform.adobe.io/data/foundation/schemaregistry/global/mixins

Headers (15)

KEY	VALUE	DESCRIPTION
Accept	application/json	Use /global/ to get all the out of the box XDM Standard components. For the components you create, use /tenant/
Content-Type	application/json	
Authorization	Bearer {{ACCESS_TOKEN}}	
x-api-key	{{API_KEY}}	
x-gw-ims-org-id	{{IMS_ORG}}	Search for "Order Details"
x-sandbox-name	{{SANDBOX_NAME}}	
Accept	application/vnd.adobe.xed-id+json	

Body

```

923     "meta:altId": "_xdm.context.experienceevent",
924     "version": "1.36.1",
925     "title": "Stitching Fields"
926   },
927   {
928     "$id": "https://ns.adobe.com/xdm/context/experienceevent-order-details",
929     "meta:altId": "_xdm.context.experienceevent-order-details",
930     "version": "1.36.1",
931     "title": "Order Details"
932   },
933   {
934     "$id": "https://ns.adobe.com/xdm/context/experienceevent-advertising",
935     "meta:altId": "_xdm.context.experienceevent-advertising",
936     "version": "1.36.1",
  
```

Status: 200 OK Time: 285 ms Size: 42.86 KB Save Response

Pretty Raw Preview Visualize JSON

Order Details

Copy this "\$id" onto a notepad

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.

The screenshot shows the Postman interface with a collection named "DEP API Lab collection / DEP XDM API Lab / Step 2 - Get Experience Event Class". A red box highlights the URL in the request header. Another red box highlights the "\$id" field in the JSON response body, which contains the URL <https://ns.adobe.com/xdm/context/experienceevent>. Red arrows point from the highlighted URL in the response to the highlighted URL in the request header.

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.

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

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

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

```

1
2   "allOf": [
3     {
4       "$ref": "https://ns.adobe.com/xdm/context/experienceevent"
5     },
6     {
7       "$ref": "https://ns.adobe.com/xdm/context/experienceevent-order-details"
8     }
9   ],
10  "title": "Orders",
11  "description": "Orders"
12
13

```

Schema ID for the "XDM Experience Event Class"

Schema ID for the "Order Details" Field Group

Each Schema must have a Title and a Description

- Execute **"Step 3 - Create Orders Schema"** by clicking the **"Send"** button
- 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.

- Note how the Schema URL are structured for Adobe standard XDM components as compared to the Customer created ones. See the use of **"dpx"** in the API response below which is the tenant namespace assigned to an IMS Org.

Body Cookies Headers (16) Test Results

Pretty Raw Preview Visualize JSON

```

1
2   "$id": "https://ns.adobe.com/dxp/schemas/110c651e4430d68b9a101000378e3d9b0488bb50cbdebba",
3   "meta:altId": "...dpx.schemas.110c651e4430d68b9a101000378e3d9b0488bb50cbdebba",
4   "meta:xdmType": "schemas",
5   "version": "1.0",
6   "title": "Orders",
7   "type": "object",
8   "description": "Orders",
9   "allOf": [
10    {
11      "$ref": "https://ns.adobe.com/xdm/context/experienceevent",
12      "type": "object",
13      "meta:xdmType": "object"
14    },
15    {
16      "$ref": "https://ns.adobe.com/xdm/context/experienceevent-order-details",
17      "type": "object",
18      "meta:xdmType": "object"
19    },
20    {
21      "$ref": "https://ns.adobe.com/dxp/mixins/b923d7af029efecd7f6f578478caec461eb9923cf6171f19",
22      "type": "object",
23      "meta:xdmType": "object"
24    }
25  ],
26  "refs": [
27    "https://ns.adobe.com/dxp/mixins/b923d7af029efecd7f6f578478caec461eb9923cf6171f19".

```

Each Schema is uniquely identified by these two IDs here:
"\$id"
"meta:altId"

4.6. Validate and verify your Schema through the UI

The screenshot shows the 'Order' schema in the Schema UI editor. The left sidebar contains sections for Composition (Schema: Order), Class (XDM ExperienceEvent), Field groups (Order Details), Identities (None), Relationships (None), and Required fields (None). The central 'Structure' pane displays a tree view of the schema fields, including 'billing' (Object), 'order' (Object), 'productListItems' (Product list items[]), 'shipping' (Shipping), 'store' (Object), '_id' (String), 'eventMergeld' (String), 'eventType' (String), 'identityMap' (Type), 'producedBy' (String), and 'timestamp' (DateTime). The right pane shows 'Schema properties' with 'Display name' set to 'Order' and 'Description' set to 'Orders'. It also includes a 'Profile' section and timestamp details: 'Created' at 08/20/2022, 8:51 AM and 'Last modified' at 08/20/2022, 8:51 AM.

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](#)

```
_dpx {  
    acqSource  
    customerID  
    personalEmail  
},  
order {  
    _dpx {  
        plan {  
            planID  
            name  
        }  
    }  
},  
productListItems[  
    _dpx {  
        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.

Search Postman

DEP API Lab collection / DEP XDM API Lab / Step 4 - Get Orders Schema

GET https://platform.adobe.io/data/foundation/schemaregistry/tenant/schemas/dxp.schemas.4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67

Headers (15)

KEY	VALUE
Accept	application/json
Content-Type	application/json
Authorization	Bearer {{ACCESS_TOKEN}}
x-api-key	{{API_KEY}}
x-gw-ims-org-id	{{IMS_ORG}}
x-sandbox-name	{{SANDBOX_NAME}}
Accept	application/vnd.adobe.xed-full+json;version=1

Accept Header

4. A Sample API “RESPONSE” looks like below:

Pretty Raw Preview Visualize JSON

```

1  {
2      "$id": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
3      "meta:altId": "_dpx.schemas.4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
4      "meta:resourceType": "schemas",
5      "version": "1.0",
6      "title": "Orders 2",
7      "type": "object",
8      "description": "Orders 2",
9      "properties": {
10         "_id": {
11             "title": "Identifier",
12             "type": "string",
13             "format": "uri-reference",
14             "description": "A unique identifier for the time-series event.",
15             "meta:xdmType": "string",
16             "meta:xdmField": "@id"
17         },
18         "billing": {
19             "title": "Billing Details",
20             "type": "object",
21             "description": "Billing related information.",
22             "properties": {
23                 "address": {
24                     "title": "Billing Address",
25                     "description": "Billing Address.",
26                     "type": "object",
27                     "meta:xdmType": "object",
28                     "properties": {}
29                 }
30             }
31         }
32     }
33 }
```

4.9. Validate and verify your Schema through the UI

The screenshot shows the 'Orders' schema structure in the Adobe Experience Platform Schema Editor. The left sidebar displays the schema's composition, including its class (XDM ExperienceEvent), field groups (Order Details), identities (None), relationships (order_dxp.planID), and required fields (None). The right panel shows the schema structure as a tree. Key parts highlighted with red boxes include:

- billing**: An object nested under the main 'Orders' schema.
- order**: An object nested under the main 'Orders' schema, containing a 'plan' object and primitive properties like 'name' and 'planID'.
- productListItems**: An array nested under the main 'Orders' schema, containing a single object with 'make' and 'model' properties.

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 (“/_dpx/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.

The screenshot shows a Postman interface with a POST request to <https://platform.adobe.io/data/foundation/schemaregistry/tenant/descriptors>. The Body tab is selected, showing a JSON payload:

```
1 {  
2   "@type": "xdm:descriptorIdentity",  
3   "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/2055da8b29eca1774fc78cd04f905c41345e2b435440a7c1",  
4   "xdm:sourceVersion": 1,  
5   "xdm:sourceProperty": "/_dpx/profileIdentities/personalEmail",  
6   "xdm:namespace": "Email",  
7   "xdm:property": "xdm:code",  
8   "xdm:isPrimary": true  
9 }
```

Annotations with red boxes and arrows explain the fields:

- “@type”: “xdm:descriptorIdentity”, labeled “Type of Descriptor”
- “xdm:sourceSchema”: “https://ns.adobe.com/dxp/schemas/2055da8b29eca1774fc78cd04f905c41345e2b435440a7c1”, labeled “This has to be the “\$id” of “Orders” Schema”
- “xdm:namespace”: “Email”, labeled “Namespace, this identity is associated with”
- “xdm:isPrimary”: true, labeled ““true” because this is the Primary Identity”

4. A Sample API “**RESPONSE**” looks like below:

```

1  "@id": "2645e525b02bd0f1ecf7f98fb5ac2ffbf77f67919ebb68a",
2  "@type": "xdm:descriptorIdentity",
3  "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
4  "xdm:sourceVersion": 1,
5  "xdm:sourceProperty": "/_dpx/personalEmail",
6  "imsOrg": "37E0399C61687C4E0A495E06@Adobe0rg",
7  "version": "1",
8  "xdm:namespace": "Email",
9  "xdm:property": "xdm:code",
10 "xdm:isPrimary": true,
11 "meta:containerId": "656e7272-a148-4504-ae72-72a148350405",
12 "meta:sandboxId": "656e7272-a148-4504-ae72-72a148350405",
13 "meta:sandboxType": "development"
14
15

```

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 (**"/_dpx/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.

```

1 {
2   "@type": "xdm:descriptorIdentity",
3   "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/2055da8b29eca1774fc78cd04f905c41345e2b435440a7c1",
4   "xdm:sourceVersion": 1,
5   "xdm:sourceProperty": "/_dpx/customerID",
6   "xdm:namespace": "CustomerID",
7   "xdm:property": "xdm:code",
8   "xdm:isPrimary": false
9 }
10
11

```

4. A Sample API “**RESPONSE**” looks like below:

```

1 {
2   "@id": "74d5fd8edac5128b076ce65e4cd5ff22578b1fea6b0d838c",
3   "@type": "xdm:descriptorIdentity",
4   "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
5   "xdm:sourceVersion": 1,
6   "xdm:sourceProperty": "/_dpx/customerID",
7   "imsOrg": "37E0399C61687C4E0A495E06@AdobeOrg",
8   "version": "1",
9   "xdm:namespace": "CustomerID",
10  "xdm:property": "xdm:code",
11  "xdm:isPrimary": false,
12  "meta:containerId": "656e7272-a148-4504-ae72-72a148350405",
13  "meta:sandboxId": "656e7272-a148-4504-ae72-72a148350405",
14  "meta:sandboxType": "development"
15

```

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.

DEP API Lab collection / DEP XDM API Lab / Step 5a - Get Order Schema and its descriptors

GET https://platform.adobe.io/data/foundation/schemaregistry/tenant//schemas/dxp.schemas.4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67

Headers (15) Params Authorization Headers (15) Body Pre-request Script Tests Settings

Headers < 8 hidden

KEY	VALUE
Accept	application/json
Content-Type	application/json
Authorization	Bearer {{ACCESS_TOKEN}}
x-api-key	((API_KEY))
x-gw-ims-org-id	((IMS_ORG))
x-sandbox-name	((SANDBOX_NAME))
Accept	application/vnd.adobe.xed-desc+json;version=1

This Accept header returns Schema as well lists corresponding descriptors.

4. A Sample API “RESPONSE” looks like below:

Body Cookies Headers (15) Test Results

Pretty Raw Preview Visualize JSON

```
1  "$id": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
2  "meta:altId": "_dxp.schemas.4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
3  "meta:resourceType": "schemas",
4  "version": "1.1",
5  "title": "Orders 2",
6  "type": "object",
7  "description": "Orders 2",
8  "allOf": [
9    {
10      "$ref": "https://ns.adobe.com/xdm/context/experienceevent",
11      "type": "object",
12      "meta:xdmType": "object"
13    },
14    {
15      "$ref": "https://ns.adobe.com/xdm/context/experienceevent-order-details",
16      "type": "object",
17      "meta:xdmType": "object"
18    },
19    {
20      "$ref": "https://ns.adobe.com/dxp/mixins/c838a96ea551c5a0002f558e7776b8a4a2c6527ed4664513",
21      "type": "object",
22      "meta:xdmType": "object"
23    }
24  ],
25  "required": [
26    "_id",
27    "timestamp"
28  ]
```

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

Pretty Raw Preview Visualize JSON

```

38     ],
39     "meta:descriptors": [
40       {
41         "@id": "2645e525b02bd0f1ecf7f98fb5ac2ffbd77f67919ebb68a",
42         "@type": "xdm:descriptorIdentity",
43         "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
44         "xdm:sourceVersion": 1,
45         "xdm:sourceProperty": "/_dpx/personalEmail",
46         "imsOrg": "37E0399C61687C4E0A495E06@AdobeOrg",
47         "version": "1",
48         "xdm:namespace": "Email",
49         "xdm:property": "xdm:code",
50         "xdm:isPrimary": true,
51         "meta:containerId": "656e7272-a148-4504-ae72-72a148350405",
52         "meta:sandboxId": "656e7272-a148-4504-ae72-72a148350405",
53         "meta:sandboxType": "development",
54         "meta:registryMetadata": {
55           "repo:createdDate": 1658529022868,
56           "repo:lastModifiedDate": 1658529022868,
57           "xdm:createdClientId": "e22e0562b3d24df0a51ffb3119b208f8",
58           "xdm:lastModifiedClientId": "e22e0562b3d24df0a51ffb3119b208f8",
59           "xdm:createdUserId": "09474C296287A3670A495FA8@techacct.adobe.com",
60           "xdm:lastModifiedUserId": "09474C296287A3670A495FA8@techacct.adobe.com",
61           "eTag": "6b86b273ff34fce19d6b804eff5a3f5747ada4eaa2f1d49c01e52ddb7875b4b"
62         }
63       },
64     },
65   },
66   {
67     "@id": "74d5fd8edac5128b076ce65e4cd5ff22578b1fea6b0d838c",
68     "@type": "xdm:descriptorIdentity",
69     "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
70     "xdm:sourceVersion": 1,
71     "xdm:sourceProperty": "/_dpx/customerID",
72     "imsOrg": "37E0399C61687C4E0A495E06@AdobeOrg",
73     "version": "1",
74     "xdm:namespace": "CustomerID",
75     "xdm:property": "xdm:code",
76     "xdm:isPrimary": false,
77     "meta:containerId": "656e7272-a148-4504-ae72-72a148350405",
78     "meta:sandboxId": "656e7272-a148-4504-ae72-72a148350405",
79     "meta:sandboxType": "development",
80     "meta:registryMetadata": {
81       "repo:createdDate": 1658529157181,
82       "repo:lastModifiedDate": 1658529157181.

```

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

The screenshot shows the 'Orders' schema in the 'Schemas > Orders' interface. The 'Structure' pane displays the schema hierarchy with nodes like 'Orders', 'acqSource', 'customerID', and 'personalEmail'. The 'personalEmail' node is highlighted with a red box. The 'Field properties' pane on the right shows the following configuration for the 'personalEmail' field:

- Note:** Add notes
- Path:** _dpx.personalEmail
- Required:**
- Array:**
- Enum:**
- Identity:**
- Primary identity:**
- Identity namespace ***: Email
- Relationship:**

The screenshot shows the 'Orders' schema in the 'Schemas' section. The 'Structure' pane displays the schema hierarchy with 'customerID' highlighted. The 'Field properties' pane on the right shows the configuration for 'customerID', specifically the 'Identity' section which is checked.

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

NAME	DATASETS	IDENTITIES	RELATIONSHIPS	ENABLED FOR PROFILE	CLASS
Orders	-	-	-	Not enabled	XDM ExperienceEvent
Customer Account Creation	1	2	1	Enabled	XDM Individual Profile
dep: Orders	2	2	3	Enabled	XDM ExperienceEvent
dep: Lookup Plan	1	1	-	Enabled	Plan
dep: Lookup Product	1	1	-	Enabled	Product
dep: Lookup Store	1	1	-	Enabled	dep: Lookup Store Class
dep: Web	1	-	-	Enabled	XDM ExperienceEvent
dep: Ecommerce	1	2	-	Enabled	XDM ExperienceEvent
dep: Billing	1	1	1	Enabled	XDM ExperienceEvent
dep: Customer Aggregates	1	1	-	Enabled	XDM Individual Profile
UPS segmentDefinition schema	1	-	-	Enabled	Segment definition
dep: Customer Active Lines	1	1	1	Enabled	XDM Individual Profile
Journey schema with Journey Fields for Journey Orchestration	-	-	-	Not enabled	Journey Orchestration Cl
Initial Experience Events schema for Journeys	-	-	-	Enabled	XDM ExperienceEvent

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.

DEP API Lab collection / DEP XDM API Lab / Step 7 - Get XDM Standard Schemas

GET https://platform.adobe.io/data/foundation/schemaregistry/tenant/schemas

Params ● Authorization Headers (15) Body ● Pre-request Script Tests Settings

Headers (8 hidden)

KEY	VALUE
Accept	application/json
Content-Type	application/json
Authorization	Bearer {{ACCESS_TOKEN}}
x-api-key	{{API_KEY}}
x-gw-ims-org-id	{{IMS_ORG}}
x-sandbox-name	{{SANDBOX_NAME}}
Accept	application/vnd.adobe.xed+json
Key	Value

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

Status: 200 OK Time: 274 ms Size: 44.58 KB Save Response ▾

Pretty Raw Preview Visualize JSON

1132 "union"
1133],
1134 "meta:allFieldAccess": true
1135 },
1136 {
1137 "\$id": "https://ns.adobe.com/dxp/schemas/d350608adecddfb7216bcab32f7702e3d88d7a35948ba0a",
1138 "meta:altId": "d_xp.schemas.d350608adecddfb7216bcab32f7702e3d88d7a35948ba0a",
1139 "meta:resourceType": "schemas",
1140 "meta:id": "1.1",
1141 "title": "dep: Lookup Product",
1142 "type": "object",
1143 "description": "dep: Lookup Product",
1144 "allOf": [
1145 {
1146 "\$ref": "https://ns.adobe.com/xdm/classes/product",
1147 "type": "object",
1148 "meta:xdmType": "object"
1149 },
1150 {
1151 "\$ref": "https://ns.adobe.com/dxp/mixins/129a6f2823f9883dad967500d5913ebd1109a90d19d2cb7a",
1152 "type": "object",
1153 "meta:xdmType": "object"
1154 },
1155],
1156 "imsOrg": "37E0399C61687C4E0A495E06@AdobeOrg",
1157 "meta:extensible": false,
1158 "meta:abstract": false,
1159 "meta:extends": [
1160 "https://ns.adobe.com/xdm/classes/product",
1161]

dep: Lookup Product Aa Abi * 1 of 2 ↑ ↓ ≡ ×

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

```

1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186     "meta:containerId": "656e7272-a148-4504-ae72-72a1",
1187     "meta:sandboxId": "656e7272-a148-4504-ae72-72a1",
1188     "meta:sandboxType": "development",
1189     "meta:tenantNamespace": "_dxp",
1190     "meta:immutableTags": [
1191         "union"
1192     ],
1193     "meta:allFieldAccess": true
1194 },
1195 {
1196     "$id": "https://ns.adobe.com/dxp/schemas/17a76a0ad31add7bbf7c6afc651c75b01add14d0cdd332e8",
1197     "meta:altId": "_dxp.schemas.17a76a0ad31add7bbf7c6afc651c75b01add14d0cdd332e8",
1198     "meta:resourceType": "schemas",
1199     "version": "1.1",
1200     "title": "dep: Lookup Plan",
1201     "type": "object",
1202     "description": "dep: Lookup Plan",
1203     "allOf": [
1204         {
1205             "$ref": "https://ns.adobe.com/xdm/classes/plan",
1206             "type": "object",
1207             "meta:xdmType": "object"
1208         },
1209         {
1210             "$ref": "https://ns.adobe.com/dxp/mixins/1a13d0bc5bc9e1c2128663d7524d5aac211865daf02dfcd4",
1211             "type": "object",
1212             "meta:xdmType": "object"
1213         }
1214     ]
1215 }

```

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

The screenshot shows the 'Orders' schema in the Adobe Experience Platform Schema Registry. The 'Structure' panel displays the schema hierarchy. A red box highlights the 'store' object node under the 'order' node. An arrow points from this highlighted node to the 'Field properties' panel on the right, which shows the path 'store.storeID'. The 'Field properties' panel also includes fields for Note, Path, Required, Identity, and Relationship.

- 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.

The screenshot shows a Postman API request for creating a relationship descriptor. The URL is <https://platform.adobe.io/data/foundation/schemaregistry/tenant/descriptors>. The Body tab shows a JSON payload:

```

1 {
2   "@type": "xdm:descriptorOneToOne",
3   "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
4   "xdm:sourceVersion": 1,
5   "xdm:sourceProperty": "/store/storeID",
6   "xdm:destinationSchema": "https://ns.adobe.com/dxp/schemas/120be72e3098652e024540cf4b3dd140f8d0c3956037517b",
7   "xdm:destinationVersion": 1
8 }
9
10

```

4. A Sample API “RESPONSE” looks like below:

```

1  {
2      "@id": "96733f2ad239b6aa31f883e98a63d68976e785991c09dcee",
3      "@type": "xdm:descriptorOneToOne",
4      "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
5      "xdm:sourceVersion": 1,
6      "xdm:sourceProperty": "/store/storeID",
7      "xdm:destinationSchema": "https://ns.adobe.com/dxp/schemas/120be72e3098652e024540cf4b3dd140f8d0c3956037517b",
8      "xdm:destinationVersion": 1,
9      "imsOrg": "37E0399C61687C4E0A495E06@AdobeOrg",
10     "version": "1",
11     "meta:containerId": "656e7272-a148-4504-ae72-72a148350405",
12     "meta:sandboxId": "656e7272-a148-4504-ae72-72a148350405",
13     "meta:sandboxType": "development"
14 }

```

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 **ArrayName[*]**.
 - 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.

```

1 {
2   "@type": "xdm:descriptorOneToOne",
3   "xdm:sourceSchema": "https://ns.adobe.com/dxp/classes/c5171a6313de98c92e152a692925a3039b18ad3ddce6f5e3",
4   "xdm:sourceVersion": 1,
5   "xdm:sourceProperty": "/productListItems[*]/SKU",
6   "xdm:destinationSchema": "https://ns.adobe.com/dxp/schemas/d350608adecddfbc7216bcab32f7702e3d88d7a35948ba0a",
7   "xdm:destinationVersion": 1
8 }

```

4. A Sample API “RESPONSE” looks like below:

```

1 {
2   "@id": "2f4183d2681fa28807ba22355544e36206031569cafce42f",
3   "@type": "xdm:descriptorOneToOne",
4   "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa",
5   "xdm:sourceVersion": 1,
6   "xdm:sourceProperty": "/productListItems[*]/SKU",
7   "xdm:destinationSchema": "https://ns.adobe.com/dxp/schemas/d350608adecddfbc7216bcab32f7702e3d88d7a35",
8   "xdm:destinationVersion": 1,
9   "imsOrg": "37E0399C61687C4E0A495E06@AdobeOrg",
10  "version": "1",
11  "meta:containerId": "656e7272-a148-4504-ae72-72a148350405",
12  "meta:sandboxId": "656e7272-a148-4504-ae72-72a148350405",

```

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.

https://platform.adobe.io/data/foundation/schemaregistry/tenant//schemas/_dpx.schemas.4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67

KEY	VALUE
Accept	application/json
Content-Type	application/json
Authorization	Bearer {{ACCESS_TOKEN}}
x-api-key	{{API_KEY}}
x-gw-ims-org-id	{{IMS_ORG}}
x-sandbox-name	{{SANDBOX_NAME}}
Accept	application/vnd.adobe.xed-desc+json;version=1
Key	Value

3. A Sample API “RESPONSE” looks like below:

```

1
2   "$id": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
3   "meta:altId": "_dpx.schemas.4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
4   "meta:resourceType": "schemas",
5   "version": "1.1",
6   "title": "Orders 2", ----->
7   "type": "object",
8   "description": "Orders 2",
9   "allOf": [
10     {
11       "$ref": "https://ns.adobe.com/xdm/context/experienceevent",
12       "type": "object",
13       "meta:xdmType": "object"
14     },
15     {
16       "$ref": "https://ns.adobe.com/xdm/context/experienceevent-order-details",
17       "type": "object",
18       "meta:xdmType": "object"
19     },
20     {
21       "$ref": "https://ns.adobe.com/dxp/mixins/c838a96ea551c5a0002f558e7776b8a4a2c6527ed4664513",
22       "type": "object",
23       "meta:xdmType": "object"
24     }
25   ],
26   "required": [
27     "_id",
28     "timestamp"
29   ],
30   "imsOrg": "37E0399C61687C4E0A495E06@AdobeOrg",
31   "meta:extensible": false,
32   "meta:abstract": false,
33   "meta:extends": [
34     ...
35   ]
...

```

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

Body Cookies Headers (15) Test Results

Pretty Raw Preview Visualize JSON ↻

```

55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
  },
  {
    "@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/d350608adecddfb7216bcab32f7702e3d88d7a35948ba0a",
    "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:createClientId": "e22e0562b3d24df0a51ff3119b208f8",
      "xdm:lastModifiedClientId": "e22e0562b3d24df0a51ff3119b208f8",
      "xdm:createUserId": "09474C296287A3670A495FA0@techacct.adobe.com",
      "xdm:lastModifiedUserId": "09474C296287A3670A495FA0@techacct.adobe.com",
      "eTag": "6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b"
    }
  },
  {
    "@id": "74d5fd8edac5128b076ce65e4cd5ff22578b1fea6b0d838c",
    "@type": "xdm:descriptorIdentity",
    "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
    "xdm:sourceVersion": 1,
    "xdm:sourceProperty": "/_dpx/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

The screenshot shows the 'Orders' schema in the 'Schemas' section. The 'Structure' tab is active, displaying the schema hierarchy. A specific field, `store.storeID`, is selected and highlighted with a red box. To the right, the 'Field properties' panel is open, showing configuration options for this field. Another red box highlights the 'Relationship' checkbox and the 'Reference schema' dropdown, which is set to 'dep: Lookup Store'. The 'Field properties' panel also includes fields for 'Format', 'Minimum length', 'Maximum length', 'Description', 'Note', 'Path', 'Required', 'Identity', and an 'Apply' button.

Schemas > Orders

Composition

Schema

Orders

Class: XDM ExperienceEvent

Field groups: Order Details (Customize...)

Identities: _dpx.customerID | string

Relationships: order_dxp.plan.planID (highlighted with a red box), store.storeID

Required fields: None

Structure

```

graph TD
    Orders[Orders] --> _dpx[_dpx]
    Orders --> billing[billing]
    Orders --> order[order]
    order --> plan[plan]
    plan --> name[name]
    name --> planID[planID]
    planID --> depLookupPlan[dep: Lookup Plan]
    
```

Field properties

- Maximum length: Maximum length
- Description: Description
- Note: Add notes
- Path: order_dxp.plan.planID
- Required:
- Array:
- Enum:
- Identity: Relationship
 - Reference schema: dep: Lookup Plan
 - Reference identity namespace: planID

Schemas > Orders

Composition

Schema

Orders

Class: XDM ExperienceEvent

Field groups: Order Details (Customize...)

Identities: _dpx.customerID | string

Relationships: productListItems.SKU (highlighted with a red box), order_dxp.plan.planID, store.storeID

Required fields: None

Structure

```

graph TD
    Orders[Orders] --> _dpx[_dpx]
    Orders --> billing[billing]
    Orders --> order[order]
    order --> productListItems[productListItems]
    productListItems --> _dpx[_dpx]
    productListItems --> selectedOptions[selectedOptions]
    productListItems --> SKU[SKU]
    SKU --> depLookupProduct[dep: Lookup Product]
    
```

Field properties

- Format:
- Minimum length:
- Maximum length:
- Description: identifier for a product defined by the vendor.
- Note: Add notes
- Path: productListItems.SKU
- Required:
- Identity: Relationship
 - Reference schema: dep: Lookup Product
 - Reference identity namespace: productID

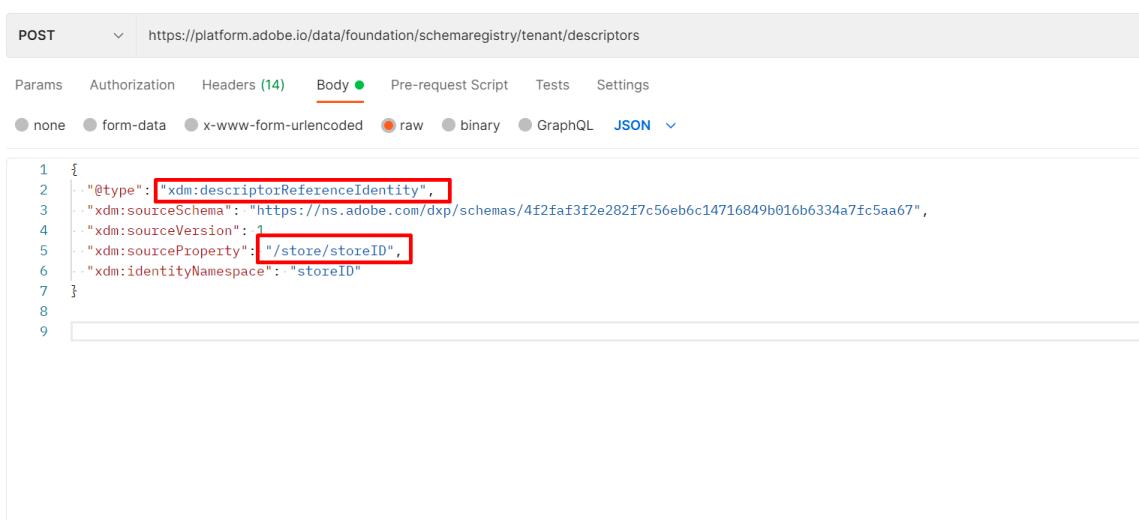
4.20. Create "Store" Reference Identity Descriptor for "Orders" Schema

- Click on **"Step 12 - Reference Descriptor for Store"** API call in the **"DEP XDM API Lab"** Folder. Do not execute it yet.
- Please note the following points:
 - You would have identified the **"relationships"** from **"Orders"** Schema in the **SID lab**.
 - 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.

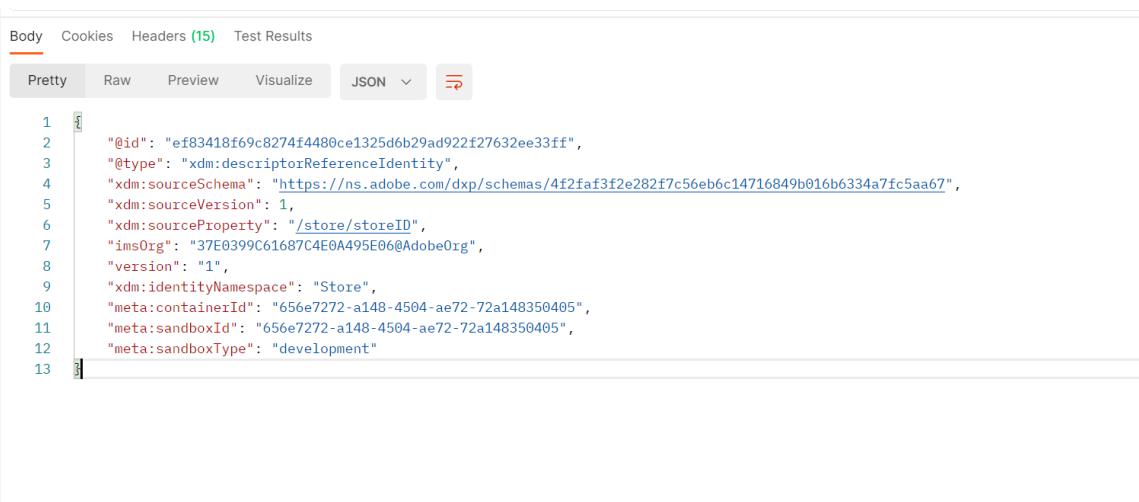


```
POST https://platform.adobe.io/data/foundation/schemaregistry/tenant/descriptors

Params Authorization Headers (14) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 {
2   "@type": "xdm:descriptorReferenceIdentity",
3   "@id": "ef83418f69c8274f4480ce1325d6b29ad922f27632ee33ff",
4   "@version": 1,
5   "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
6   "xdm:sourceVersion": 1,
7   "xdm:sourceProperty": "/store/storeID",
8   "xdm:identityNamespace": "storeID"
9 }
```

4. A Sample API “RESPONSE” looks like below:



```
Body Cookies Headers (15) Test Results
Pretty Raw Preview Visualize JSON ↻

1 {
2   "@id": "ef83418f69c8274f4480ce1325d6b29ad922f27632ee33ff",
3   "@type": "xdm:descriptorReferenceIdentity",
4   "@version": 1,
5   "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/4f2faf3f2e282f7c56eb6c14716849b016b6334a7fc5aa67",
6   "xdm:sourceVersion": 1,
7   "xdm:sourceProperty": "/store/storeID",
8   "imsOrg": "37E0399C61687C4E0A495E06@AdobeOrg",
9   "version": "1",
10  "xdm:identityNamespace": "Store",
11  "meta:containerId": "656e7272-a148-4504-ae72-72a148350405",
12  "meta:sandboxId": "656e7272-a148-4504-ae72-72a148350405",
13  "meta:sandboxType": "development"
```

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:descriptorReferenceldentity**” 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:descriptorReferenceldentity**” 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

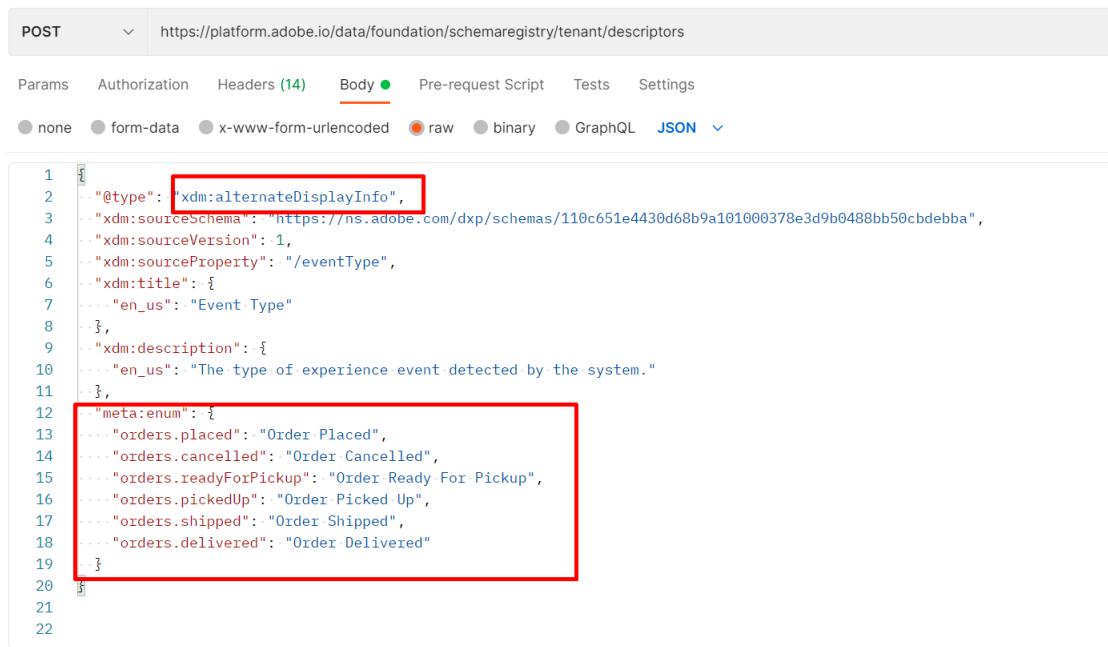
4.24. Extend Event Types for Order Schema

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.



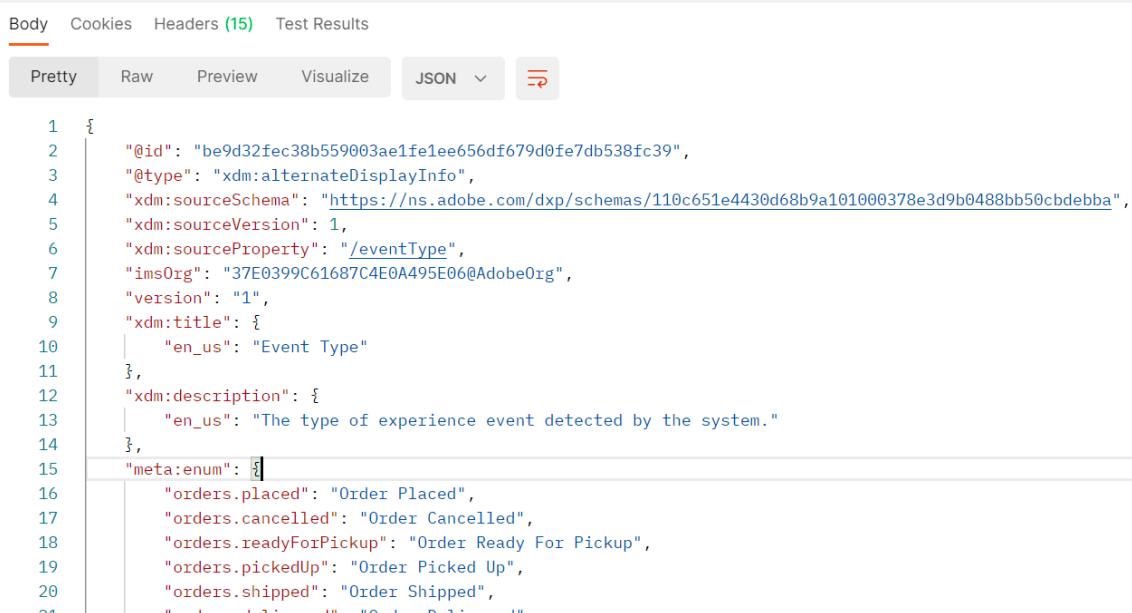
POST https://platform.adobe.io/data/foundation/schemaregistry/tenant/descriptors

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

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

```
1  "@type": "xdm:alternateDisplayInfo",
2  "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/110c651e4430d68b9a101000378e3d9b0488bb50cbdebba",
3  "xdm:sourceVersion": 1,
4  "xdm:sourceProperty": "/eventType",
5  "xdm:title": {
6    "en_us": "Event Type"
7  },
8  "xdm:description": {
9    "en_us": "The type of experience event detected by the system."
10 },
11 "meta:enum": {
12   "orders.placed": "Order Placed",
13   "orders.cancelled": "Order Cancelled",
14   "orders.readyForPickup": "Order Ready For Pickup",
15   "orders.pickedUp": "Order Picked Up",
16   "orders.shipped": "Order Shipped",
17   "orders.delivered": "Order Delivered"
18 }
19
20
21
22
```

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



Body Cookies Headers (15) Test Results

Pretty Raw Preview Visualize **JSON**

```
1  {
2    "@id": "be9d32fec38b559003ae1fe1ee656df679d0fe7db538fc39",
3    "@type": "xdm:alternateDisplayInfo",
4    "xdm:sourceSchema": "https://ns.adobe.com/dxp/schemas/110c651e4430d68b9a101000378e3d9b0488bb50cbdebba",
5    "xdm:sourceVersion": 1,
6    "xdm:sourceProperty": "/eventType",
7    "imsOrg": "37E0399C61687C4E0A495E06@AdobeOrg",
8    "version": "1",
9    "xdm:title": {
10      "en_us": "Event Type"
11    },
12    "xdm:description": {
13      "en_us": "The type of experience event detected by the system."
14    },
15    "meta:enum": [
16      "orders.placed": "Order Placed",
17      "orders.cancelled": "Order Cancelled",
18      "orders.readyForPickup": "Order Ready For Pickup",
19      "orders.pickedUp": "Order Picked Up",
20      "orders.shipped": "Order Shipped",
21      "orders.delivered": "Order Delivered"
22    ]
23  }
```

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

4.25. Verify and validate your schema through UI

Field properties

VALUE	LABEL
advertising.compl...	Advertising Compl...
advertising.timePL...	Advertising Time P...
advertising.federat...	Advertising Feder...
advertising.clicks	Advertising Clicks
advertising.conver...	Advertising Conve...
advertising.firstQu...	Advertising First Q...

Field properties

VALUE	LABEL
media.sessionStart	Media sessionStart
media.stateStart	Media stateStart
media.stateEnd	Media stateEnd
media.statesUpdate	Media statesUpdate
orders.placed	Order Placed
orders.cancelled	Order Cancelled
orders.readyForPic...	Order Ready For Pi...
orders.pickedUp	Order Picked Up
orders.shipped	Order Shipped
orders.delivered	Order Delivered

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

1. Assume we forgot to add a custom field “name” under **order==>_dpx==>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.

Body Cookies Headers (14) Test Results 200 OK 542 ms 3.78 KB Save Response

Pretty Raw Preview Visualize JSON ≡

```

72     "version": "1.1",
73     "title": "dep: Extended Order Details"
74   },
75   {
76     "$id": "https://ns.adobe.com/dxp/mixins/685713cf107bd776cc65c2b1e35fd6bd6f0a8a82634ddab5",
77     "meta:altId": "_dpx.mixins.685713cf107bd776cc65c2b1e35fd6bd6f0a8a82634ddab5",
78     "version": "1.0",
79     "title": "Order Details (Customized-1660800263213)"
80   },
81 ]
,
```

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.

PATCH https://platform.adobe.io/data/foundation/schemaregistry/tenant/mixins/_dpx.mixins.685713cf107bd776cc65c2b1e35fd6bd6f0a8a82634ddab5 Send

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

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

```

1 [
2   {
3     "op": "add", ←
4     "path": "/definitions/customFields/properties/order/properties/_dpx/properties/plan/properties/name", →
5     "value": {
6       "title": "Plan Name",
7       "type": "string",
8       "description": "Plan name."
9     }
10   }
]
```

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.