HCL Digital Experience

HDX-DEV-100
Experience API for Beginners

Edition: July 2024

Herbert Hilhorst – herbert.hilhorst@hcl-software.com

Meghna Pradhan – meghna.pradhan@hcl-software.com

Table of Contents

Author(s)

This document was created by the following Subject Matter Experts:

Herbert Hilhorst
Company: HCLSoftware

Bio

Herbert Hilhorst is an HCL Digital Experience (DX) Technical Advisor at HCLSoftware.

Contact: herbert.hilhorst@hcl-software.com

Meghna Pradhan
Company: HCLSoftware

Bio

Meghna Pradhan is a DX Trainer with HCL with an extensive experience working on all fronts of HCL DX.

Contact: meghna.pradhan@hcl-software.com

Introduction

This hands-on lab guides you through exposing the content on HCL Digital Experience (DX) platform using its advance Experience API capabilities. You will experience how quick and easy it is to use these APIs and reuse content in other applications and perform other actions in DX in a programmatic way.
In this DX developer lab, you play the role of Gene, a developer for the fictitious Woodburn Studio company.

Gene Hayes, Developer, based in Chicago (USA)

As a Web Developer, you will learn how to use the Experience API Explorer swagger documentation to test the API and then use an external application (Postman) to work with it.

Prerequisites

1. Completion of HDX-INTRO and HDX-BU-100 courses, including the labs
2. Access to download the Lab Resources. In the same place where you have found this lab, you will find corresponding resources which you may download and unzip in your Desktop. This helps you to run the lab more easily, and you may later replace it by your own ones.

You will be using the following user IDs and passwords:

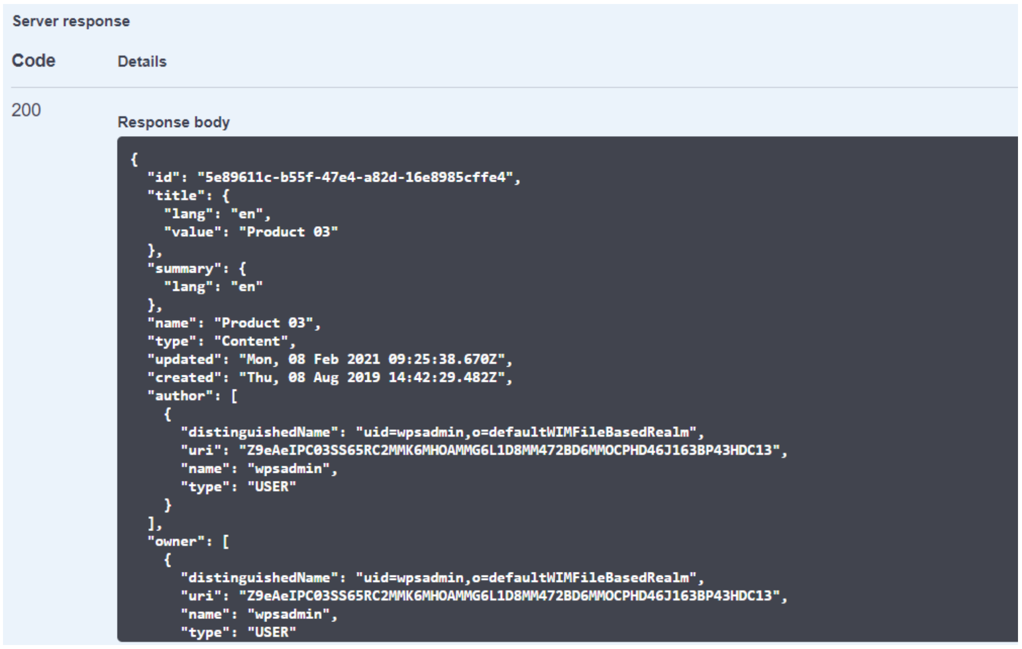
Purpose	**User**	**Password**
SoFy Login	Your official email id	Your password
SoFy Solution Console Login	sol-admin	<from SoFy solution>
Developer Gene Hayes	ghayes (or wpsadmin)	HCL-Dem0 (or wpsadmin)

Lab Overview

In this lab there are several parts to get you started developing with the DX Experience API. These are shortly introduced now.

Part 1: Explore Experience API

You will use the Experience API swagger documentation to learn how to create the right API calls to login into Experience API and retrieve any DX content in JSON format. Here is an example response:



Server response

Code Details

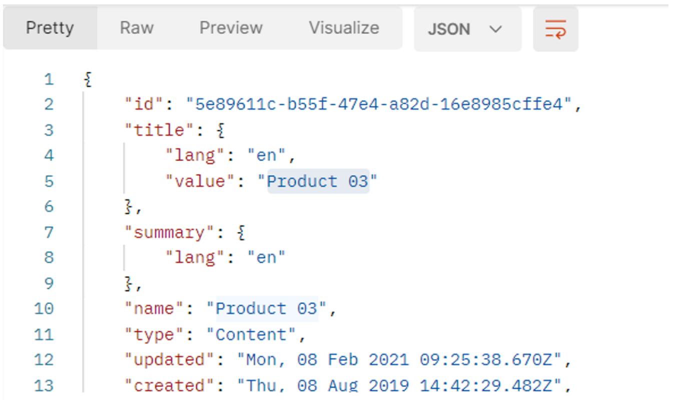
200

Response body

```
{
  "id": "5e89611c-b55f-47e4-a82d-16e8985cffe4",
  "title": {
    "lang": "en",
    "value": "Product 03"
  },
  "summary": {
    "lang": "en"
  },
  "name": "Product 03",
  "type": "Content",
  "updated": "Mon, 08 Feb 2021 09:25:38.670Z",
  "created": "Thu, 08 Aug 2019 14:42:29.482Z",
  "author": [
    {
      "distinguishedName": "uid=wpsadmin,o=defaultWIMFileBasedRealm",
      "uri": "Z9eAeIPC03SS65RC2MMK6MHOAMMG6L1D8MM472BD6MMOCPHD46J163BP43HDC13",
      "name": "wpsadmin",
      "type": "USER"
    }
  ],
  "owner": [
    {
      "distinguishedName": "uid=wpsadmin,o=defaultWIMFileBasedRealm",
      "uri": "Z9eAeIPC03SS65RC2MMK6MHOAMMG6L1D8MM472BD6MMOCPHD46J163BP43HDC13",
      "name": "wpsadmin",
      "type": "USER"
```

Part 2: Use Experience API Externally

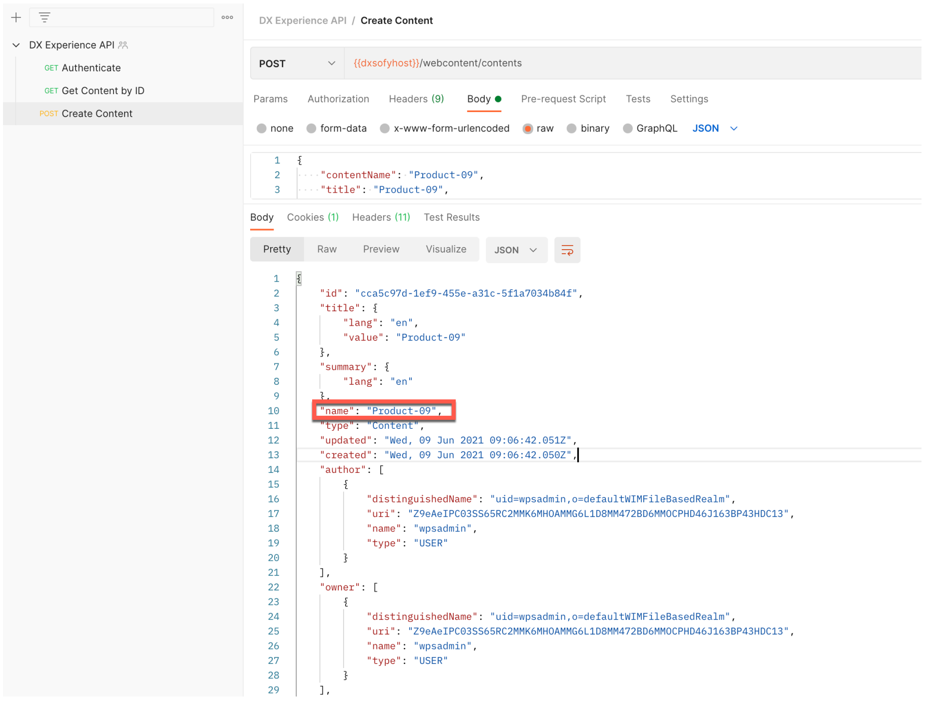
Then you will use an external application, in this case Postman, to login into the DX Experience API and retrieve any DX content as JSON.



```
{
    "id": "5e89611c-b55f-47e4-a82d-16e8985cffe4",
    "title": {
        "lang": "en",
        "value": "Product 03"
    },
    "summary": {
        "lang": "en"
    },
    "name": "Product 03",
    "type": "Content",
    "updated": "Mon, 08 Feb 2021 09:25:38.670Z",
    "created": "Thu, 08 Aug 2019 14:42:29.482Z",
```

Part 3: Create new DX Content Externally using Experience API

And, again using Postman, you will create a new DX content, **Product 09**, using the Experience API.



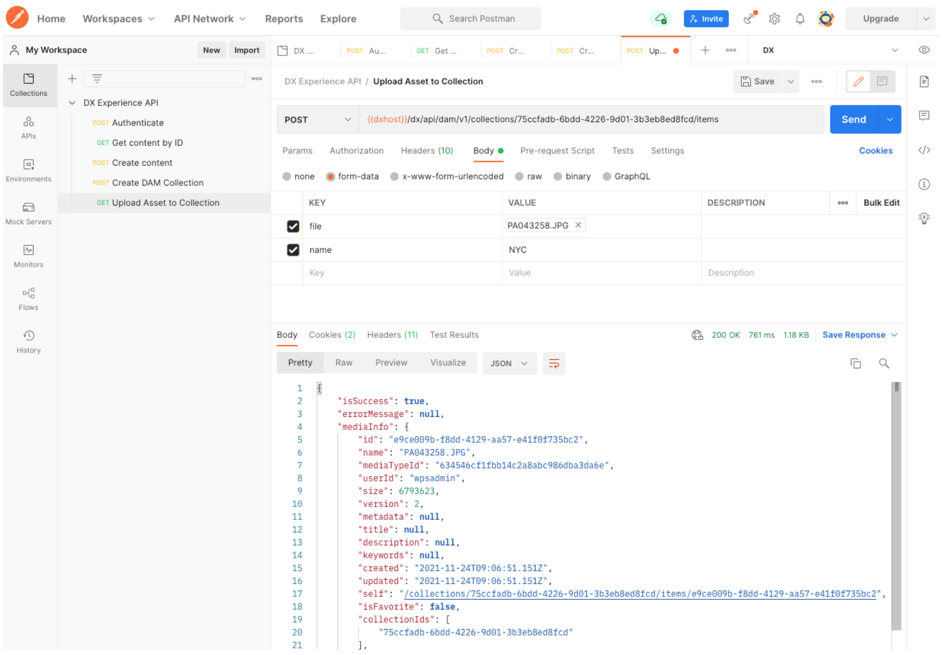
```
POST {{dxsofyhost}}/webcontent/contents
```

```
{
    "contentName": "Product-09",
    "title": "Product-09",
```

```
{
    "id": "cca5c97d-1ef9-455e-a31c-5f1a7034b84f",
    "title": {
        "lang": "en",
        "value": "Product-09"
    },
    "summary": {
        "lang": "en"
    },
    "name": "Product-09",
    "type": "Content",
    "updated": "Wed, 09 Jun 2021 09:06:42.051Z",
    "created": "Wed, 09 Jun 2021 09:06:42.050Z",
    "author": [
        {
            "distinguishedName": "uid=wpsadmin,o=defaultWIMFileBasedRealm",
            "uri": "Z9eAeIPC03SS65RC2MMK6MHOAMMG6L1D8MM472BD6MMOCPHD46J163BP43HDC13",
            "name": "wpsadmin",
            "type": "USER"
        }
    ],
    "owner": [
        {
            "distinguishedName": "uid=wpsadmin,o=defaultWIMFileBasedRealm",
            "uri": "Z9eAeIPC03SS65RC2MMK6MHOAMMG6L1D8MM472BD6MMOCPHD46J163BP43HDC13",
            "name": "wpsadmin",
            "type": "USER"
        }
    ],
```

Part 4: Upload a DAM Image and Add to Web Content

And, again using Postman, you will use the Digital Asset Management API or short, DAM API, to create a new collection and upload an image to the collection. Then you will use the WCM V2 API again to add the image to the content you created in the previous part. And learn how to reuse this collection of APIs call in an programming language.



```
POST {{dxhost}}/dx/api/dam/v1/collections/75ccfadb-6bdd-4226-9d01-3b3eb8ed8fcd/items
```

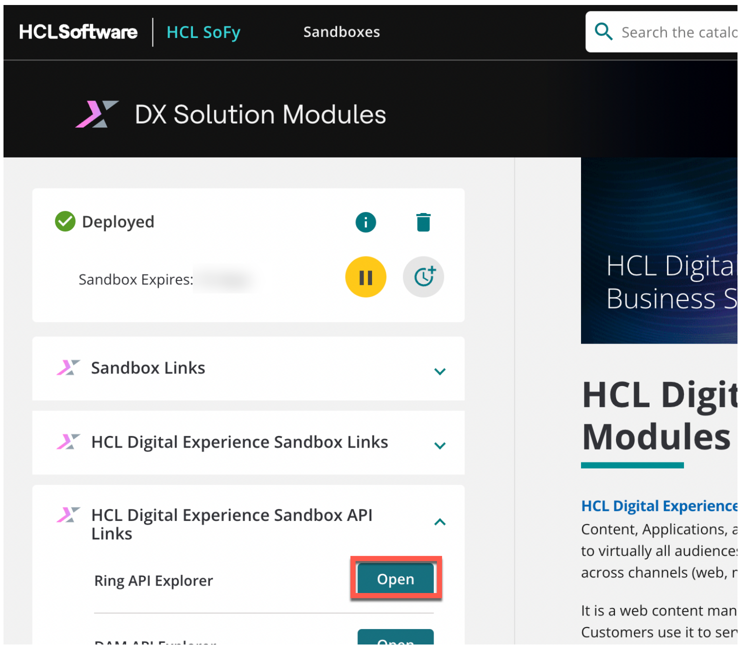
	KEY	VALUE	DESCRIPTION
✓	file	PA043258.JPG	
✓	name	NYC	
	Key	Value	Description

```
{
    "isSuccess": true,
    "errorMessage": null,
    "mediaInfo": {
        "id": "e9ce009b-f8dd-4129-aa57-e41f0f735bc2",
        "name": "PA043258.JPG",
        "mediaTypeId": "634546cf1fbb14c2a8abc986dba3da6e",
        "userId": "wpsadmin",
        "size": 6793623,
        "version": 2,
        "metadata": null,
        "title": null,
        "description": null,
        "keywords": null,
        "created": "2021-11-24T09:06:51.151Z",
        "updated": "2021-11-24T09:06:51.151Z",
        "self": "/collections/75ccfadb-6bdd-4226-9d01-3b3eb8ed8fcd/items/e9ce009b-f8dd-4129-aa57-e41f0f735bc2",
        "isFavorite": false,
        "collectionIds": [
            "75ccfadb-6bdd-4226-9d01-3b3eb8ed8fcd"
        ],
```

Part 1: Explore Experience API

You will first use the Experience API swagger documentation to learn how to create the right API calls to login into Experience API and retrieve any DX content in JSON format.
For that you need to have the URL to access this Experience API Explorer for the Core API, which is your host with the path /dx/api/core/v1/explorer, e.g. for your HCL SoFy sandbox this is something like https://dx.sbx<your sandbox id>.play.hclsofy.com/dx/api/core/v1/explorer. As you are using HCL SoFy, you may use it to get easily access to this and the other APIs.

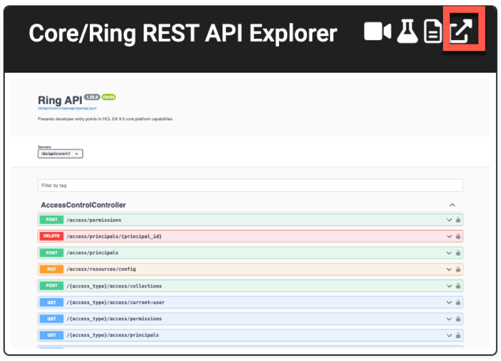
1. Access the HCL Digital Experience Sandbox API Links on your sandbox. You see all the Experience API links of all available APIs. You will start with the Ring API. You may later check the others as well. Click **Open** as shown:



Content, Applications, a
to virtually all audience
across channels (web, r

It is a web content man
Customers use it to ser

2. Alternatively, you may access the explorer using the DX Solution Modules Demo Home Page, and open it in a dedicated page.



Core/Ring REST API Explorer

Ring API 1.33.0 OAS3

/dx/api/core/v1/openapi/openapi.json

Presents developer entry points to HCL DX 9.5 core platform capabilities.

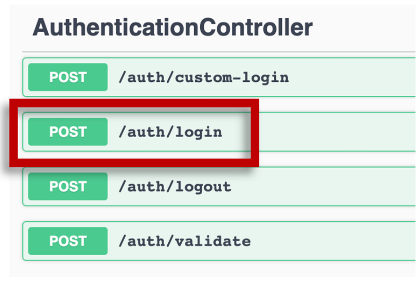
Servers
/dx/api/core/v1

Filter by tag

AccessControlController

```
POST    /access/permissions
DELETE  /access/principals/{principal_id}
POST    /access/principals
PUT     /access/resources/config
POST    /{access_type}/access/collections
GET     /{access_type}/access/current-user
GET     /{access_type}/access/permissions
GET     /{access_type}/access/principals
```

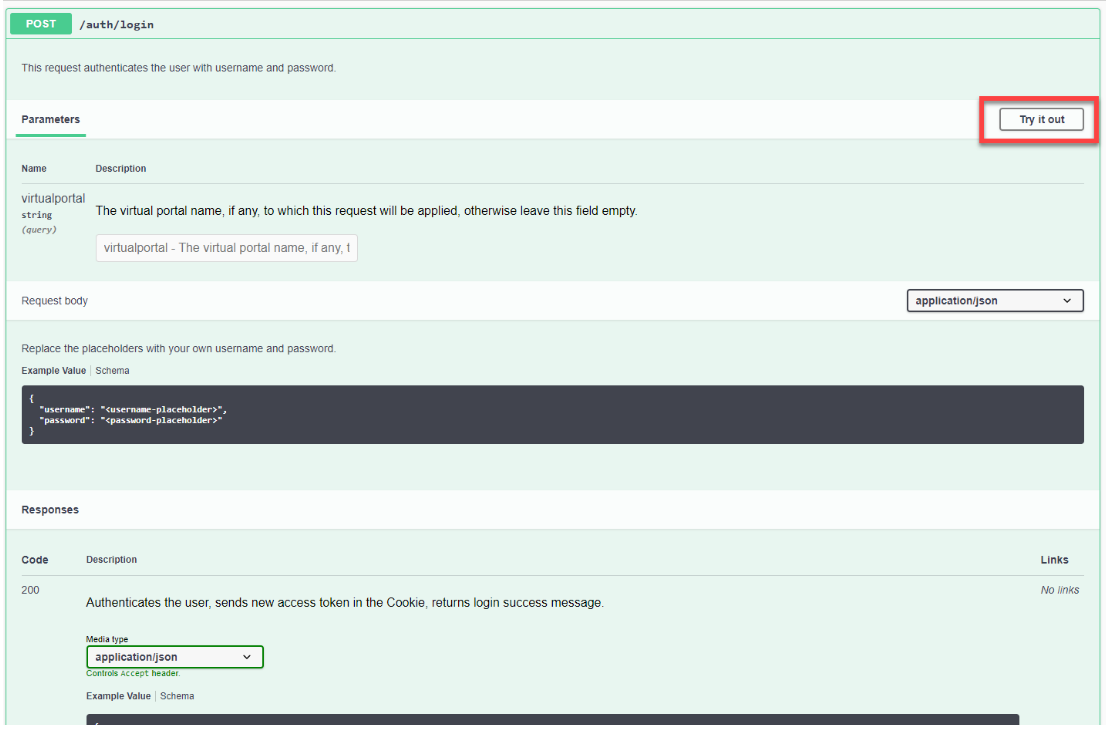
3. This gives you access to the swagger documentation of the Core API, called “ringapi-server”. In this documentation, it is possible to trial any of these methods. To have access to resources on the server, you need to be authenticated. This can be modified, but is set up this way by default. To authenticate, there is a login API under the Authentication Controller. Scroll down to the **Authentication Controller**. Under the **AuthenticationController** click on the **/auth/login**. You can see that this is a POST method call to get access to its details.



AuthenticationController

```
POST    /auth/custom-login
POST    /auth/login
POST    /auth/logout
POST    /auth/validate
```

4. You have access to its details, parameters and responses. Now try it out. Notice you can set the media type in responses. Keep it on application/json and click **Try it out**.



POST /auth/login

This request authenticates the user with username and password.

Parameters

Try it out

Name	Description
virtualportal string *(query)*	The virtual portal name, if any, to which this request will be applied, otherwise leave this field empty. virtualportal - The virtual portal name, if any, t

Request body — application/json

Replace the placeholders with your own username and password.

Example Value | Schema

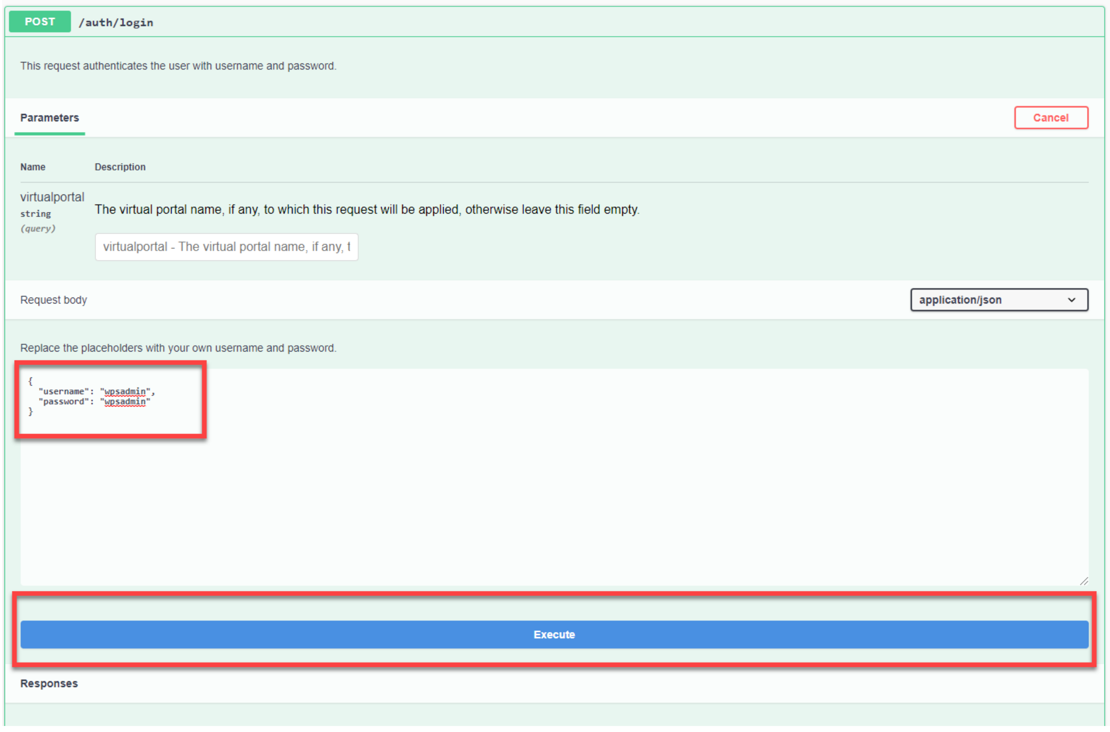
```
{
  "username": "<username-placeholder>",
  "password": "<password-placeholder>"
}
```

Responses

Code	Description	Links
200	Authenticates the user, sends new access token in the Cookie, returns login success message. Media type application/json Controls Accept header. Example Value | Schema	*No links*

5. Under parameters, in the Request Body section, you need to provide the DX server user name and password credentials. Enter a user with the right credentials in JSON format, e.g. for **wpsadmin**, which shows like (and which you may copy & paste).
{
"username": "wpsadmin",
"password": "wpsadmin"
}

6. Then run it. Click **Execute**.



POST /auth/login

This request authenticates the user with username and password.

Parameters

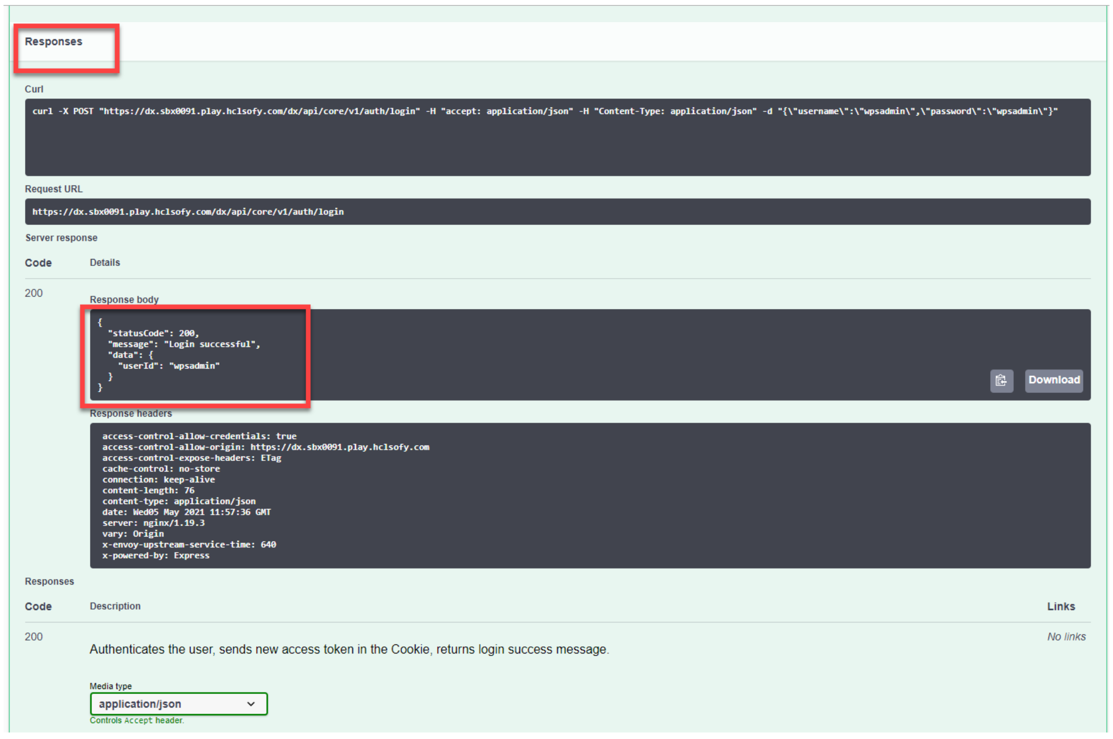
Name	Description
virtualportal string *(query)*	The virtual portal name, if any, to which this request will be applied, otherwise leave this field empty. virtualportal - The virtual portal name, if any, t

Request body: application/json

Replace the placeholders with your own username and password.

```
{
  "username": "wpsadmin",
  "password": "wpsadmin"
}
```

7. Check the Curl and Request URL that was created to execute, as you will use this later again. Then scroll down to verify the that the return Code is 200 (which means successful) and Response body stating the message “Login successful”. You are logged in now.



Responses

Curl

```
curl -X POST "https://dx.sbx0091.play.hclsofy.com/dx/api/core/v1/auth/login" -H "accept: application/json" -H "Content-Type: application/json" -d "{\"username\":\"wpsadmin\",\"password\":\"wpsadmin\"}"
```

Request URL

```
https://dx.sbx0091.play.hclsofy.com/dx/api/core/v1/auth/login
```

Server response

Code: 200

Response body

```
{
  "statusCode": 200,
  "message": "Login successful",
  "data": {
    "userId": "wpsadmin"
  }
}
```

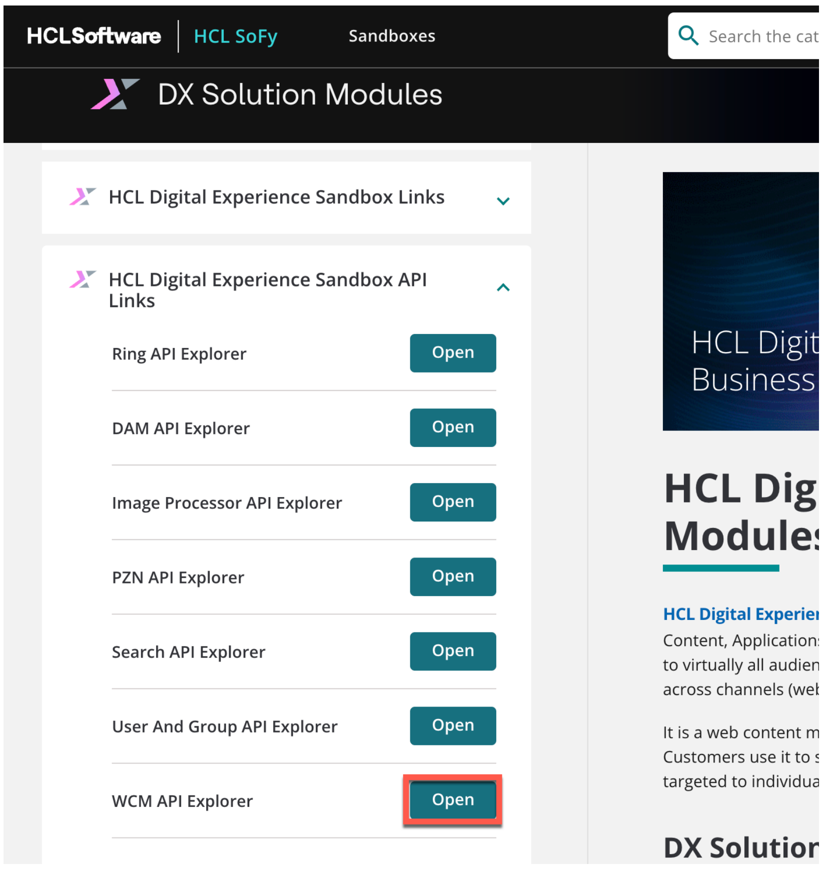
Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: https://dx.sbx0091.play.hclsofy.com
access-control-expose-headers: ETag
cache-control: no-store
connection: keep-alive
content-length: 76
content-type: application/json
date: Wed05 May 2021 11:57:36 GMT
server: nginx/1.19.3
vary: Origin
x-envoy-upstream-service-time: 640
x-powered-by: Express
```

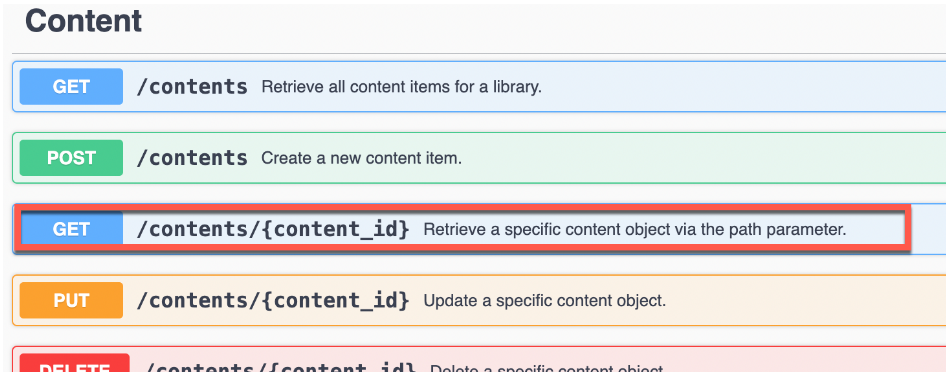
Responses

Code	Description	Links
200	Authenticates the user, sends new access token in the Cookie, returns login success message. Media type: application/json. Controls Accept header.	*No links*

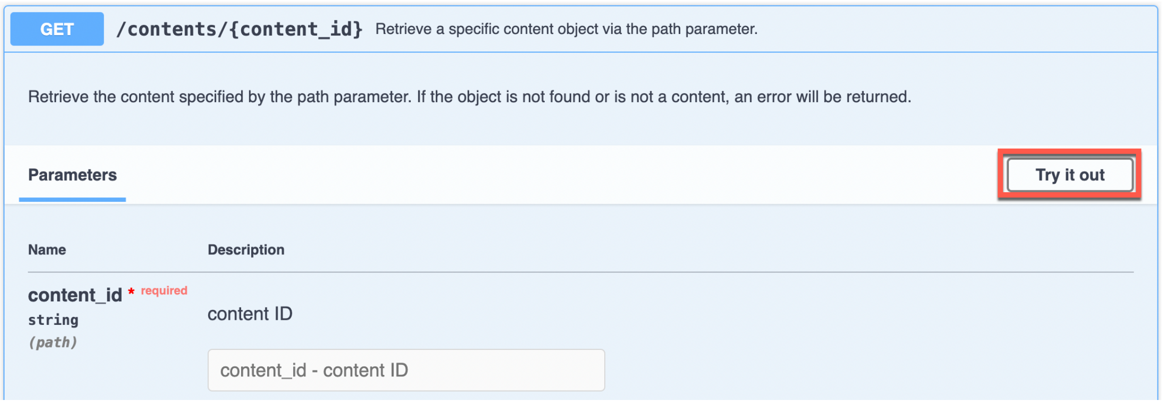
8. Now you need to retrieve the JSON data of a DX content. This is stored in WCM, so you need to use the WCM API. Go back to your SoFy sandbox and next to **WCM API Explorer** click **Open**.



9. Scroll down to the **Content** and click **/contents/{content_id}** to open that method. Notice that this is a GET request, as you will retrieve the JSON of a content.


Content
GET /contents Retrieve all content items for a library.
POST /contents Create a new content item.
GET /contents/{content_id} Retrieve a specific content object via the path parameter.
PUT /contents/{content_id} Update a specific content object.
DELETE /contents/{content_id} Delete a specific content object

10. This method requires a mandatory value for **content_id**. Click **Try it out**.

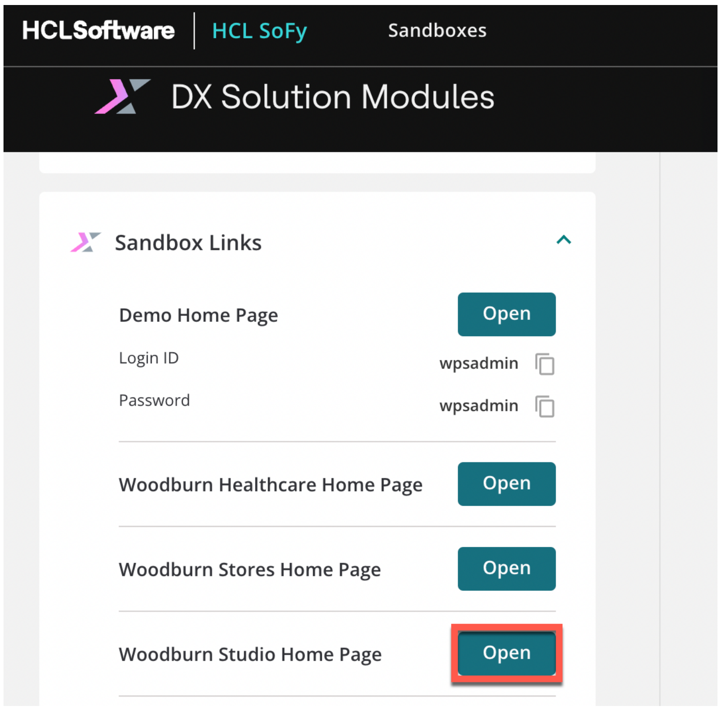


GET /contents/{content_id} Retrieve a specific content object via the path parameter.

Retrieve the content specified by the path parameter. If the object is not found or is not a content, an error will be returned.

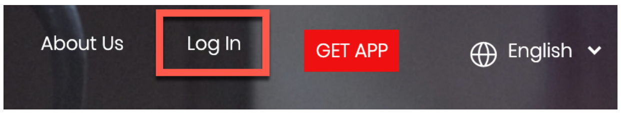
Name	Description
content_id * required string (path)	content ID content_id - content ID

11. To fetch the **content_id**, go to the Content Composer on the DX server. From your sandbox you can easily find your DX server URL (with /wps/portal). Next to **Woodburn Studio Home Page**, click **Open**.

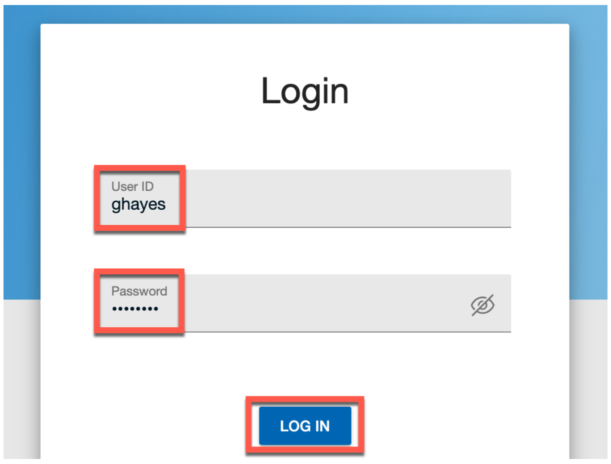


Demo Home Page	Open
Login ID	wpsadmin
Password	wpsadmin
Woodburn Healthcare Home Page	Open
Woodburn Stores Home Page	Open
Woodburn Studio Home Page	Open

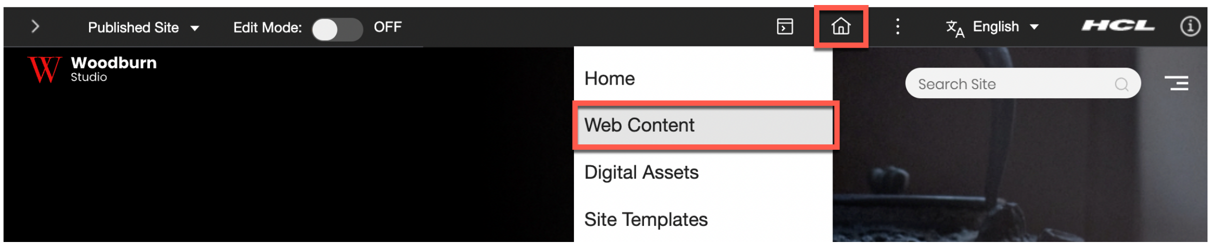
12. Then log in as Gene. Click **Log in**.



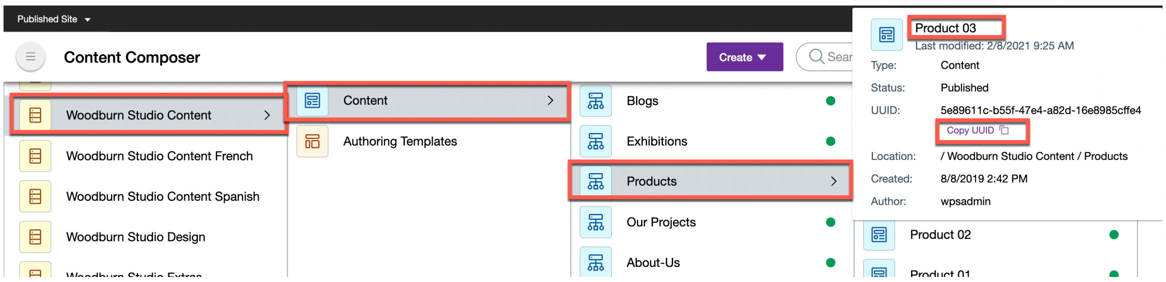
13. Then use the credentials of Gene (User ID: **ghayes**, Password: **HCL-Dem0** – if you do not have these, you may use the default administrator account with User ID: **wpsadmin,** Password: **wpsadmin)**



14. Open the applications menu and click **Web Content**.

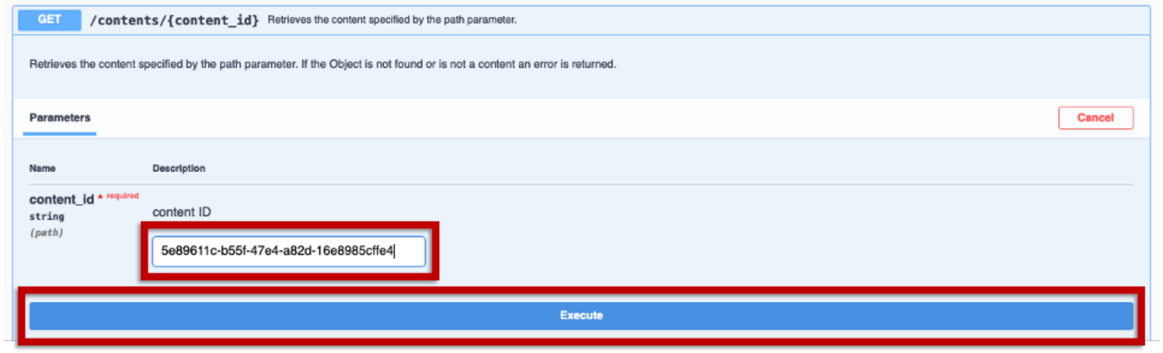


15. Scroll down and navigate to **WoodBurn Studio Content** - **Content** - **Products**, hover over **Product 03** content (or any other content of your choice) and click **Copy UUID** to put the UUID into your clipboard:



Product 03
Last modified: 2/8/2021 9:25 AM
Type: Content
Status: Published
UUID: 5e89611c-b55f-47e4-a82d-16e8985cffe4
Location: / Woodburn Studio Content / Products
Created: 8/8/2019 2:42 PM
Author: wpsadmin

16. Enter the copied UUID of the content in the **content_id** field and click **Execute**.

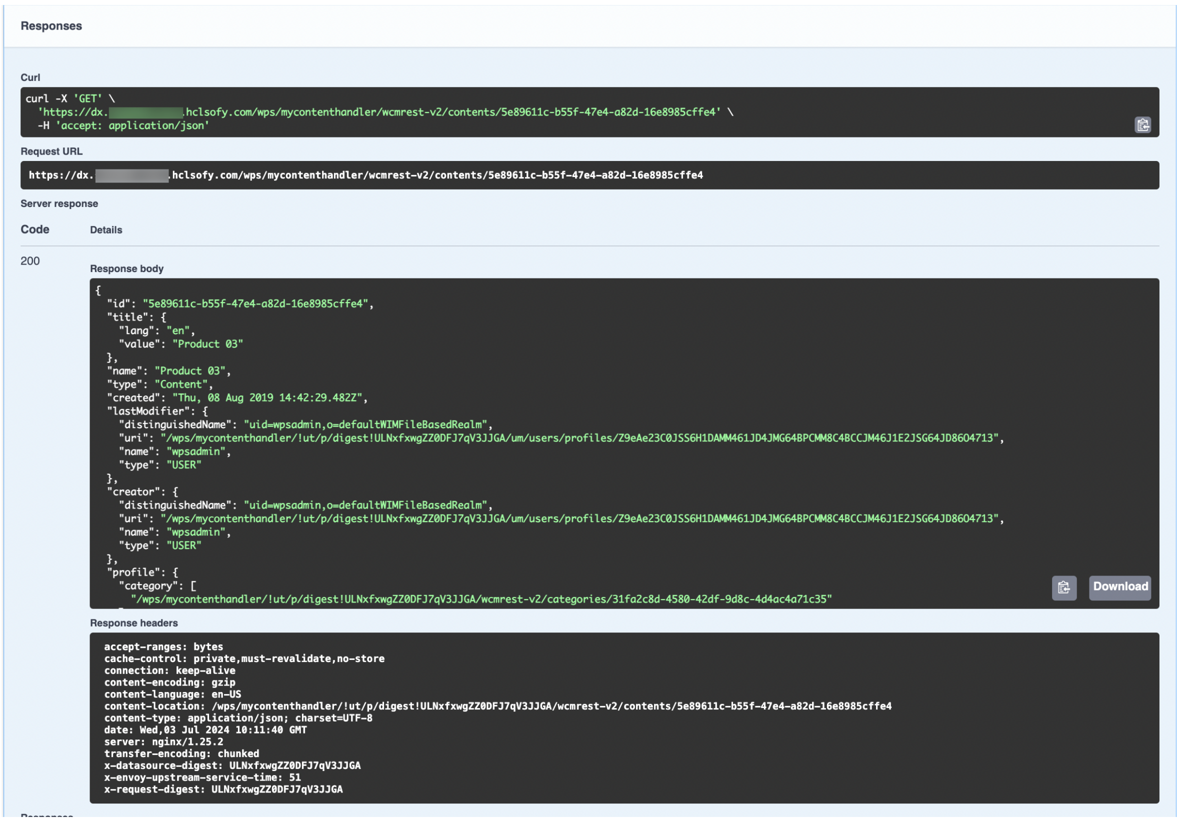


/contents/{content_id} Retrieves the content specified by the path parameter.

Retrieves the content specified by the path parameter. If the Object is not found or is not a content an error is returned.

Name	Description
content_id * required string (path)	content ID 5e89611c-b55f-47e4-a82d-16e8985cffe4

17. Again check the Curl and Request URL that you will use later, and then verify under the **Responses** section, that the response code is 200 (successful again) and the JSON response of the content you retrieved. Check out the Request URL that you will use further on in the lab.



Responses

Curl

```
curl -X 'GET' \
  'https://dx.[redacted].hclsofy.com/wps/mycontenthandler/wcmrest-v2/contents/5e89611c-b55f-47e4-a82d-16e8985cffe4' \
  -H 'accept: application/json'
```

Request URL

```
https://dx.[redacted].hclsofy.com/wps/mycontenthandler/wcmrest-v2/contents/5e89611c-b55f-47e4-a82d-16e8985cffe4
```

Server response

Code Details

200

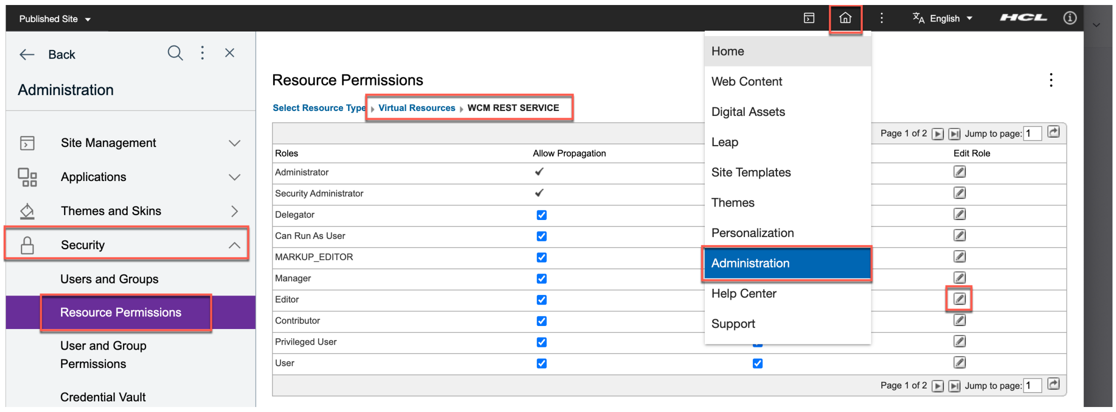
Response body

```
{
  "id": "5e89611c-b55f-47e4-a82d-16e8985cffe4",
  "title": {
    "lang": "en",
    "value": "Product 03"
  },
  "name": "Product 03",
  "type": "Content",
  "created": "Thu, 08 Aug 2019 14:42:29.482Z",
  "lastModifier": {
    "distinguishedName": "uid=wpsadmin,o=defaultWIMFileBasedRealm",
    "uri": "/wps/mycontenthandler/!ut/p/digest!ULNxfxwgZZ0DFJ7qV3JJGA/um/users/profiles/Z9eAe23C0JSS6H1DAMM461JD4JMG64BPCMM8C4BCCJM46J1E2JSG64JD8604713",
    "name": "wpsadmin",
    "type": "USER"
  },
  "creator": {
    "distinguishedName": "uid=wpsadmin,o=defaultWIMFileBasedRealm",
    "uri": "/wps/mycontenthandler/!ut/p/digest!ULNxfxwgZZ0DFJ7qV3JJGA/um/users/profiles/Z9eAe23C0JSS6H1DAMM461JD4JMG64BPCMM8C4BCCJM46J1E2JSG64JD8604713",
    "name": "wpsadmin",
    "type": "USER"
  },
  "profile": {
    "category": [
      "/wps/mycontenthandler/!ut/p/digest!ULNxfxwgZZ0DFJ7qV3JJGA/wcmrest-v2/categories/31fa2c8d-4580-42df-9d8c-4d4ac4a71c35"
```

Response headers

```
accept-ranges: bytes
cache-control: private,must-revalidate,no-store
connection: keep-alive
content-encoding: gzip
content-language: en-US
content-location: /wps/mycontenthandler/!ut/p/digest!ULNxfxwgZZ0DFJ7qV3JJGA/wcmrest-v2/contents/5e89611c-b55f-47e4-a82d-16e8985cffe4
content-type: application/json; charset=UTF-8
date: Wed,03 Jul 2024 10:11:40 GMT
server: nginx/1.25.2
transfer-encoding: chunked
x-datasource-digest: ULNxfxwgZZ0DFJ7qV3JJGA
x-envoy-upstream-service-time: 51
x-request-digest: ULNxfxwgZZ0DFJ7qV3JJGA
```

18. For some of the REST API calls, like search, you need to change the access permissions on the WCM REST SERVICE and grant the anonymous users editor access. This setting is only required to allow anonymous users use the API. It does not override the access you may have set on the actual content itself. In this lab, you will need to configure this. You need to be logged in as an administrator, e.g. Harry (hpappus/HCL-Dem0) or wpsadmin/wpsadmin, but you may also use Gene (ghayes/HCL-Dem0). In the **Practitioner Studio**, under **Administration – Security - Resource Permissions - Virtual Resources** and click **Assign Access** for **WCM REST SERVICES**. Next to the **Editor** role, click the **Edit Role** icon.



Resource Permissions

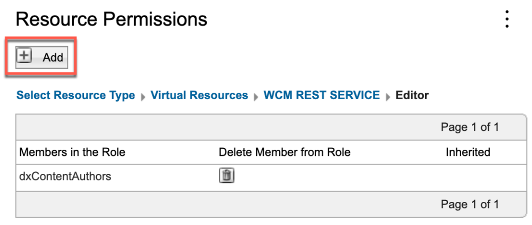
Select Resource Type ▸ Virtual Resources ▸ WCM REST SERVICE

Page 1 of 2 Jump to page: 1

Roles	Allow Propagation		Edit Role
Administrator	✓		✎
Security Administrator	✓		✎
Delegator	☑		✎
Can Run As User	☑		✎
MARKUP_EDITOR	☑		✎
Manager	☑		✎
Editor	☑		✎
Contributor	☑		✎
Privileged User	☑		✎
User	☑	☑	✎

Page 1 of 2 Jump to page: 1

19. Click **Add**.



Resource Permissions

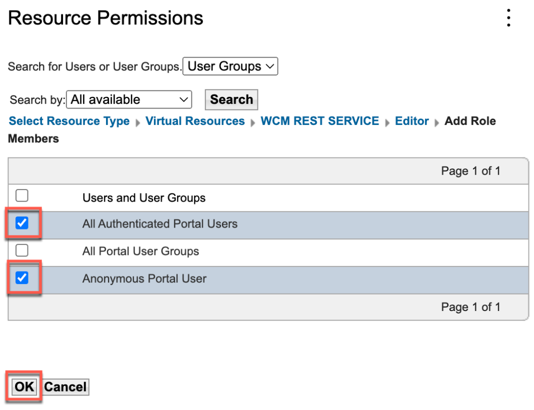
Add

Select Resource Type › Virtual Resources › WCM REST SERVICE › Editor

Members in the Role	Delete Member from Role	Inherited
dxContentAuthors	🗑	

Page 1 of 1

20. Select **All Authenticated Portal Users** and **Anonymous Portal User** and click **OK**.



Resource Permissions

Search for Users or User Groups: User Groups

Search by: All available Search

Select Resource Type › Virtual Resources › WCM REST SERVICE › Editor › Add Role Members

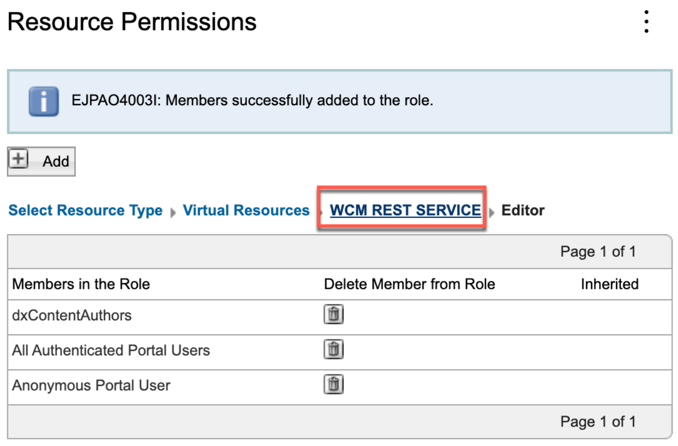
Page 1 of 1

☐	Users and User Groups
☑	All Authenticated Portal Users
☐	All Portal User Groups
☑	Anonymous Portal User

Page 1 of 1

OK Cancel

21. Then go back to the virtual resource. Click **WCM REST SERVICE**.



Resource Permissions

EJPAO4003I: Members successfully added to the role.

Add

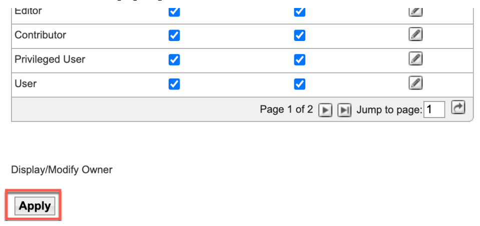
Select Resource Type › Virtual Resources › WCM REST SERVICE › Editor

Page 1 of 1

Members in the Role	Delete Member from Role	Inherited
dxContentAuthors	🗑	
All Authenticated Portal Users	🗑	
Anonymous Portal User	🗑	

Page 1 of 1

22. And then **Apply** and confirm with **OK**.



Editor	☑	☑	✎
Contributor	☑	☑	✎
Privileged User	☑	☑	✎
User	☑	☑	✎

Page 1 of 2 Jump to page: 1

Display/Modify Owner

Apply

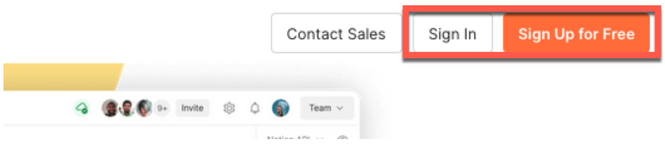
Congratulations! You have successfully logged in and retrieved an existing content in JSON format using the Experience API explorer interface!

Part 2: Use Experience API Externally

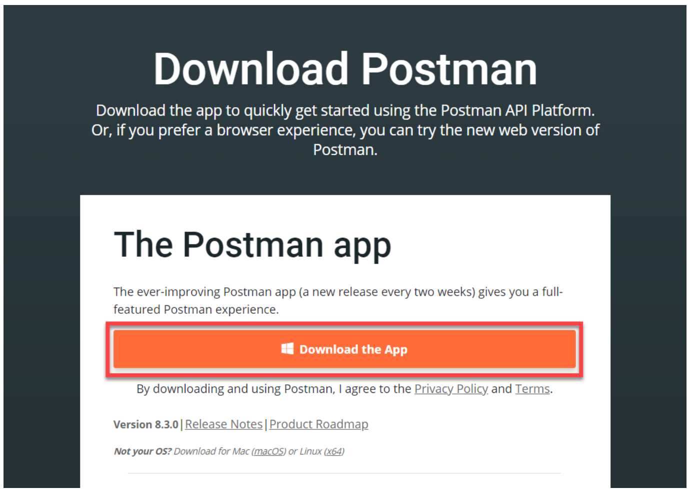
In this part, you will use an external application, in this case Postman, to login into the DX Experience API and retrieve any DX content as JSON. By running this in Postman, you should get a good idea how you may use the Experience API in any other external application.

Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster. It gives you a good idea on how you may use the Experience APIs and allow you to use it in other applications later. You may use a free version of it in this lab.

1. You may use Postman online or use the Postman client. Access https://www.postman.com/ and click **Sign Up for Free** (or if you have an account **Sign In**).



2. If you wish to install Postman, go to the following link and download **Postman app** on your desktop as per your operating system (Windows/macOS/Linux): https://www.postman.com/downloads/.


Download Postman

Download the app to quickly get started using the Postman API Platform. Or, if you prefer a browser experience, you can try the new web version of Postman.

The Postman app

The ever-improving Postman app (a new release every two weeks) gives you a full-featured Postman experience.

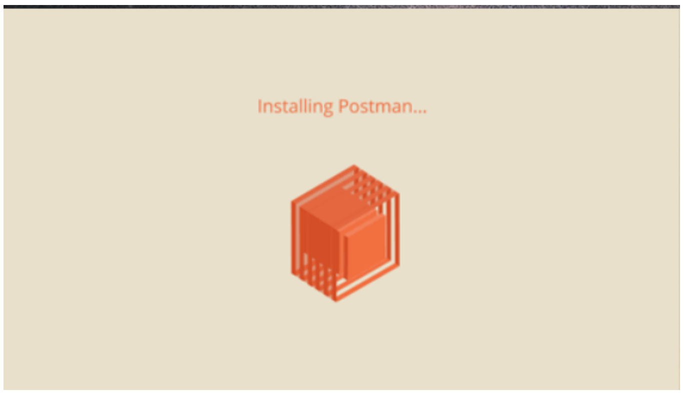
Download the App

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

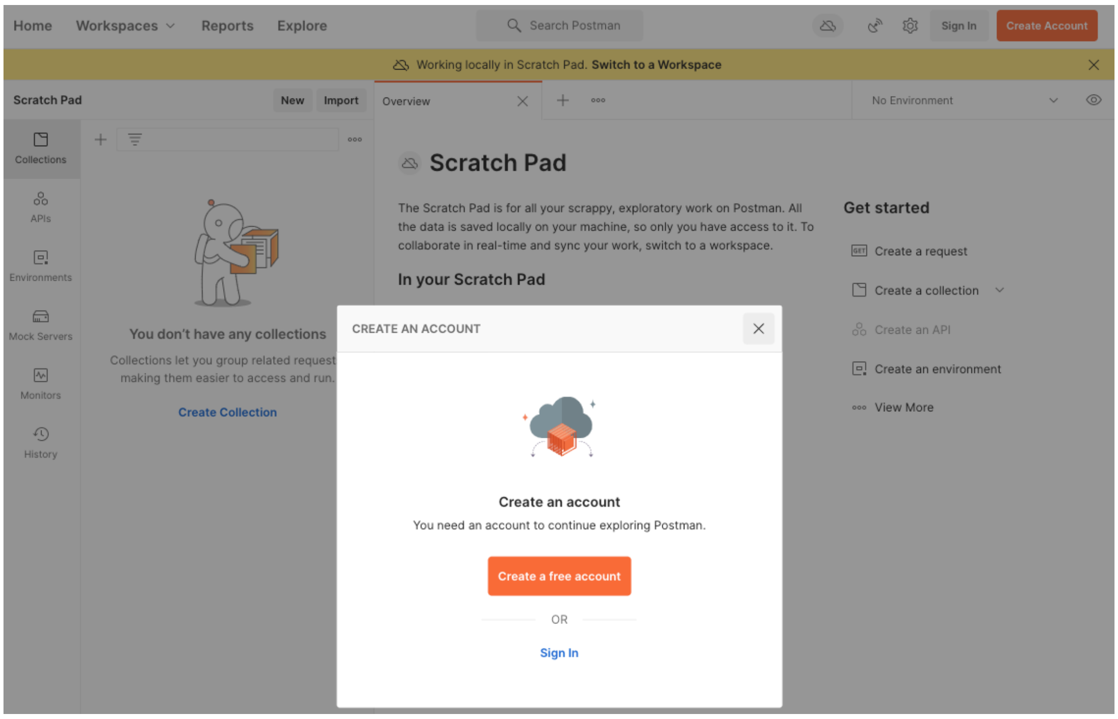
Version 8.3.0 | Release Notes | Product Roadmap

Not your OS? Download for Mac (macOS) or Linux (x64)

3. Go to the download folder and double-click the download executable for to install. The installation will automatically start.



4. The Postman site and application will appear as follows. In the case of the Postman application, you can create an account, but it is not needed to complete the lab. Otherwise, close the dialog.

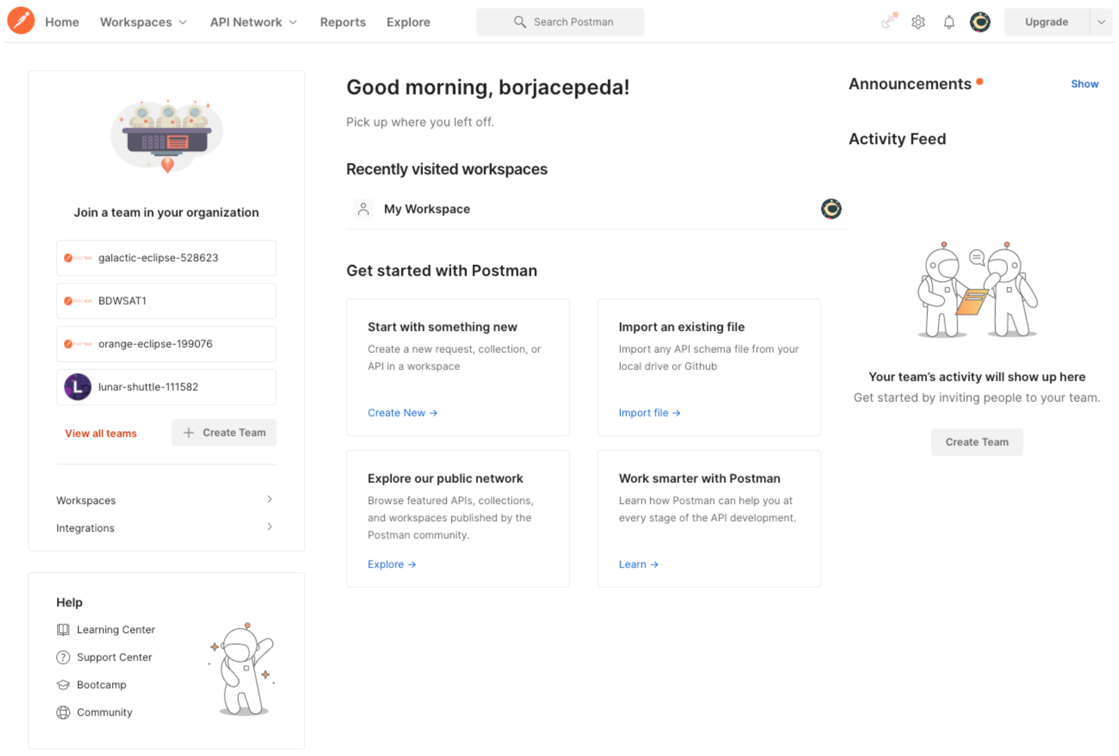


Collections let you group related requests, making them easier to access and run.

The Scratch Pad is for all your scrappy, exploratory work on Postman. All the data is saved locally on your machine, so only you have access to it. To collaborate in real-time and sync your work, switch to a workspace.

You need an account to continue exploring Postman.

5. If you do sign up, you will see the following page after registering and logging in.



Create a new request, collection, or API in a workspace

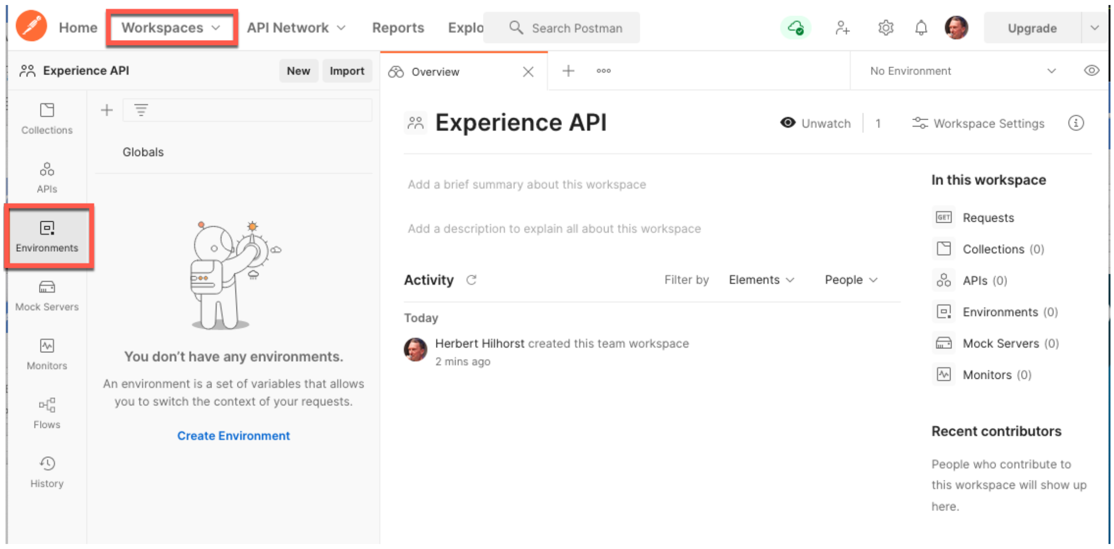
Import any API schema file from your local drive or Github

Browse featured APIs, collections, and workspaces published by the Postman community.

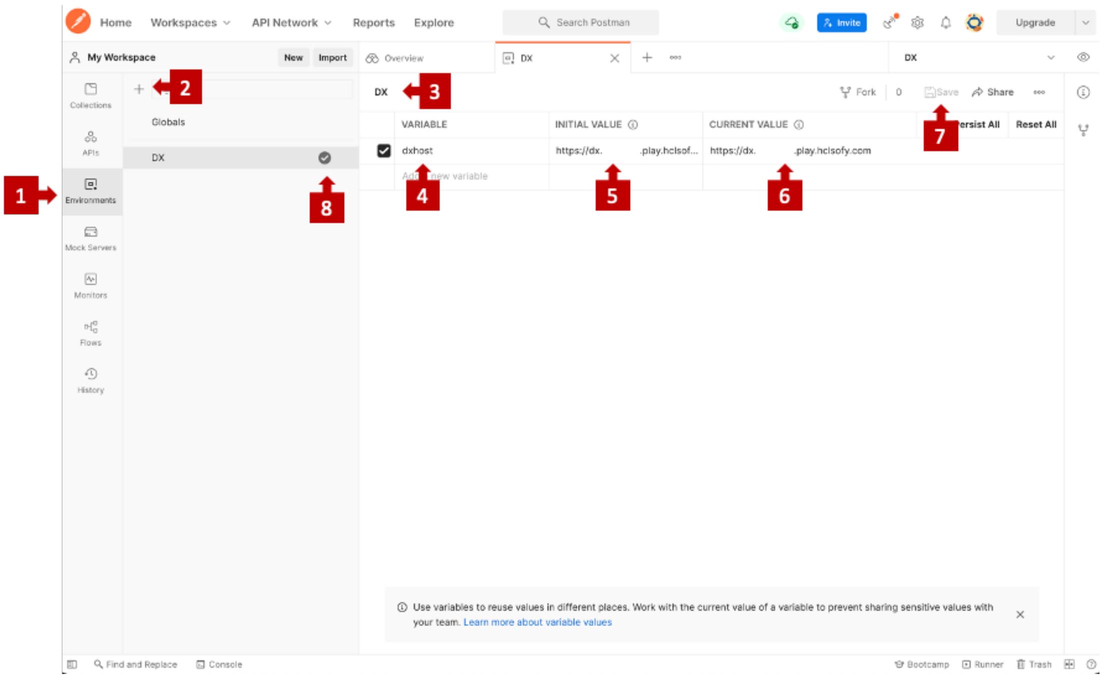
Learn how Postman can help you at every stage of the API development.

Get started by inviting people to your team.

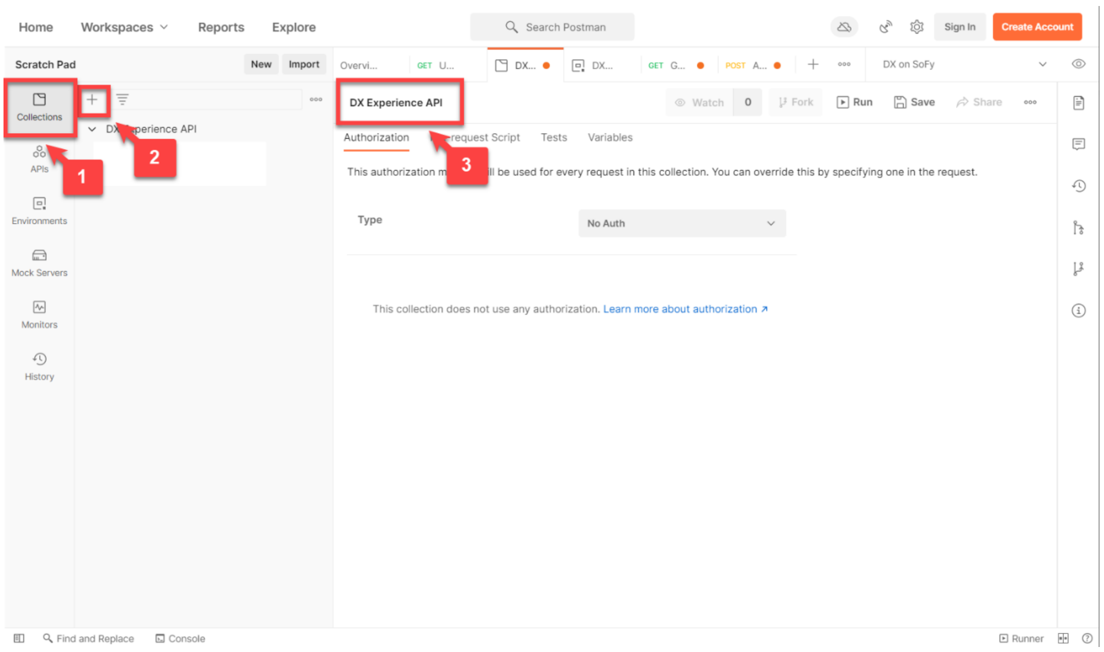
6. Now go to Workspaces, and then to the **Environments** tab (1).



7. Here you specify the details on the environment your Postman should work with. Click **+** to create a new environment, as shown in step 2. Then name the environment, e.g. **DX** (3). Create a Variable that you will use to specify the path of the DX Explorer. Name it **dxhost** (4) and set its initial and current value to the URL of your DX Explorer: ***https://dx.sbx<replace with your sand box number>.play.hclsofy.com*** (5 and 6). Finally, save your environment variable by clicking **Save** (7) and make sure to select the **Set active** checkbox near the name of the variable, under **Globals**, to make it activated (8).



8. Now create a new collection that will get the sequence of requests (API calls) to manage. Click **Collection** (1)**,** then **+** (2) and enter the name of the collection, e.g. **DX Experience API** (3). If you have any problems during the lab, you may use the downloaded file **DX Experience API.postman_collection.json** that you may import as a collection.

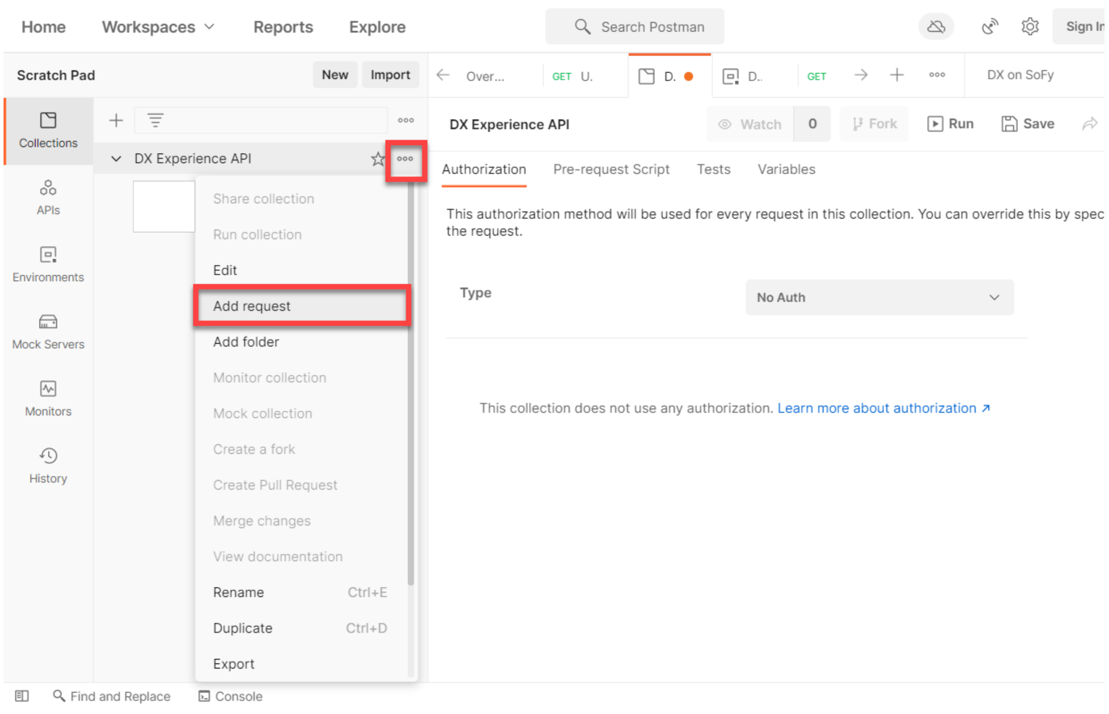


This authorization method will be used for every request in this collection. You can override this by specifying one in the request.

Type: No Auth

This collection does not use any authorization. Learn more about authorization ↗

9. In a collection, you manage one or multiple requests. You want to log in first and the get the JSON of a content. Add a new request to the collection by opening the collection menu and selecting **Add request**.

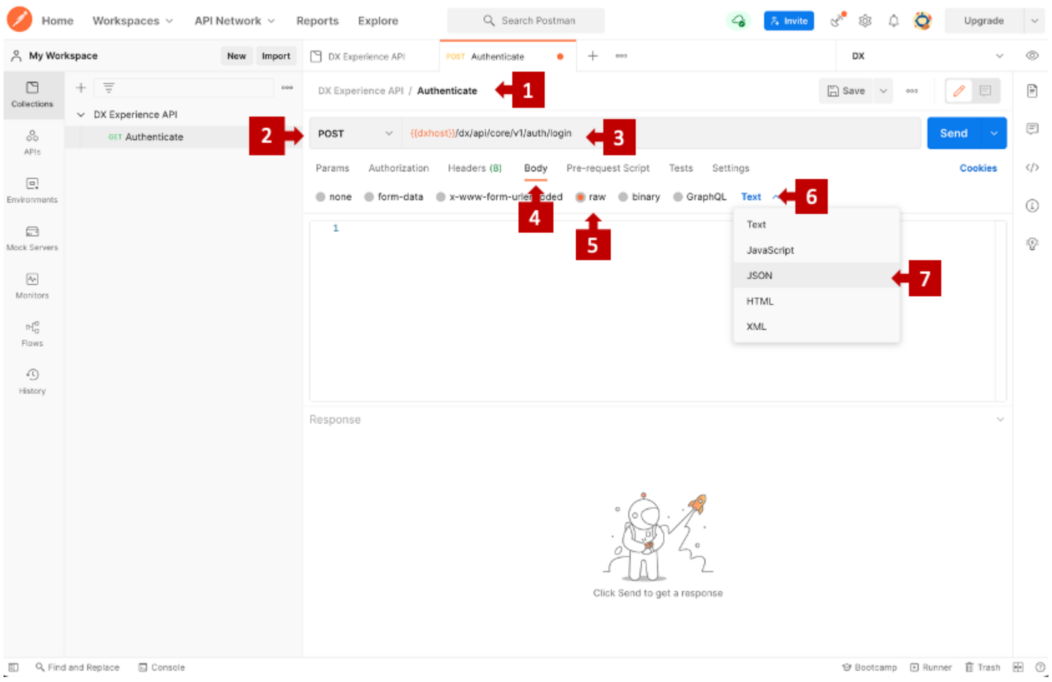


This authorization method will be used for every request in this collection. You can override this by spec the request.

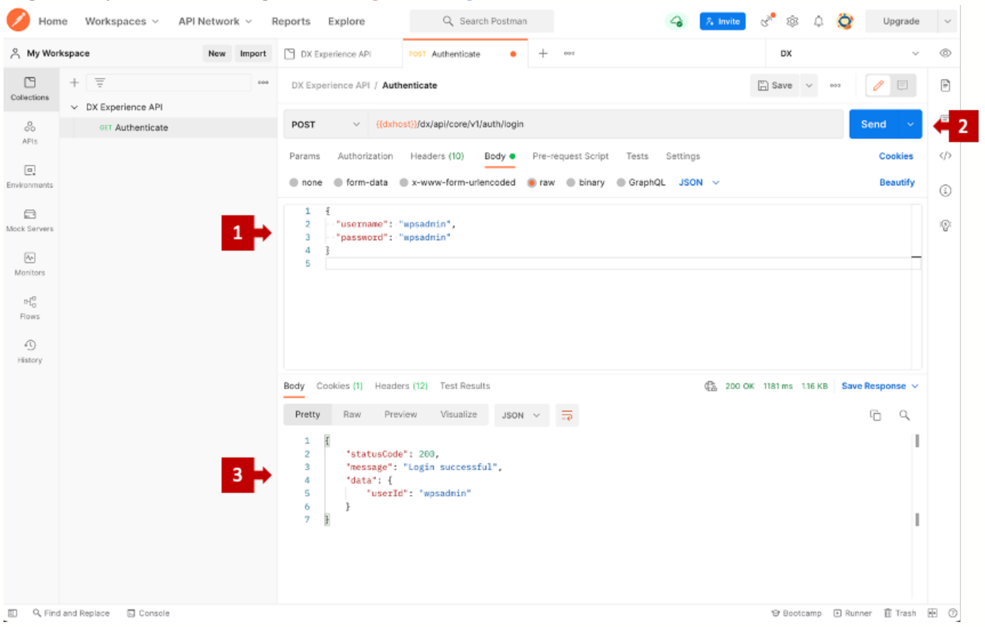
Type: No Auth

This collection does not use any authorization. Learn more about authorization ↗

10. Then create the **Authenticate** request. Name it **Authenticate** (1). The login is a Post request, so select the **POST** type of request (2). Then enter the request URL using the dxhost variable you have just created, as you have found in the Explorer and used in the previous part of this lab, as following **{{dxhost}}/dx/api/core/v1/auth/login** (3)**.** Go to the **Body** tab of the request (4). Select the **raw** radio button (5). Open the dropdown to select the type of input (6). Select **JSON** (7).



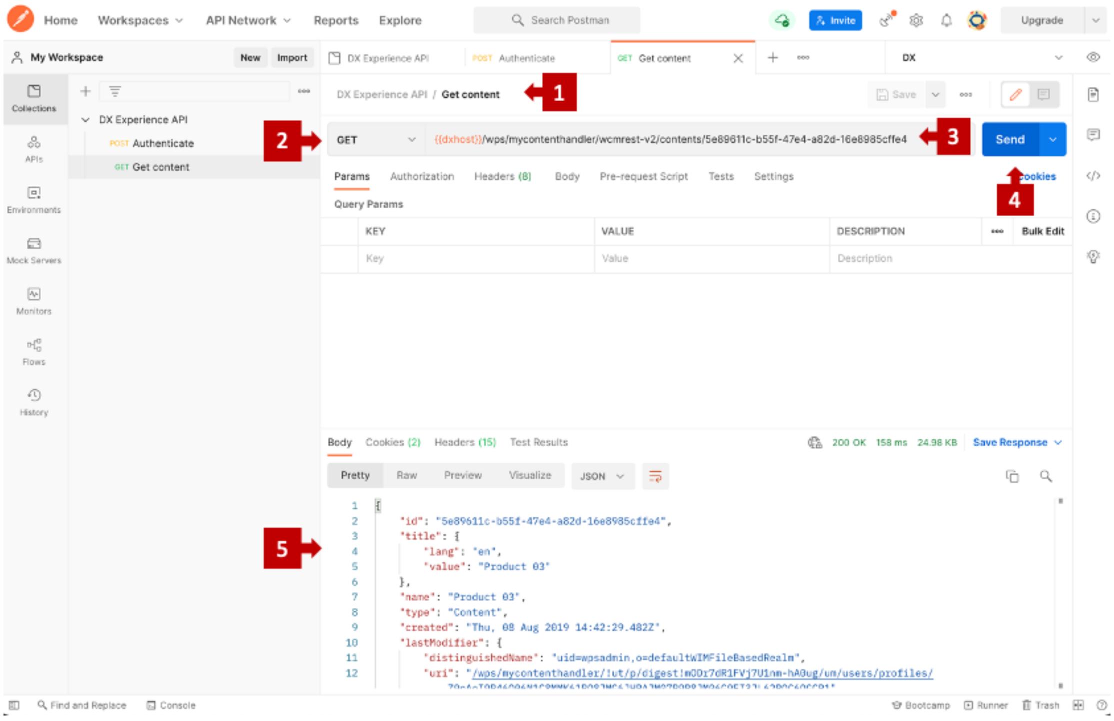
11. Enter (1) the JSON body value, like in the previous part, as
{
  "username": "wpsadmin",
  "password": "wpsadmin"
}
Then execute this request by clicking **Send** (2) and verify your response (3) for a successful login response message message": "Login successful":



```
{
    "username": "wpsadmin",
    "password": "wpsadmin"
}
```

```
{
    "statusCode": 200,
    "message": "Login successful",
    "data": {
        "userId": "wpsadmin"
    }
}
```

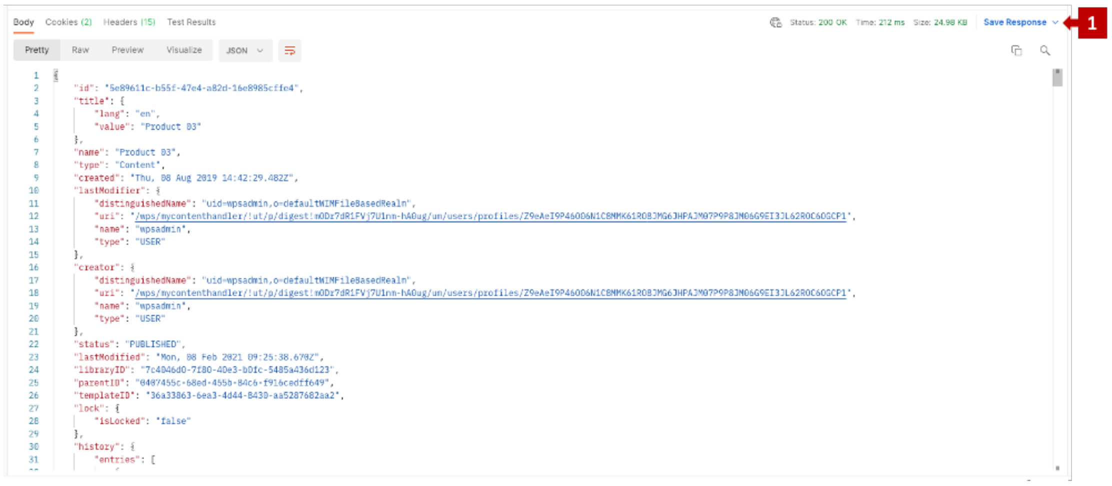
12. Then add a new request to get the content to the same collection. Open the collection menu again and selecting **Add request**. Enter the name of the request as **Get content** (1). This is a GET request, so keep **GET** as type of request (2). Enter the request URL, again obtained from the Explorer, as following **{{dxhost}}/wps/mycontenthandler/wcmrest-v2/contents/<replace with the same content UUID from the previous part>** (3). And execute this request by clicking on **Send** (4). Your response should appear under **Body** (5).



```
GET  {{dxhost}}/wps/mycontenthandler/wcmrest-v2/contents/5e89611c-b55f-47e4-a82d-16e8985cffe4
```

```
{
    "id": "5e89611c-b55f-47e4-a82d-16e8985cffe4",
    "title": {
        "lang": "en",
        "value": "Product 03"
    },
    "name": "Product 03",
    "type": "Content",
    "created": "Thu, 08 Aug 2019 14:42:29.482Z",
    "lastModifier": {
        "distinguishedName": "uid=wpsadmin,o=defaultWIMFileBasedRealm",
        "uri": "/wps/mycontenthandler/!ut/p/digest!m0Dr7dR1FVj7U1nn-hA0ug/um/users/profiles/
```

13. Verify that the execution was successful by checking the response body. You should receive the content **Product 03** in the JSON format.



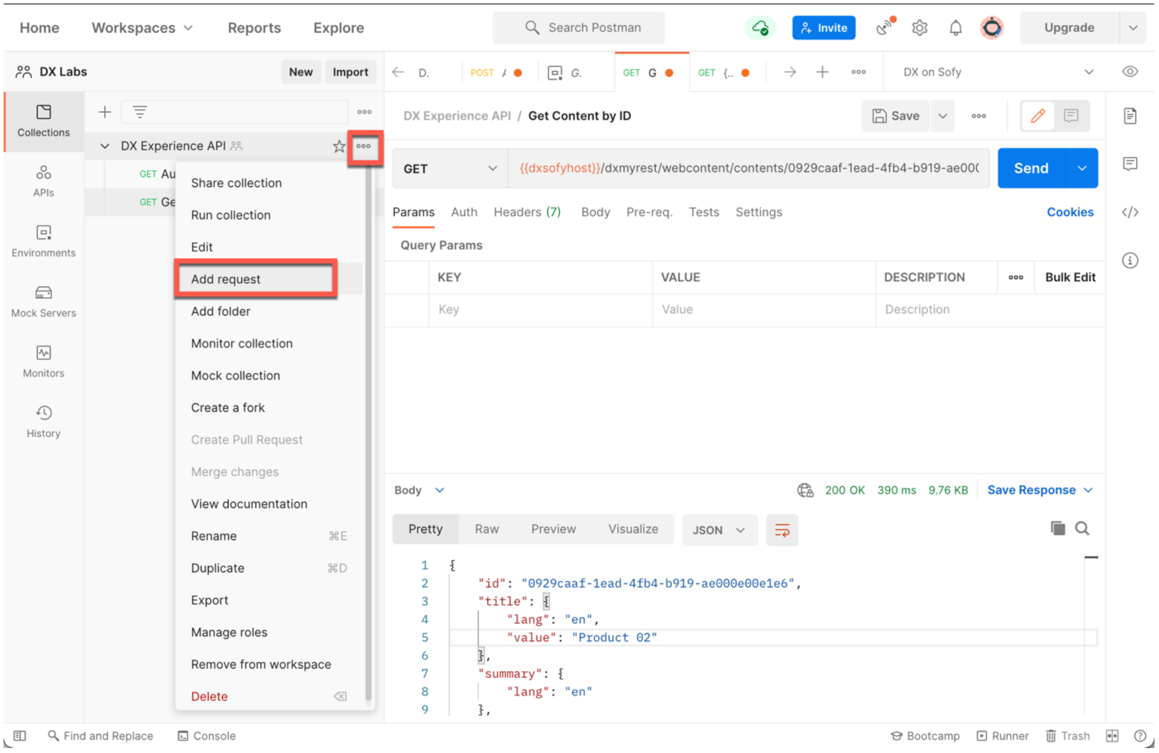
```
{
    "id": "5e89611c-b55f-47e4-a82d-16e8985cffe4",
    "title": {
        "lang": "en",
        "value": "Product 03"
    },
    "name": "Product 03",
    "type": "Content",
    "created": "Thu, 08 Aug 2019 14:42:29.482Z",
    "lastModifier": {
        "distinguishedName": "uid=wpsadmin,o=defaultWIMFileBasedRealm",
        "uri": "/wps/mycontenthandler/!ut/p/digest!m0Dr7dR1FVj7U1nn-hA0ug/um/users/profiles/Z9eAeI9P46006N1C8MMK61R08JMG6JHPAJM07P9P8JM06G9EI3JL62R0C60GCP1",
        "name": "wpsadmin",
        "type": "USER"
    },
    "creator": {
        "distinguishedName": "uid=wpsadmin,o=defaultWIMFileBasedRealm",
        "uri": "/wps/mycontenthandler/!ut/p/digest!m0Dr7dR1FVj7U1nn-hA0ug/um/users/profiles/Z9eAeI9P46006N1C8MMK61R08JMG6JHPAJM07P9P8JM06G9EI3JL62R0C60GCP1",
        "name": "wpsadmin",
        "type": "USER"
    },
    "status": "PUBLISHED",
    "lastModified": "Mon, 08 Feb 2021 09:25:38.670Z",
    "libraryID": "7c4046d0-7f80-40e3-b0fc-5485a436d123",
    "parentID": "0487455c-68ed-455b-84c6-f916cedff649",
    "templateID": "36a33863-6ea3-4d44-8430-aa5287682aa2",
    "lock": {
        "isLocked": "false"
    },
    "history": {
        "entries": [
```

Congratulations! You have now learnt how to call the experience API methods to receive content from DX in JSON format externally.

Part 3: Create new DX Content Externally using Experience API

In this part, you will create a new DX content, **Product 09**, using the Experience API externally, again using Postman.

1. Add again a new request to create a new content to your current collection. Open the collection menu again and select **Add request**.

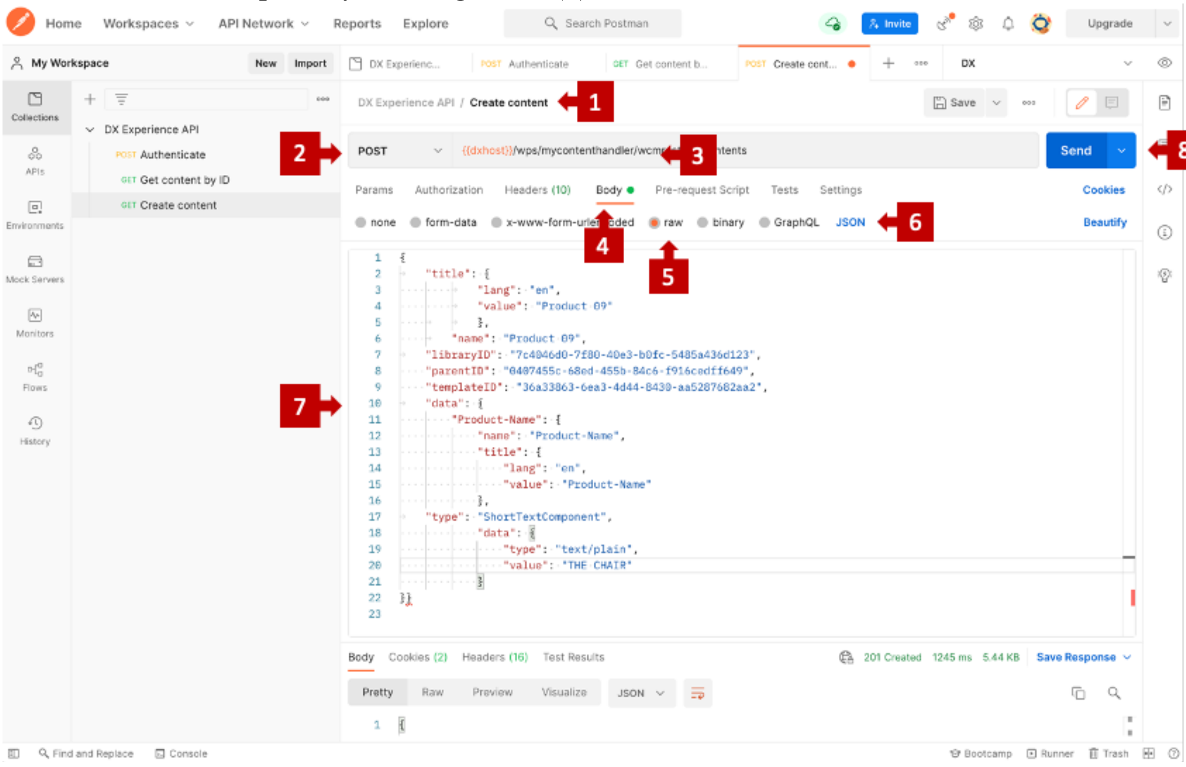


```
{
    "id": "0929caaf-1ead-4fb4-b919-ae000e00e1e6",
    "title": {
        "lang": "en",
        "value": "Product 02"
    },
    "summary": {
        "lang": "en"
    },
```

2. Then enter the details of this create content request. Name it **Create content** (1) and select the **POST** type of request (2). Then enter the request URL you saw in the first part: **{{dxhost}}/wps/mycontenthandler/wcmrest-v2/contents** (3). In the **Body** tab (4), select the input type as **raw** with the radio button (5). Select the **JSON** input type (6). Enter the JSON below to create a new content, using the same template as of the **Product 03** content fetched above and using the same site area with new name **Product 09** and new **Product-Name** short text element with value '**THE CHAIR**" (or anything else that allows you to distinguish it easily later on) (7):

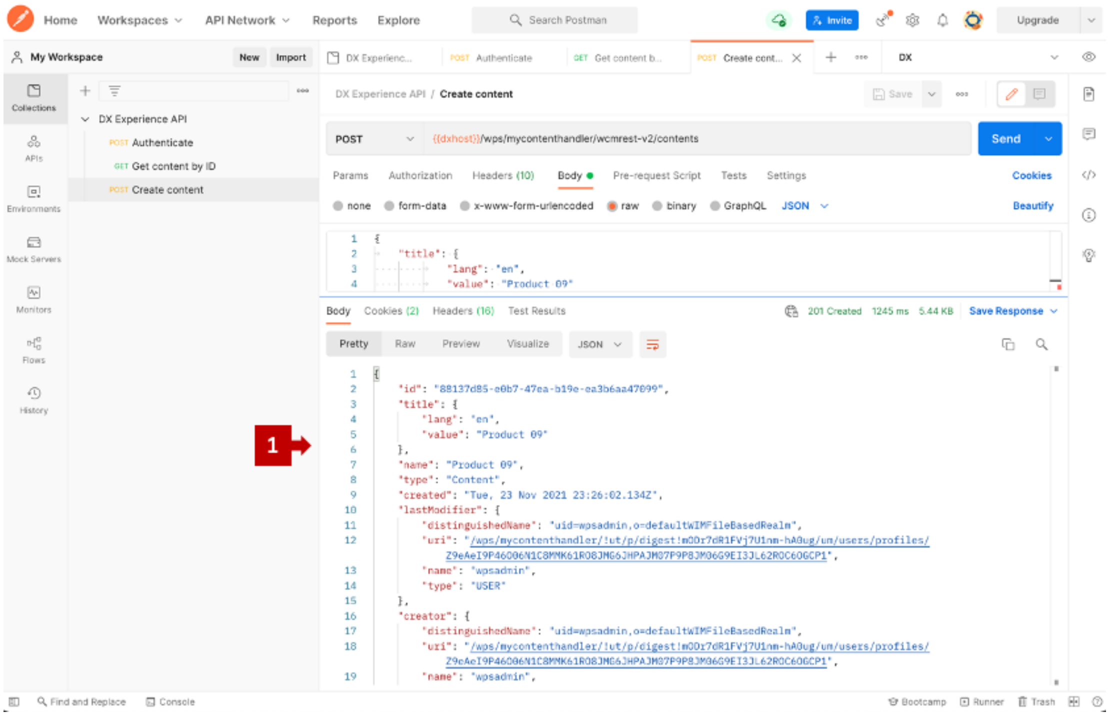
```
{
"title": {
        "lang": "en",
        "value": "Product 09"
        },
    "name": "Product 09",
"libraryID": "7c4046d0-7f80-40e3-b0fc-5485a436d123",
"parentID": "0407455c-68ed-455b-84c6-f916cedff649",
"templateID": "36a33863-6ea3-4d44-8430-aa5287682aa2",
"data": {
    "Product-Name": {
        "name": "Product-Name",
        "title": {
            "lang": "en",
            "value": "Product-Name"
        },
        "type": "ShortTextComponent",
        "data": {
            "type": "text/plain",
            "value": "THE CHAIR"
        }
}
}
```

And execute this request by clicking **Send** (8).



```
{
    "title": {
        "lang": "en",
        "value": "Product 09"
        },
      "name": "Product 09",
    "libraryID": "7c4046d0-7f80-40e3-b0fc-5485a436d123",
    "parentID": "0407455c-68ed-455b-84c6-f916cedff649",
    "templateID": "36a33863-6ea3-4d44-8430-aa5287682aa2",
    "data": {
      "Product-Name": {
        "name": "Product-Name",
        "title": {
          "lang": "en",
          "value": "Product-Name"
        },
    "type": "ShortTextComponent",
        "data": {
          "type": "text/plain",
          "value": "THE CHAIR"
        }
}}
```

3. Verify that the execution was successful by checking the response body. You should receive the content **Product 09** in the JSON format in the Body section (1).



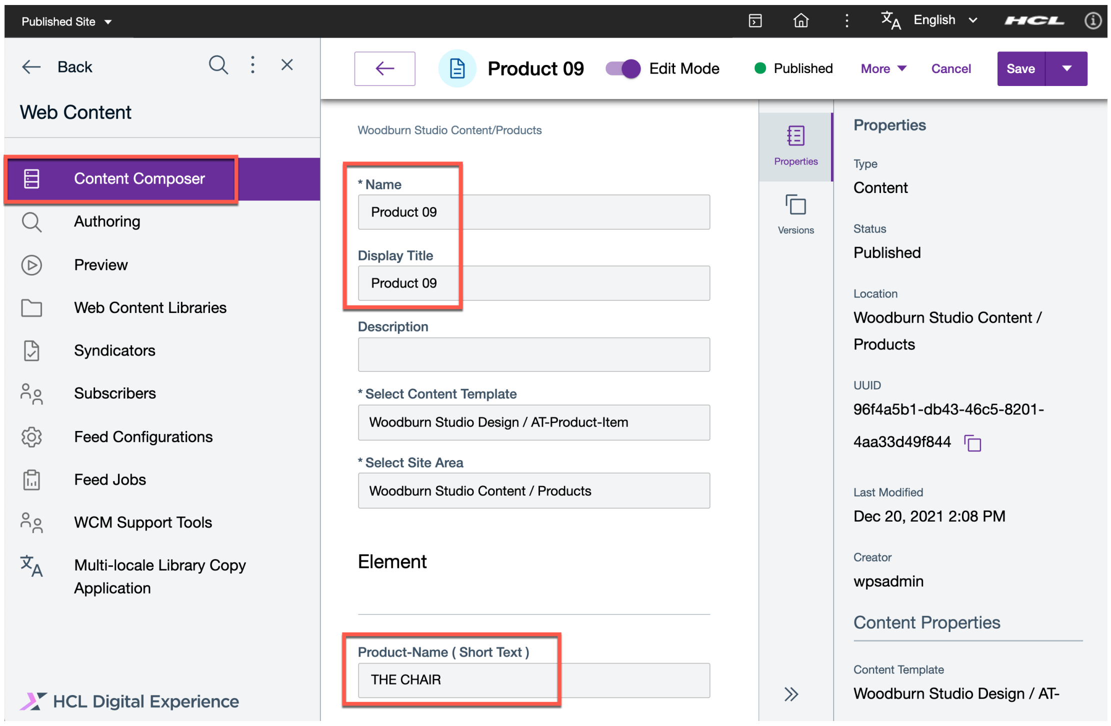
POST {{dxhost}}/wps/mycontenthandler/wcmrest-v2/contents

```
{
    "title": {
        "lang": "en",
        "value": "Product 09"
```

201 Created 1245 ms 5.44 KB

```
{
    "id": "88137d85-e0b7-47ea-b19e-ea3b6aa47099",
    "title": {
        "lang": "en",
        "value": "Product 09"
    },
    "name": "Product 09",
    "type": "Content",
    "created": "Tue, 23 Nov 2021 23:26:02.134Z",
    "lastModifier": {
        "distinguishedName": "uid=wpsadmin,o=defaultWIMFileBasedRealm",
        "uri": "/wps/mycontenthandler/!ut/p/digest!m0Dr7dR1FVj7U1nm-hA0ug/um/users/profiles/Z9eAeI9P46006N1C8MMK61R08JMG6JHPAJM07P9P8JM06G9EI3JL62R0C60GCP1",
        "name": "wpsadmin",
        "type": "USER"
    },
    "creator": {
        "distinguishedName": "uid=wpsadmin,o=defaultWIMFileBasedRealm",
        "uri": "/wps/mycontenthandler/!ut/p/digest!m0Dr7dR1FVj7U1nm-hA0ug/um/users/profiles/Z9eAeI9P46006N1C8MMK61R08JMG6JHPAJM07P9P8JM06G9EI3JL62R0C60GCP1",
        "name": "wpsadmin",
```

4. And verify in the Content Composer that this new product is added.



Product 09 — Edit Mode — Published

Woodburn Studio Content/Products

* Name: Product 09

Display Title: Product 09

Description:

* Select Content Template: Woodburn Studio Design / AT-Product-Item

* Select Site Area: Woodburn Studio Content / Products

Element

Product-Name (Short Text): THE CHAIR

Properties

Type: Content

Status: Published

Location: Woodburn Studio Content / Products

UUID: 96f4a5b1-db43-46c5-8201-4aa33d49f844

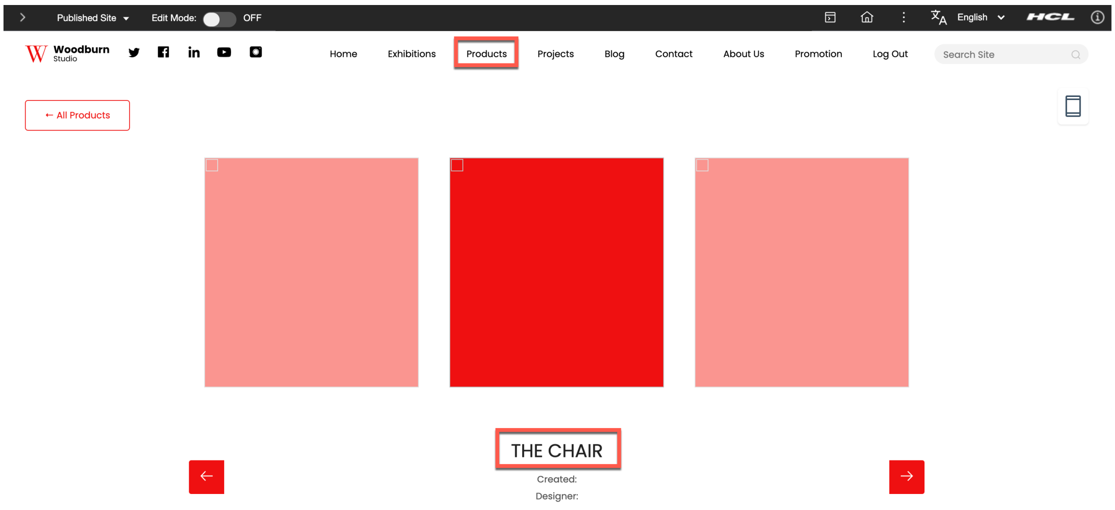
Last Modified: Dec 20, 2021 2:08 PM

Creator: wpsadmin

Content Properties

Content Template: Woodburn Studio Design / AT-

5. You may also check in the Woodburn Studio products page if your new product is added and how it looks like. There are no images assigned to it yet. You will add that in the next part.

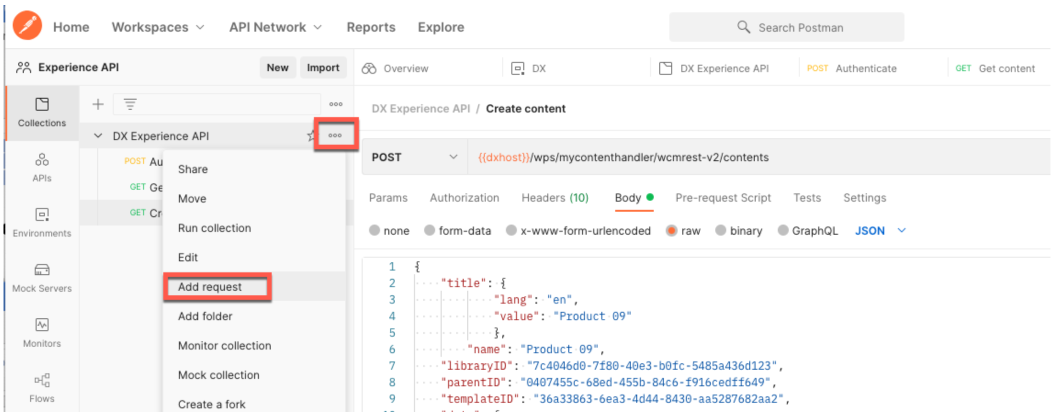


Congratulations! You have created new content on the DX server, externally, using the Postman application.

Part 4: Upload Image to DAM and Add to Web Content

In this part of the lab, you will use the Digital Asset Management API or short, DAM API, to create a new collection and upload an image to the collection. Then you will use the WCM V2 API again to add the image to the content you created in the previous part. And learn how to reuse this collection of API calls in any programming language.

1. Add a new request to create this Collection first. Open the collection menu and select **Add request**.

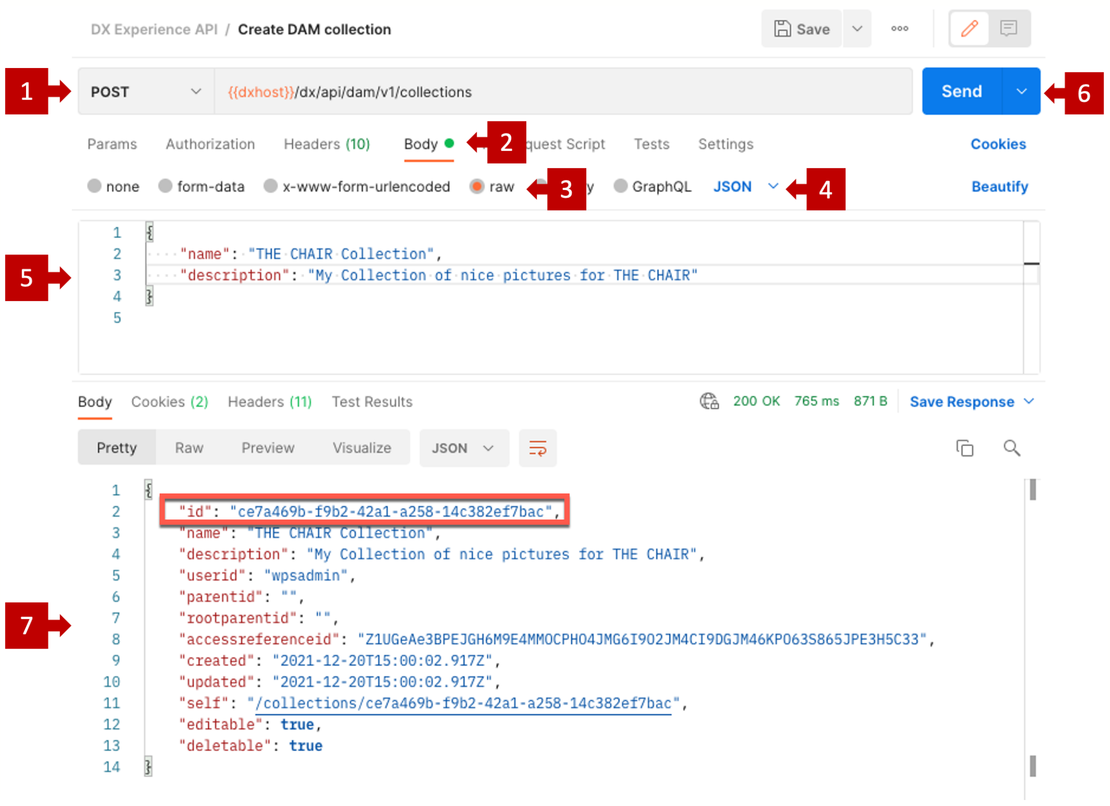


```
POST {{dxhost}}/wps/mycontenthandler/wcmrest-v2/contents
```

```
{
    "title": {
            "lang": "en",
            "value": "Product 09"
            },
        "name": "Product 09",
    "libraryID": "7c4046d0-7f80-40e3-b0fc-5485a436d123",
    "parentID": "0407455c-68ed-455b-84c6-f916cedff649",
    "templateID": "36a33863-6ea3-4d44-8430-aa5287682aa2",
```

2. Then enter the details to create the new Collection. Name it **Create DAM collection**. Select the POST type of request. Enter the request URL that you can find using the DAM API Explorer: *{{dxhost}}/dx/api/dam/v1/collections*. In the **Body** tab (2), select the input type as **raw** (3) from the radio button. Select the **JSON** (4) input type. Enter the JSON below to create a new Collection with your name and description of choice, e.g. with THE CHAIR in the name as shown (5). Click **Send** (6) and check the response JSON with the id of your new DAM collection (7). Copy this id, as you need it in the next step. You could also put this into a variable and use this later. This allows you further automation. More details on using variables in https://learning.postman.com/docs/sending-requests/variables/.

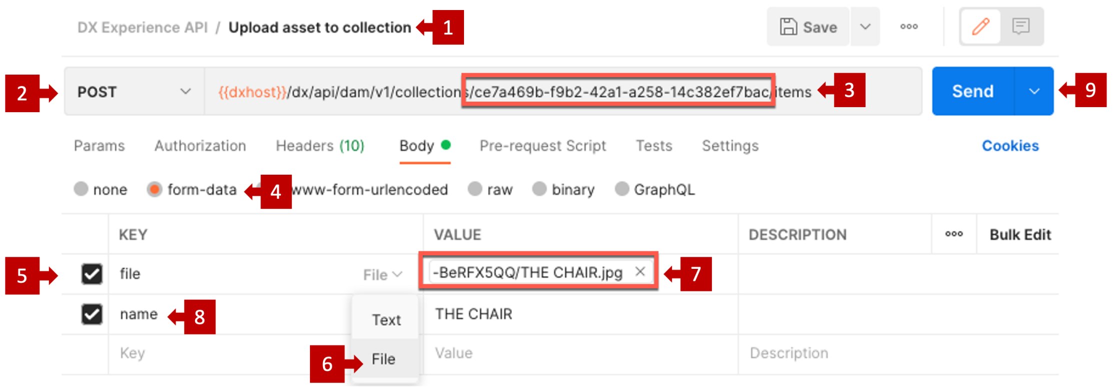
```
{
    "name": "THE CHAIR Collection",
    "description": "My Collection of nice pictures for THE CHAIR"
}
```



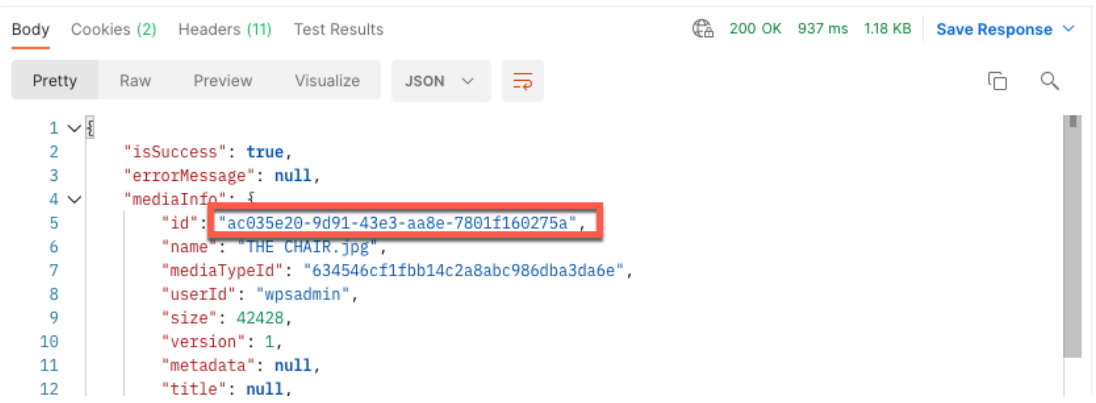
```
{
    "name": "THE CHAIR Collection",
    "description": "My Collection of nice pictures for THE CHAIR"
}
```

```
{
    "id": "ce7a469b-f9b2-42a1-a258-14c382ef7bac",
    "name": "THE CHAIR Collection",
    "description": "My Collection of nice pictures for THE CHAIR",
    "userid": "wpsadmin",
    "parentid": "",
    "rootparentid": "",
    "accessreferenceid": "Z1UGeAe3BPEJGH6M9E4MMOCPHO4JMG6I9O2JM4CI9DGJM46KPO63S865JPE3H5C33",
    "created": "2021-12-20T15:00:02.917Z",
    "updated": "2021-12-20T15:00:02.917Z",
    "self": "/collections/ce7a469b-f9b2-42a1-a258-14c382ef7bac",
    "editable": true,
    "deletable": true
}
```

3. Now upload an image to your new collection to be used the content you have created. Add a new request. Open the collection menu again and select **Add request**. Then enter the details to create the new request. Name it **Upload asset to collection** (1). Select the **POST** type of request (2). Enter the request URL using the **id** provided in the previous step: **{{dxhost}}/dx/api/dam/v1/collections/<Your DAM collection ID>/items** (3). In the Body tab, select the form-data input type, as this allows you to select the different values easily (4). Then add the key-value pair for the **file** (5), select the type **File** (6), and use **Select Files** to select the picture of your choice from your workstation. You may use the image from the lab resources you have downloaded. (7). Create a new key-value pair for the **name** (8) of the asset to be uploaded to the collection. Click **Send** (9).

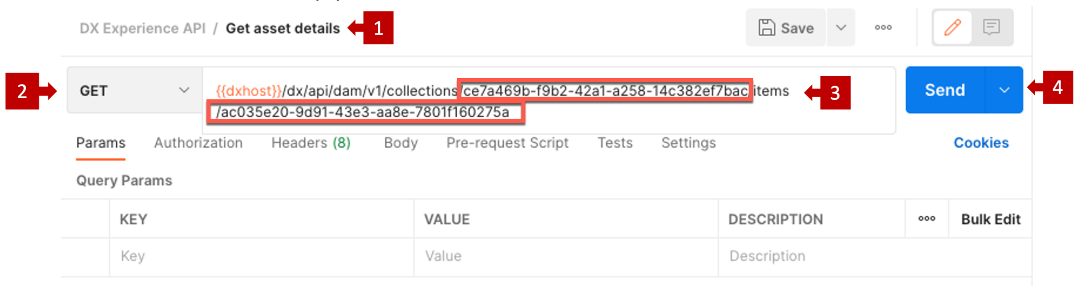


4. Then copy the id of the uploaded asset from the JSON in the response. You will use this for your content you will then update.



```
{
    "isSuccess": true,
    "errorMessage": null,
    "mediaInfo": {
        "id": "ac035e20-9d91-43e3-aa8e-7801f160275a",
        "name": "THE CHAIR.jpg",
        "mediaTypeId": "634546cf1fbb14c2a8abc986dba3da6e",
        "userId": "wpsadmin",
        "size": 42428,
        "version": 1,
        "metadata": null,
        "title": null,
```

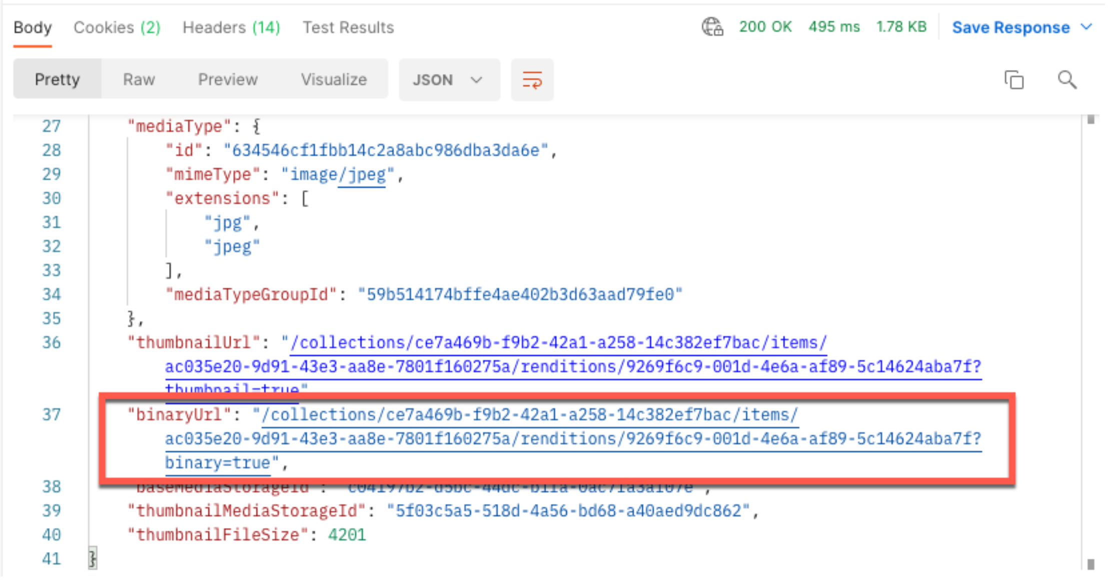
5. You then need to get the image details to add it to your new content. You can get the details from the response JSON. Open the collection menu again and select **Add request**. Then enter the details to create the new request. Name it **Get asset details** (1). Select the **GET** type of request (2). Enter the request URL using the **collection id** and asset id from the previous step: **{{dxhost}}/dx/api/dam/v1/collections/<Your DAM collection ID>/items/<Your asset id>** (3) and click **Send** (4).



```
{{dxhost}}/dx/api/dam/v1/collections/ce7a469b-f9b2-42a1-a258-14c382ef7bac/items
/ac035e20-9d91-43e3-aa8e-7801f160275a
```

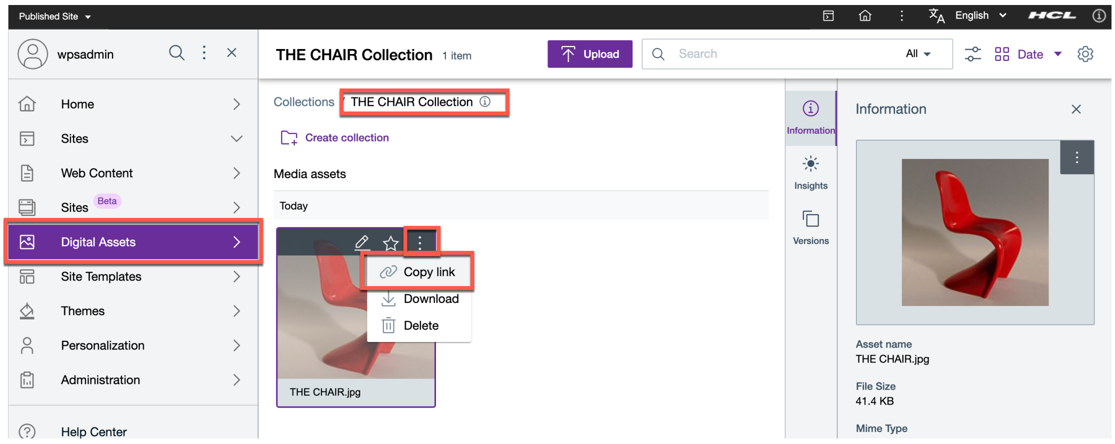
KEY	VALUE	DESCRIPTION
Key	Value	Description

6. At the bottom of the JSON response, you will find a property called “binaryUrl”. This is the URL of your image. You will notice that the hostname and the DAM API context are missing in that “binaryURL”. Therefore, you will have to add **{{dxhost}}/dx/api/dam/v1/** to make it a working URL. Copy this.



```
    "mediaType": {
        "id": "634546cf1fbb14c2a8abc986dba3da6e",
        "mimeType": "image/jpeg",
        "extensions": [
            "jpg",
            "jpeg"
        ],
        "mediaTypeGroupId": "59b514174bffe4ae402b3d63aad79fe0"
    },
    "thumbnailUrl": "/collections/ce7a469b-f9b2-42a1-a258-14c382ef7bac/items/
        ac035e20-9d91-43e3-aa8e-7801f160275a/renditions/9269f6c9-001d-4e6a-af89-5c14624aba7f?
        thumbnail=true",
    "binaryUrl": "/collections/ce7a469b-f9b2-42a1-a258-14c382ef7bac/items/
        ac035e20-9d91-43e3-aa8e-7801f160275a/renditions/9269f6c9-001d-4e6a-af89-5c14624aba7f?
        binary=true",
    "baseMediaStorageId": "c04197b2-d5bc-44dc-b11a-0ac71a3a107e",
    "thumbnailMediaStorageId": "5f03c5a5-518d-4a56-bd68-a40aed9dc862",
    "thumbnailFileSize": 4201
}
```

7. Another way is to get the image URL is to use the **Digital Assets** in the Practitioner Studio. In your new collection, select your uploaded image, open the asset menu and click **Copy link**.

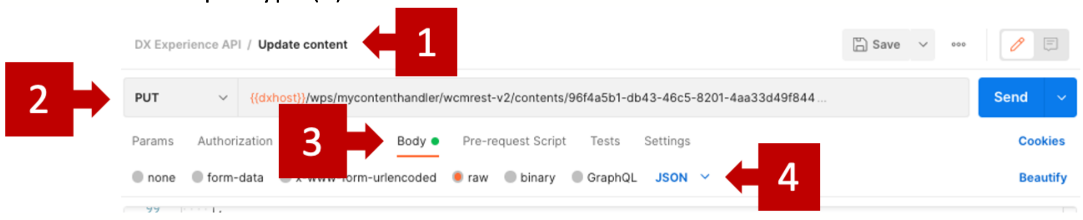


Asset name
THE CHAIR.jpg

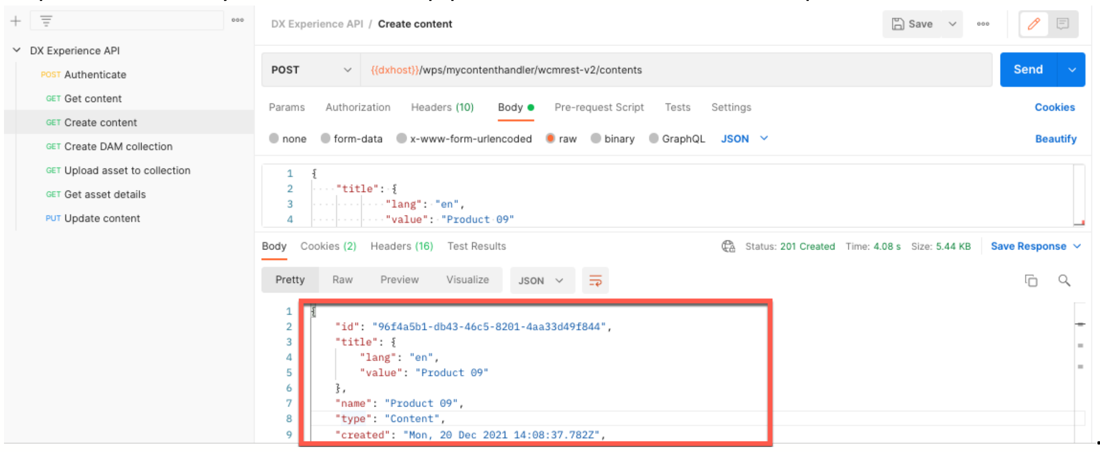
File Size
41.4 KB

Mime Type

8. Then add a new request to update your previously created content. Open the collection menu again and select **Add request**. Name it **Update content**. Select the **PUT** type of request and enter the request URL using the ID of the content: **{{dxhost}}/wps/mycontenthandler/wcmrest-v2/contents/<UUID of content to be updated>** (1). In the Body tab (2), select the input type as raw from the radio button (3) and select the JSON input type (4).



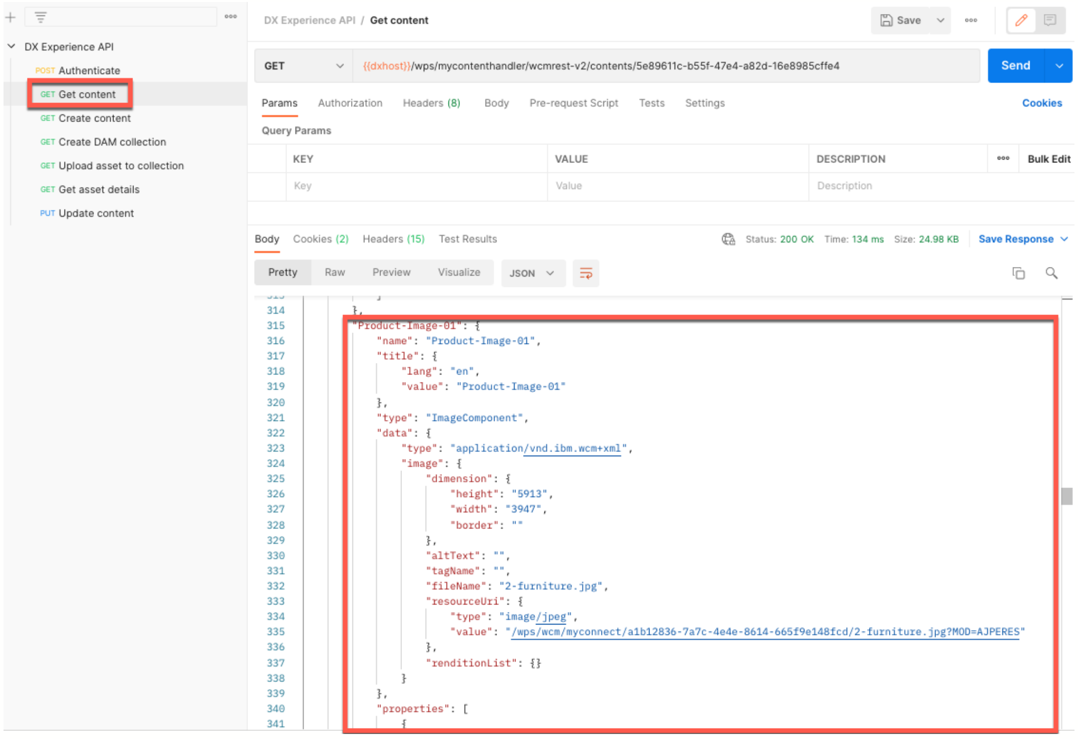
9. Then you need to create the right JSON to update your existing content. From your **Create content** request, copy the full Body response JSON and put that into the **Update content** request JSON entry. This was the top part of the Create content response.



```
{
    "title": {
            "lang": "en",
            "value": "Product 09"
```

```
{
    "id": "96f4a5b1-db43-46c5-8201-4aa33d49f844",
    "title": {
        "lang": "en",
        "value": "Product 09"
    },
    "name": "Product 09",
    "type": "Content",
    "created": "Mon, 20 Dec 2021 14:08:37.782Z",
```

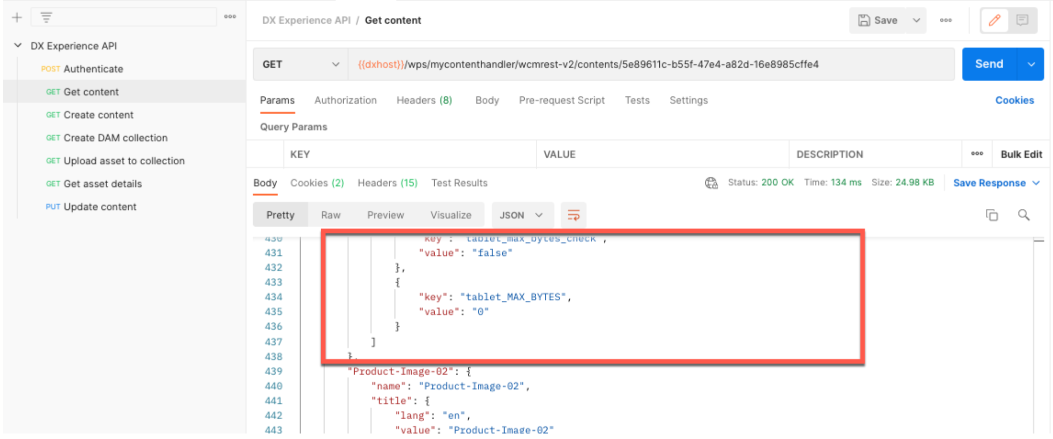
10. Then from the Get content Body response, just get the **Product-Image-01** data entry. Here you see where it starts:



```
GET {{dxhost}}/wps/mycontenthandler/wcmrest-v2/contents/5e89611c-b55f-47e4-a82d-16e8985cffe4
```

```
    },
    "Product-Image-01": {
        "name": "Product-Image-01",
        "title": {
            "lang": "en",
            "value": "Product-Image-01"
        },
        "type": "ImageComponent",
        "data": {
            "type": "application/vnd.ibm.wcm+xml",
            "image": {
                "dimension": {
                    "height": "5913",
                    "width": "3947",
                    "border": ""
                },
                "altText": "",
                "tagName": "",
                "fileName": "2-furniture.jpg",
                "resourceUri": {
                    "type": "image/jpeg",
                    "value": "/wps/wcm/myconnect/a1b12836-7a7c-4e4e-8614-665f9e148fcd/2-furniture.jpg?MOD=AJPERES"
                },
                "renditionList": {}
            }
        },
        "properties": [
            {
```

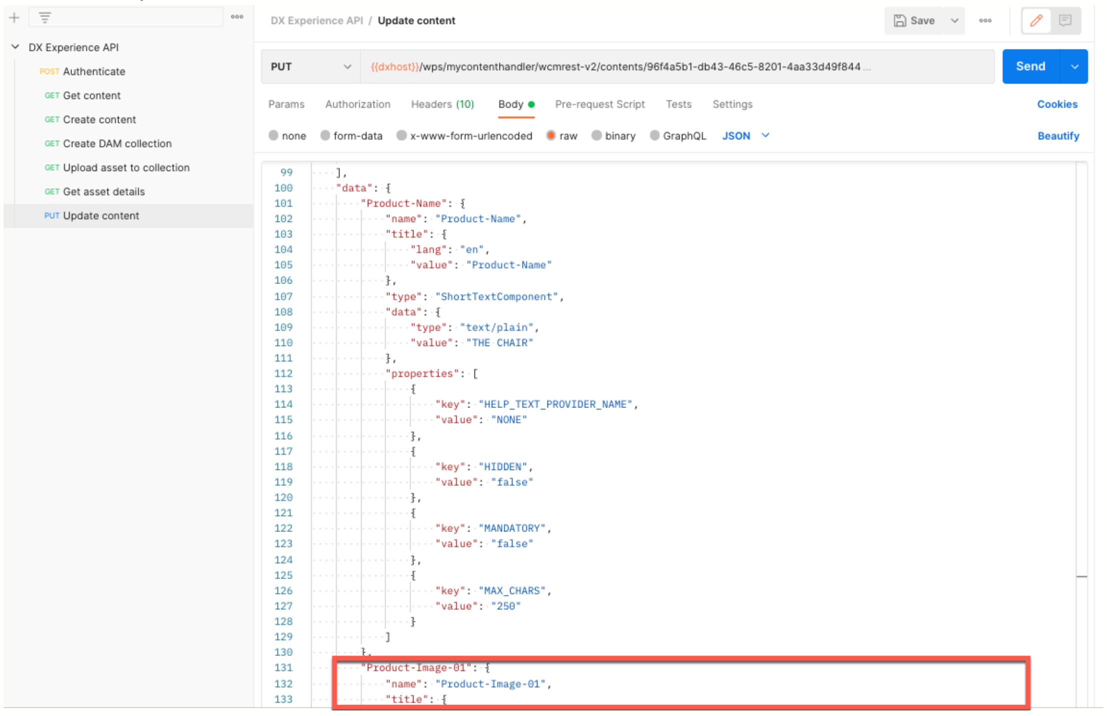
11. And it ends where Product-Image-02 starts:



```
GET {{dxhost}}/wps/mycontenthandler/wcmrest-v2/contents/5e89611c-b55f-47e4-a82d-16e8985cffe4
```

```
                    "key": "tablet_max_bytes_check",
                    "value": "false"
                },
                {
                    "key": "tablet_MAX_BYTES",
                    "value": "0"
                }
            ]
        },
        "Product-Image-02": {
            "name": "Product-Image-02",
            "title": {
                "lang": "en",
                "value": "Product-Image-02"
```

12. Copy this part and insert this after the **Product-Name** data entry in your new **Update content** request:



PUT {{dxhost}}/wps/mycontenthandler/wcmrest-v2/contents/96f4a5b1-db43-46c5-8201-4aa33d49f844 ...

```
    ],
    "data": {
        "Product-Name": {
            "name": "Product-Name",
            "title": {
                "lang": "en",
                "value": "Product-Name"
            },
            "type": "ShortTextComponent",
            "data": {
                "type": "text/plain",
                "value": "THE CHAIR"
            },
            "properties": [
                {
                    "key": "HELP_TEXT_PROVIDER_NAME",
                    "value": "NONE"
                },
                {
                    "key": "HIDDEN",
                    "value": "false"
                },
                {
                    "key": "MANDATORY",
                    "value": "false"
                },
                {
                    "key": "MAX_CHARS",
                    "value": "250"
                }
            ]
        },
        "Product-Image-01": {
            "name": "Product-Image-01",
            "title": {
```

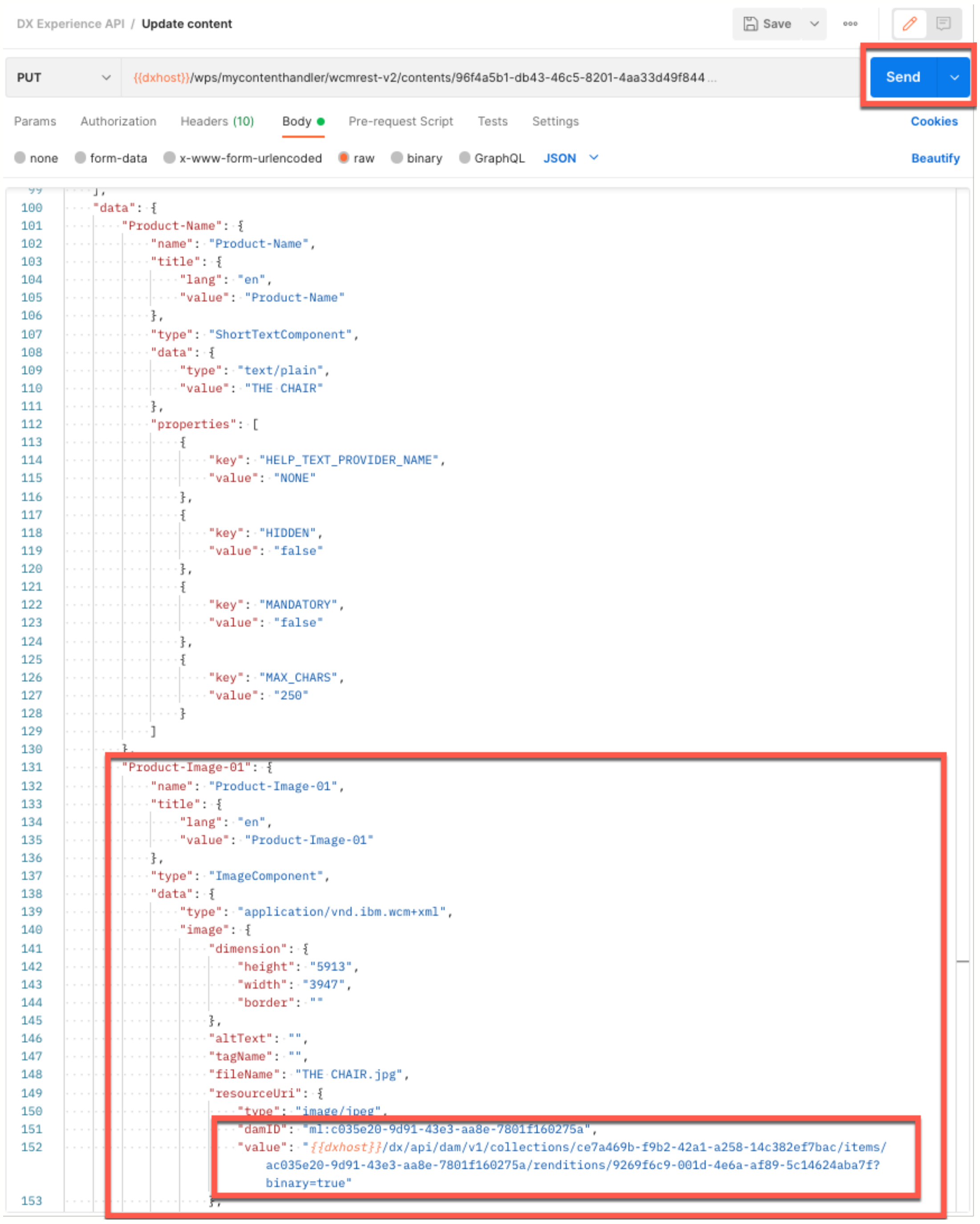
13. Then update the file name and resourceURI from

```
   "fileName": "2.1-furniture.jpg",
   "resourceUri": {
        "type": "image/jpeg",
        "value": "/wps/wcm/myconnect/c6b1f550-16ce-4dd5-9ff4-ea996ce4e88f/2.1-
furniture.jpg?MOD=AJPERES"
   },
```

14. With the details from your image file.

```
 "fileName": "<name of your file, e.g. THE CHAIR.JPG",
 "resourceUri": {
      "type": "image/jpeg",
      "damID": "ml:<PASTE HERE THE ASSET ID FROM THE RESPONSE IN STEP 2>",
      "value": "<PASTE HERE THE IMAGE LINK COPIED FROM THE DAM, or from the Get asset
details preceded with your host {{dxhost}}/dx/api/dam/v1/>"
      },
```

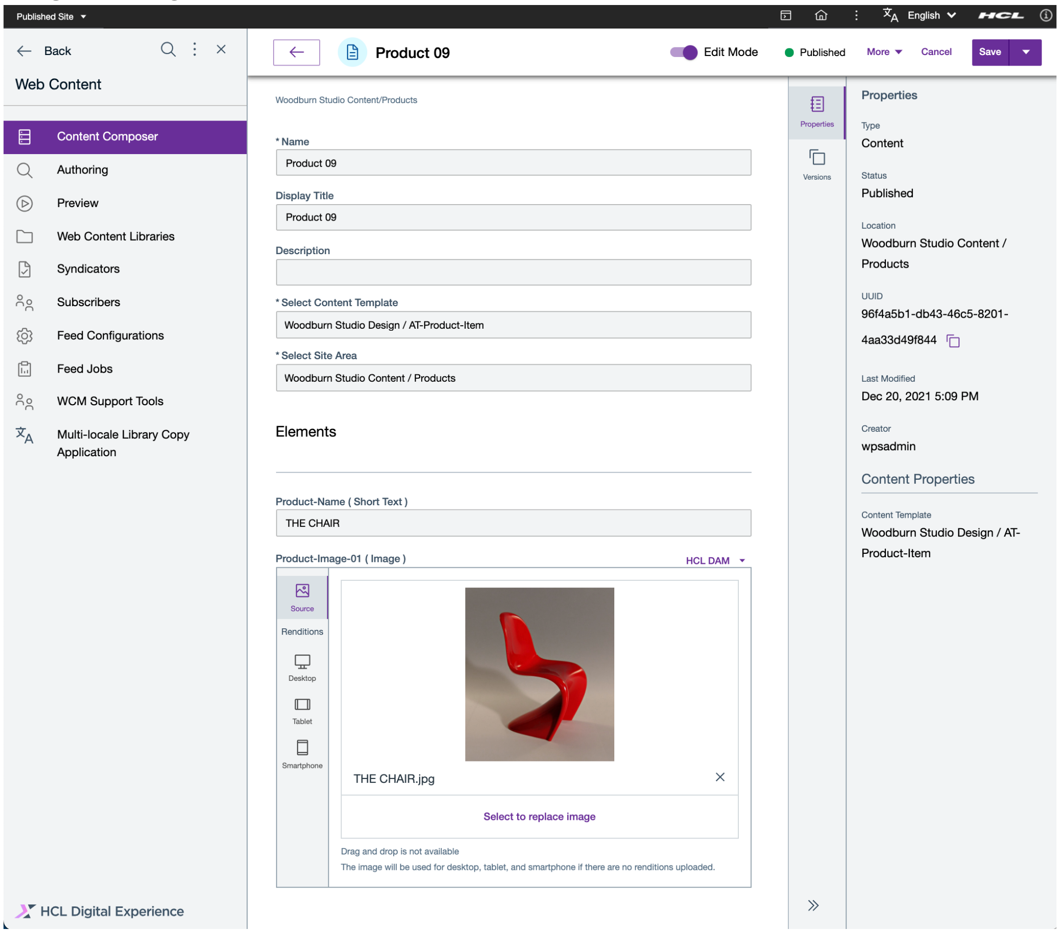
15. And click **Send** to execute.



```
PUT {{dxhost}}/wps/mycontenthandler/wcmrest-v2/contents/96f4a5b1-db43-46c5-8201-4aa33d49f844 ...
```

```
    ],
    "data": {
        "Product-Name": {
            "name": "Product-Name",
            "title": {
                "lang": "en",
                "value": "Product-Name"
            },
            "type": "ShortTextComponent",
            "data": {
                "type": "text/plain",
                "value": "THE CHAIR"
            },
            "properties": [
                {
                    "key": "HELP_TEXT_PROVIDER_NAME",
                    "value": "NONE"
                },
                {
                    "key": "HIDDEN",
                    "value": "false"
                },
                {
                    "key": "MANDATORY",
                    "value": "false"
                },
                {
                    "key": "MAX_CHARS",
                    "value": "250"
                }
            ]
        },
        "Product-Image-01": {
            "name": "Product-Image-01",
            "title": {
                "lang": "en",
                "value": "Product-Image-01"
            },
            "type": "ImageComponent",
            "data": {
                "type": "application/vnd.ibm.wcm+xml",
                "image": {
                    "dimension": {
                        "height": "5913",
                        "width": "3947",
                        "border": ""
                    },
                    "altText": "",
                    "tagName": "",
                    "fileName": "THE CHAIR.jpg",
                    "resourceUri": {
                        "type": "image/jpeg",
                        "damID": "ml:c035e20-9d91-43e3-aa8e-7801f160275a",
                        "value": "{{dxhost}}/dx/api/dam/v1/collections/ce7a469b-f9b2-42a1-a258-14c382ef7bac/items/
                            ac035e20-9d91-43e3-aa8e-7801f160275a/renditions/9269f6c9-001d-4e6a-af89-5c14624aba7f?
                            binary=true"
                    },
```

16. Then check if your content is updated correctly in the Content Composer. Edit **Woodburn Studio Content -> Content -> Products -> Product 09** and you should see the added **Product-Image-01** image.



Woodburn Studio Content/Products

* Name: Product 09

Display Title: Product 09

Description:

* Select Content Template: Woodburn Studio Design / AT-Product-Item

* Select Site Area: Woodburn Studio Content / Products

Elements

Product-Name (Short Text): THE CHAIR

Product-Image-01 (Image)

THE CHAIR.jpg

Drag and drop is not available

The image will be used for desktop, tablet, and smartphone if there are no renditions uploaded.

Properties

Type: Content

Status: Published

Location: Woodburn Studio Content / Products

UUID: 96f4a5b1-db43-46c5-8201-4aa33d49f844

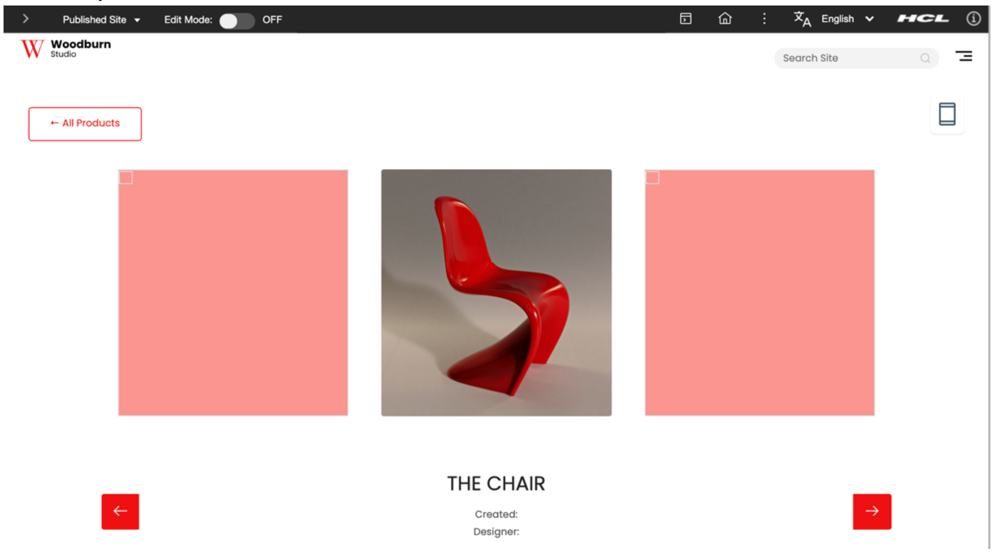
Last Modified: Dec 20, 2021 5:09 PM

Creator: wpsadmin

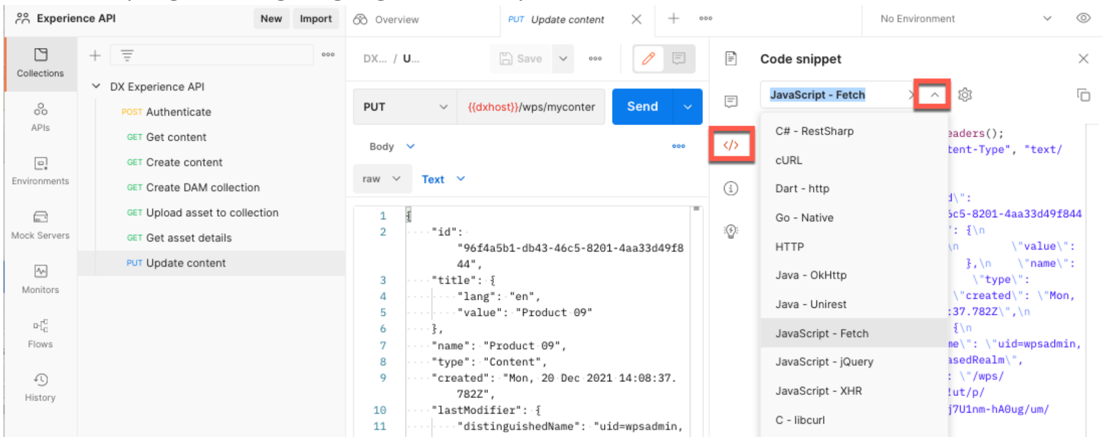
Content Properties

Content Template: Woodburn Studio Design / AT-Product-Item

17. You may also check on the WoodBurn Studio Products page again to see if it shows there correctly.

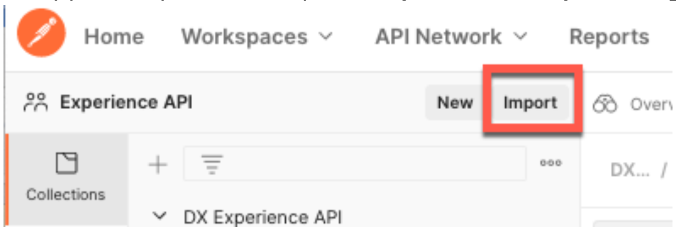


18. Finally, you may use your collection of API call in any code. For that open the Code and select the programming language in which you want to use this.

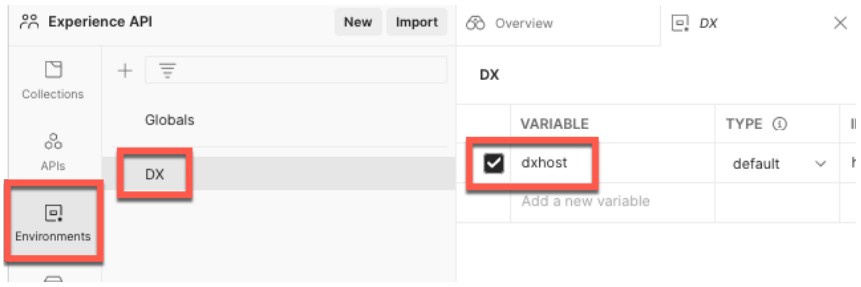


```
{
    "id":
        "96f4a5b1-db43-46c5-8201-4aa33d49f8
        44",
    "title": {
        "lang": "en",
        "value": "Product 09"
    },
    "name": "Product 09",
    "type": "Content",
    "created": "Mon, 20 Dec 2021 14:08:37.
        782Z",
    "lastModifier": {
        "distinguishedName": "uid=wpsadmin,
```

19. Finally, if you have any issues in building this collection, you may import the lab file you unzipped on your Desktop: **DX Experience API.postman_collection.json**.



20. And check the **Environment** variable **DX** – **dxhost** with the correct value of your DX server host and ensure it is enabled.



Congratulations ! You have successfully used the DAM API to create a new collection and upload an image to it. And then use the WCM V2 API again to add this image into an existing content. You also may now use it in any code.

Conclusion

In this lab, you learned to use the Experience API Explorer to discover and try the APIs delivered with HCL Digital Experience. You applied this to two APIs that allowed you to log in and then retrieve a DX content item in JSON format.
You then learned how to use an external application, in this case Postman, to retrieve the same DX content item by calling the DX Experience API methods. You created a new DX content by accessing the DX server using the Postman application and finally used the DAM API to upload a new image and use that in a Web Content, again using the Experience APIs. And you learned how you may easily code this in any programming language.

This may give you a good start to migrate any existing content into HCL Digital Experience and reuse it in any touchpoint.

You may use the Help Center to learn more on the Experience APIs using https://opensource.hcltechsw.com/digital-experience/latest/extend_dx/apis/hcl_experience_api.
This gives more information to all the different APIs, like the GraphQL: https://opensource.hcltechsw.com/digital-experience/latest/extend_dx/apis/hcl_experience_api/openapi_example_API_calls.
It also gives access to a sample application to show developers how the HCL Experience API can be used to build compelling user interfaces using modern technologies. The core UI frameworks being used in the Sample Content UI are React and Redux. You can try the Sample Content UI implementation using: https://opensource.hcltechsw.com/digital-experience/latest/extend_dx/apis/hcl_experience_api/sample_content_ui/

Resources

Refer to the following resources to learn more:

HCL Digital Experience Home - https://hclsw.co/dx

HCL Digital Experience on HCL Solutions Factory (SoFy) - https://hclsofy.com/

HCL Software - https://hclsw.co/software

HCL Product Support - https://hclsw.co/product-support

HCL DX Product Documentation - https://hclsw.co/dx-product-documentation

HCL DX Support Q&A Forum - https://hclsw.co/dx-support-forum

HCL DX Video Playlist on YouTube - https://hclsw.co/dx-video-playlist

HCL DX Product Ideas - https://hclsw.co/dx-ideas

HCL DX Product Demos - https://hclsw.co/dx-product-demo

HCL DX Did You Know? Videos - https://hclsw.co/dx-dyk-videos

HCL DX GitHub - https://hclsw.co/dx-github

HCL DX Web Developer Toolkit - https://github.com/HCL-TECH-SOFTWARE/WebDevToolkitForDx

Legal statements

Disclaimers