Technology Blogs by Members

Explore a vibrant mix of technical expertise, industry insights, and tech buzz in member blogs covering SAP products, technology, and events. Get in the mix!

Blog What are you looking for today?

Using Postman For OData / Netweaver Gateway Testing CRUD Methods

abprad
Active Participant

2020 Mar 27 10:14 AM

29 Kudos 56,163

SAP Managed Tags: SAP Fiori, ABAP Connectivity, ABAP Development, OData, SAPUI5, SAP Gateway, NW ABAP Gateway (OData)

Introduction:

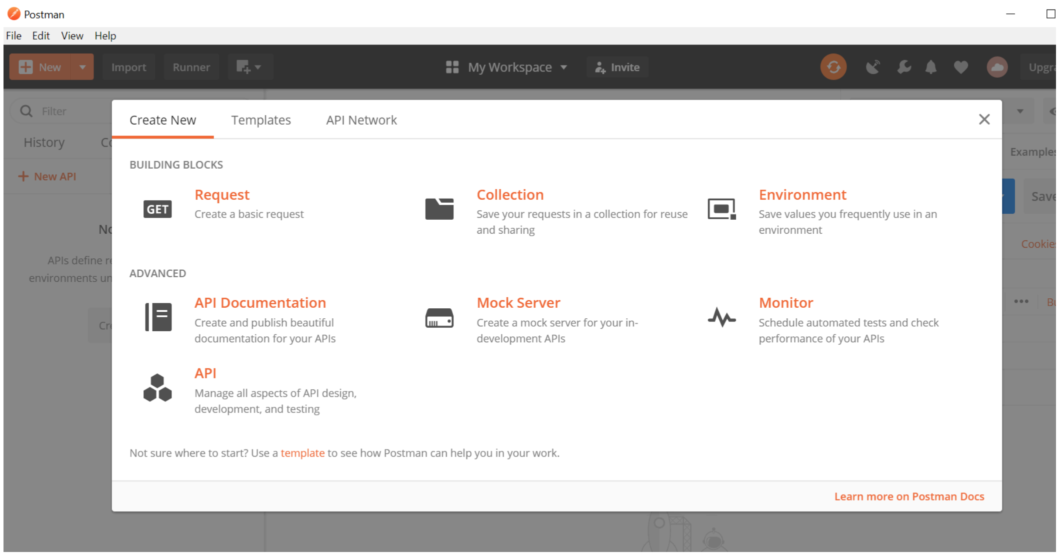
If you have ever used an inbuilt gateway client in SAP for testing your OData services, you must have wondered, isn't there a better & more effective tool to do this testing. Something that gives users an options not to memorize multiple Parameters or group their requests in folders - so that next time when they resume testing, it should start from where they left.

To solve most of these problems, we can make use of a lightweight Open Source Tool called Postman. As per official documentation, "Postman is a tool to design, build, and test APIs".

Enough of the theory , lets dive in to the Tool.

GET Request

- Go to https://www.postman.com/to download and install Postman on your machine.
- Once the Postman is installed, you will get the below screen once you open it



BUILDING BLOCKS

Request
Create a basic request

Collection
Save your requests in a collection for reuse and sharing

Environment
Save values you frequently use in an environment

ADVANCED

API Documentation
Create and publish beautiful documentation for your APIs

Mock Server
Create a mock server for your in-development APIs

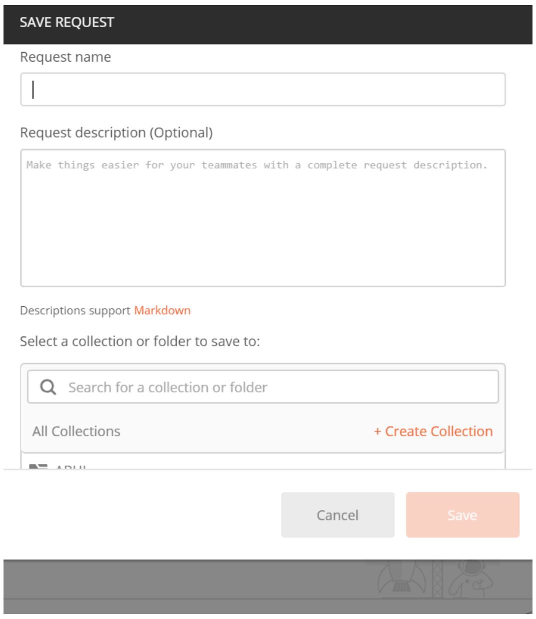
Monitor
Schedule automated tests and check performance of your APIs

API
Manage all aspects of API design, development, and testing

Not sure where to start? Use a template to see how Postman can help you in your work.

Learn more on Postman Docs

- There are multiple options as you can see in the screenshot, but let us only concentrate on the "GET" icon - Request. Click on “GET”.
- Once you do that, you get another popup (gosh again!)
- It prompts you to enter a request name and create a collection. (Screen shot below)
- **Collection** - Collection refers to a group of request (GET, POST, and PUT etc. etc.)
- Collections are particularly useful when you are testing say for different OData Services at a time, segregating them into different collections can help you test in a much efficient / cleaner way.


SAVE REQUEST

Request name

Request description (Optional)

Make things easier for your teammates with a complete request description.

Descriptions support Markdown

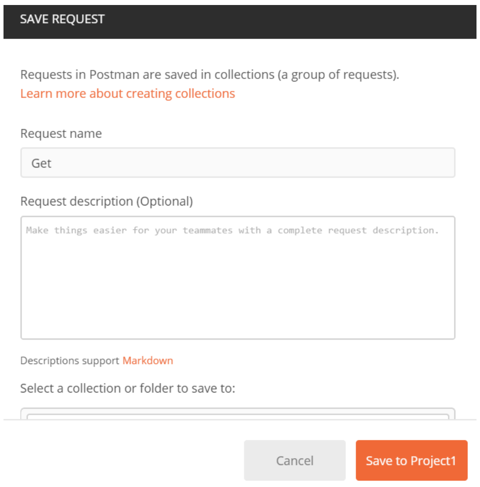
Select a collection or folder to save to:

Search for a collection or folder

All Collections + Create Collection

Cancel Save

- Create a **collection** and Name a Request, Let's say for example, Get is my Request name and Project1 is my collection. Press "Save" to the save the request into collection.


SAVE REQUEST

Requests in Postman are saved in collections (a group of requests).
Learn more about creating collections

Request name

Get

Request description (Optional)

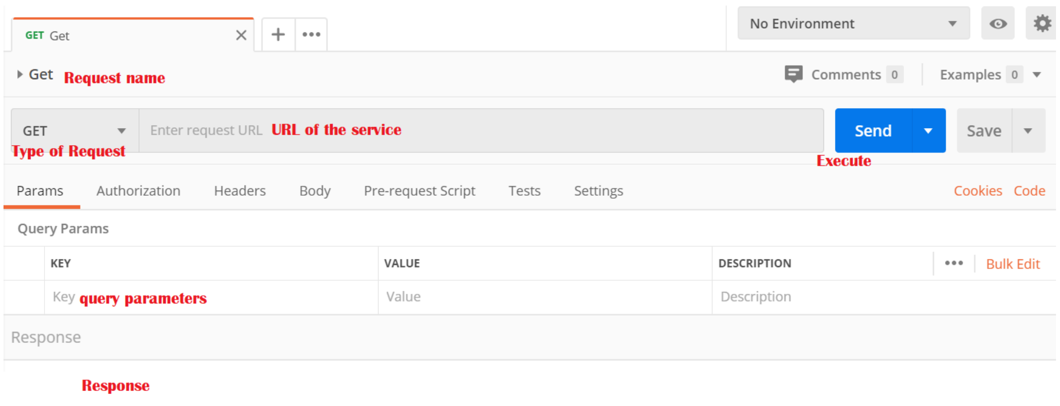
Make things easier for your teammates with a complete request description.

Descriptions support Markdown

Select a collection or folder to save to:

Cancel Save to Project1

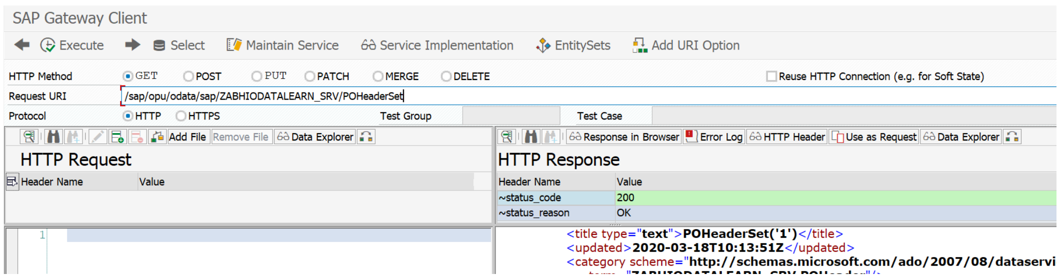
- You will get the below screen (see my comments in red, describing the screen)



- As our request name is "Get", let us start with a GET request.

- In SAP, Gateway Client - (/IWFND/GW_CLIENT), we get the below screen with the URL if we want to test a service.

- URL- /sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet



SAP Gateway Client

HTTP Method: [✓ GET] POST PUT PATCH MERGE DELETE — Reuse HTTP Connection (e.g. for Soft State)

Request URI: /sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet

Protocol: [✓ HTTP] HTTPS — Test Group — Test Case

HTTP Request

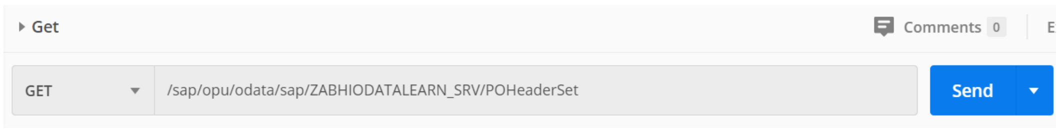
Header Name	Value

HTTP Response

Header Name	Value
~status_code	200
~status_reason	OK

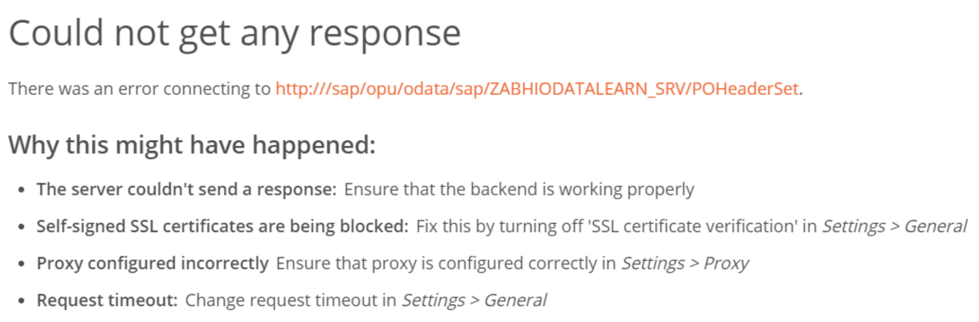
```
<title type="text">POHeaderSet('1')</title>
<updated>2020-03-18T10:13:51Z</updated>
<category scheme="http://schemas.microsoft.com/ado/2007/08/dataservi
```

- Let us use this URL in Postman and see what happens. Whoa! There is no response.



Get

GET /sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet



Could not get any response

There was an error connecting to http:///sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet.

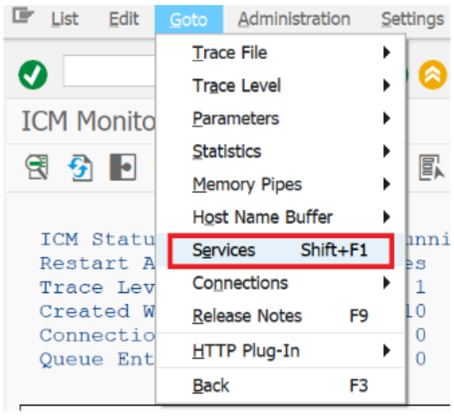
Why this might have happened:

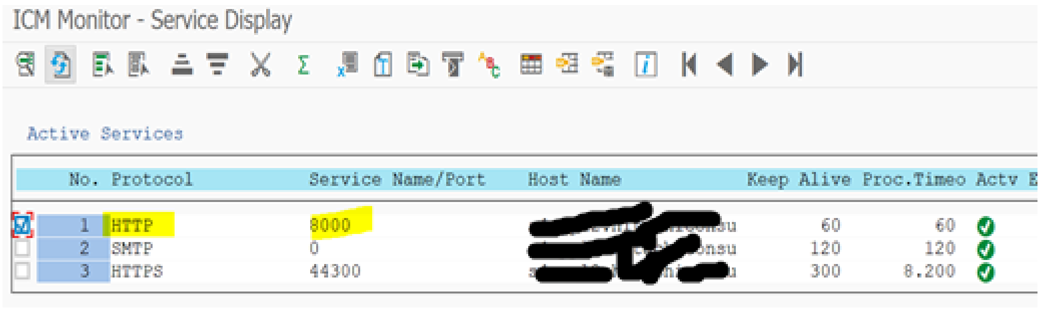
- **The server couldn't send a response:** Ensure that the backend is working properly
- **Self-signed SSL certificates are being blocked:** Fix this by turning off 'SSL certificate verification' in *Settings > General*
- **Proxy configured incorrectly** Ensure that proxy is configured correctly in *Settings > Proxy*
- **Request timeout:** Change request timeout in *Settings > General*

- Why? It is because Postman doesn't know which HTTP address this service lies at. Each service has to be in a server. When we use Gateway Client on SAP, we are using SAP's server by default. Therefore, we need to tell Postman the exact address where our service resides. (Just like you write your entire address on a letter, with the Pin Code).

- Now if you are not aware of the HTTP address of your server, follow the below steps to get it.

- Open T-Code - SMICM - > in the menu Go To-> Service. Note the HTTP Port Name and Host name. Ping the host name in Command Prompt to get the IP address. Or Just open your SAP logon pad and note the application server details for the system you are using.

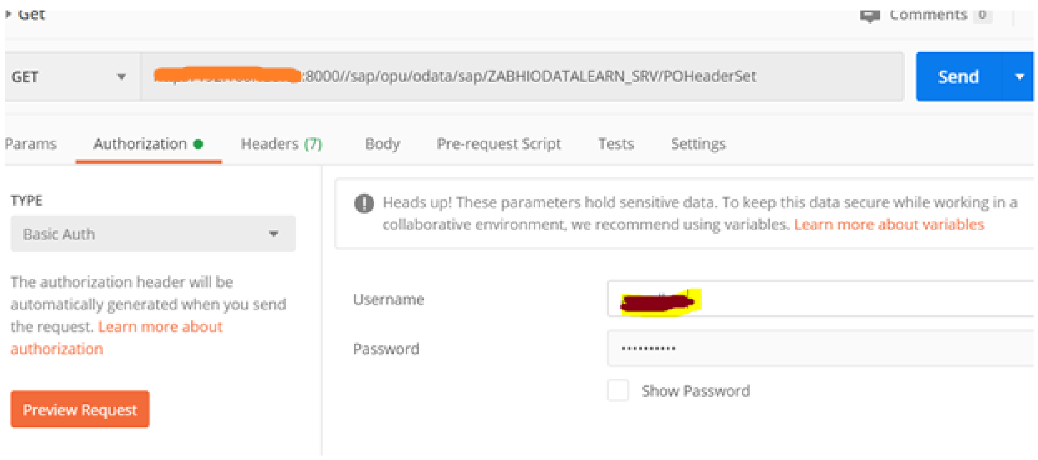



ICM Monitor - Service Display

Active Services

No.	Protocol	Service Name/Port	Host Name	Keep Alive	Proc.Timeo	Actv E
1	HTTP	8000	[redacted]	60	60	✓
2	SMTP	0	[redacted]	120	120	✓
3	HTTPS	44300	[redacted]	300	8.200	✓

- So , now our URL should be like below
- HTTP://applicationserver:portname/sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet
- In this case its
- http://XXX.XXX.139.62:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet
- Let's use this URL now to query in Postman. However, before that we need to provide our authentication details in the query so that the server allows us to query the service.
- To do so, click on the Authorization Tab, Choose "Basic Auth" from Type and enter your username and password for the SAP system you are connecting. Give the URL that we generated just now and Press Send.


GET [redacted]:8000//sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet

Authorization

TYPE: Basic Auth

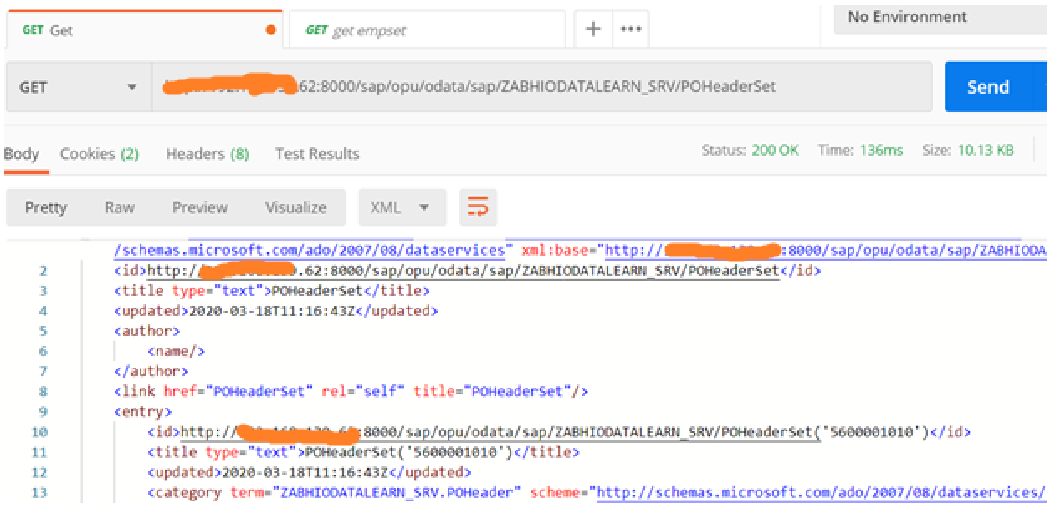
The authorization header will be automatically generated when you send the request. Learn more about authorization

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables

Username: [redacted]

Password: ••••••••••

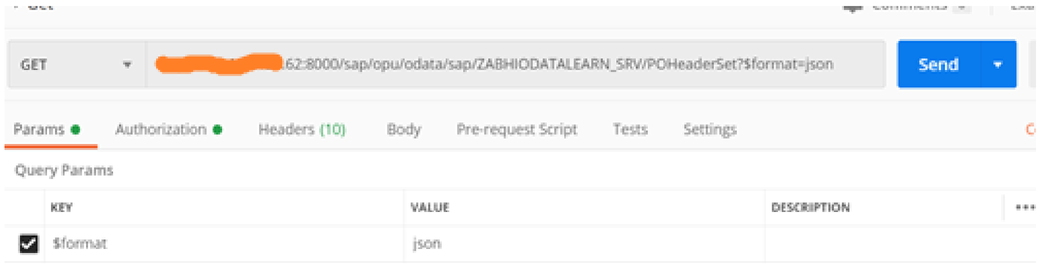
- Hurray! We now see data in the response in XML format!!


GET [redacted]62:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet

Status: 200 OK Time: 136ms Size: 10.13 KB

```
/schemas.microsoft.com/ado/2007/08/dataservices" xml:base="http://[redacted]:8000/sap/opu/odata/sap/ZABHIODA
<id>http:/[redacted].62:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet</id>
<title type="text">POHeaderSet</title>
<updated>2020-03-18T11:16:43Z</updated>
<author>
    <name/>
</author>
<link href="POHeaderSet" rel="self" title="POHeaderSet"/>
<entry>
    <id>http://[redacted]:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet('5600001010')</id>
    <title type="text">POHeaderSet('5600001010')</title>
    <updated>2020-03-18T11:16:43Z</updated>
    <category term="ZABHIODATALEARN_SRV.POHeader" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/
```

- I don't like seeing XML, its looks rather clumsy!
- Let's add some parameters to get data in JSON format.
- Go to the "Params" tab and add a $format = json filter as shown in below screenshot.

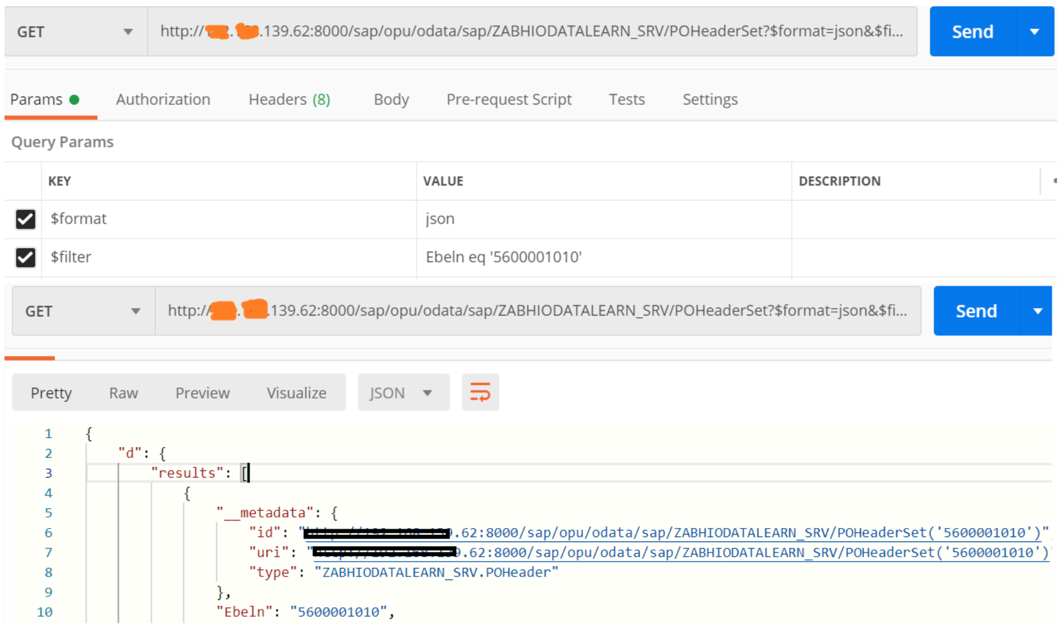

GET [redacted]62:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet?$format=json

Query Params

	KEY	VALUE	DESCRIPTION
✓	$format	json	

- The Params tab allows adding different parameters without having to worry about concatenating them properly.
- Let's add another parameter, $Filter. If you see closely, once you add the parameters in the Params tab, they are auto added into the URL.

- Let's Press Send and see results. We have the data in JSON format and the filter works too.



GET http://[redacted].[redacted].139.62:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet?$format=json&$fi... Send

Params ● Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIPTION
✓	$format	json	
✓	$filter	Ebeln eq '5600001010'	

GET http://[redacted].[redacted].139.62:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet?$format=json&$fi... Send

Pretty Raw Preview Visualize JSON

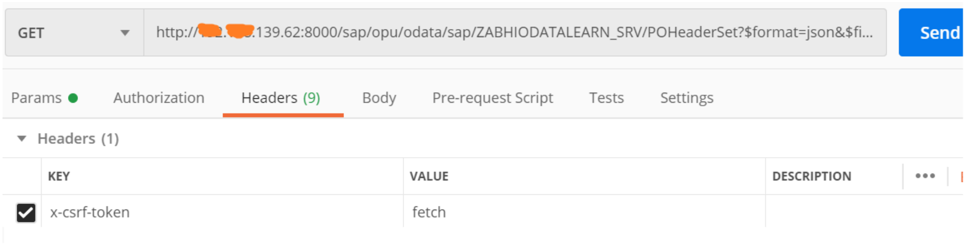
```
{
    "d": {
        "results": [
            {
                "__metadata": {
                    "id": "[redacted].62:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet('5600001010')"
                    "uri": "[redacted].62:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet('5600001010')
                    "type": "ZABHIODATALEARN_SRV.POHeader"
                },
                "Ebeln": "5600001010",
```

- Let us check the final part for GET requests.

  - When using GET we can fetch the X-CSRF-TOKEN to use for POST and PUT statements from POSTMAN.

  - X-CSRF-TOKEN is an identifier SAP sends for Cross Site Forgery Protection. In simple terms, it is a token to say that you are allowed to update into SAP.

  - Go to the headers tab in GET request and add a header X-CSRF-TOKEN and value as fetch.



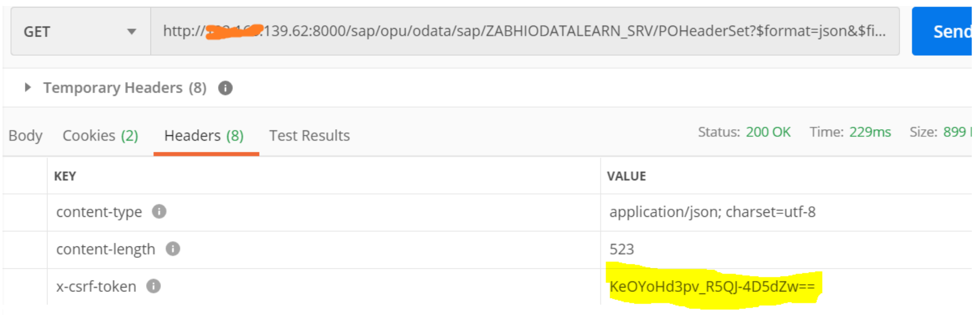
GET http://[redacted].[redacted].139.62:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet?$format=json&$fi... Send

Params ● Authorization Headers (9) Body Pre-request Script Tests Settings

Headers (1)

	KEY	VALUE	DESCRIPTION
✓	x-csrf-token	fetch	

  - Press Send, the response Header now contains an X-CSRF-TOKEN sent by SAP that is valid for generally 24 hours and is unique for your id.



GET http://[redacted]139.62:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/POHeaderSet?$format=json&$fi... Send

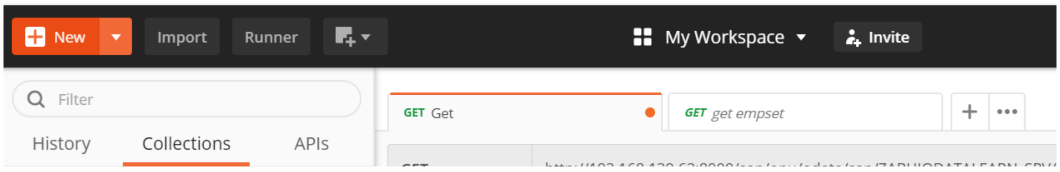
Temporary Headers (8)

Body Cookies (2) Headers (8) Test Results Status: 200 OK Time: 229ms Size: 899

KEY	VALUE
content-type	application/json; charset=utf-8
content-length	523
x-csrf-token	KeOYoHd3pv_R5QJ-4D5dZw==

POST Request

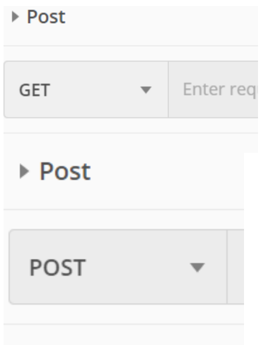
- Now, we know how the GET request works in POSTMAN. Let us see how to use PUT and POST in Postman.



- Create a new request, the same popup comes to enter request name and add that into a collection.

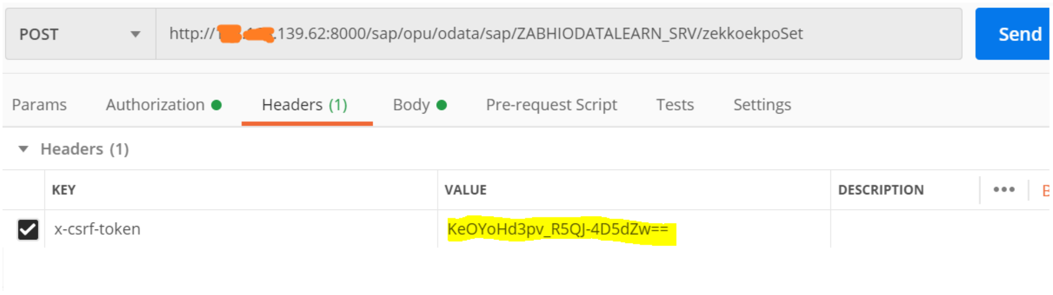
- Name this request as **Post** and add it into the collection created from previous step.

- Change the type of this Request to **POST** from the dropdown.



- Go to **Authorization** tab , select **Basic Auth** and enter your username and password for the SAP System

- We have fetched the **X-CSRF-TOKEN** in the **GET** request; add this fetched token into the header as shown below.

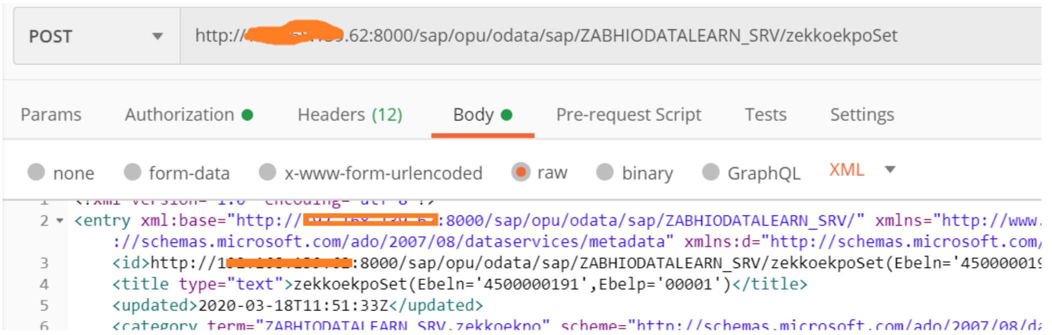

POST http://[redacted]139.62:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/zekkoekpoSet Send

Params | Authorization ● | Headers (1) | Body ● | Pre-request Script | Tests | Settings

Headers (1)

	KEY	VALUE	DESCRIPTION
✓	x-csrf-token	KeOYoHd3pv_R5QJ-4D5dZw==	

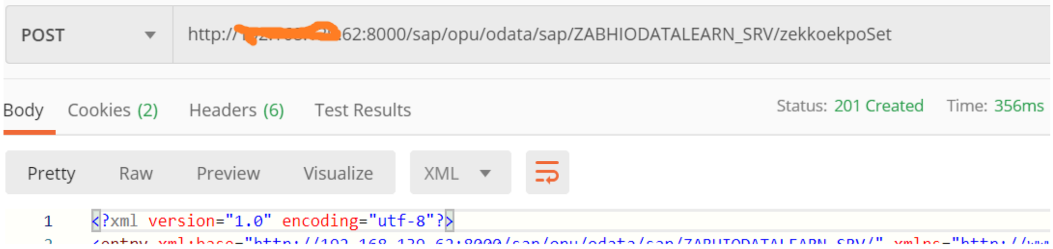
- Give the URL for post (we learnt how to create this if we don't know the server details)
- The request details for Post are sent in the body of the request. Go to the Body tab and copy the response from GET request into the body of the POST request.
- Select "raw" button and paste the xml. Edit your values and Press “Send” to see what happens.
- The Item is created in the table and you get a 201-status code. Go ahead and verify if the data is inserted in your table.


POST http://[redacted].62:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/zekkoekpoSet

Params | Authorization ● | Headers (12) | Body ● | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | [● raw] | binary | GraphQL | XML

```
<?xml version="1.0" encoding="utf-8"?>
<entry xml:base="http://[redacted]:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/" xmlns="http://www.
    ://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns:d="http://schemas.microsoft.com/
    <id>http://1[redacted]:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/zekkoekpoSet(Ebeln='45000001
    <title type="text">zekkoekpoSet(Ebeln='4500000191',Ebelp='00001')</title>
    <updated>2020-03-18T11:51:33Z</updated>
    <category term="ZABHIODATALEARN_SRV.zekkoekpo" scheme="http://schemas.microsoft.com/ado/2007/08/d
```


POST http://[redacted]62:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/zekkoekpoSet

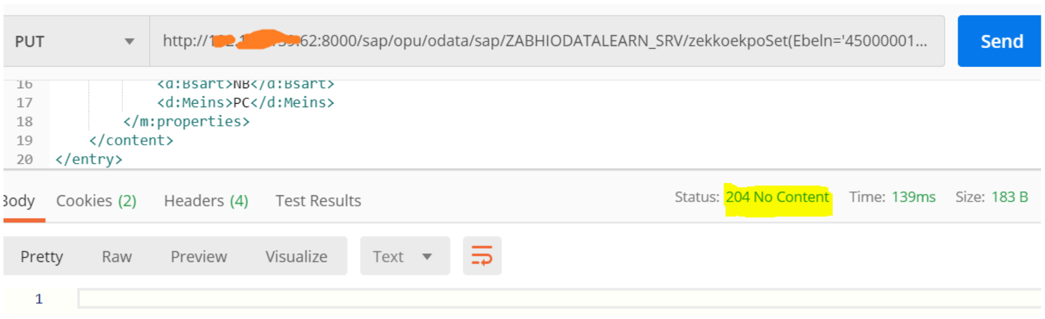
Body | Cookies (2) | Headers (6) | Test Results   Status: 201 Created   Time: 356ms

Pretty | Raw | Preview | Visualize | XML

```
<?xml version="1.0" encoding="utf-8"?>
<entry xml:base="http://192.168.139.62:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/" xmlns="http://www
```

PUT Request

- The process for **PUT** request remains the same, just change the **POST** request to **PUT** when creating a new request.
- Provide the URL for **PUT** with the key to update and press send.
- A status code 204 indicates update was performed. Verify your details in the table.



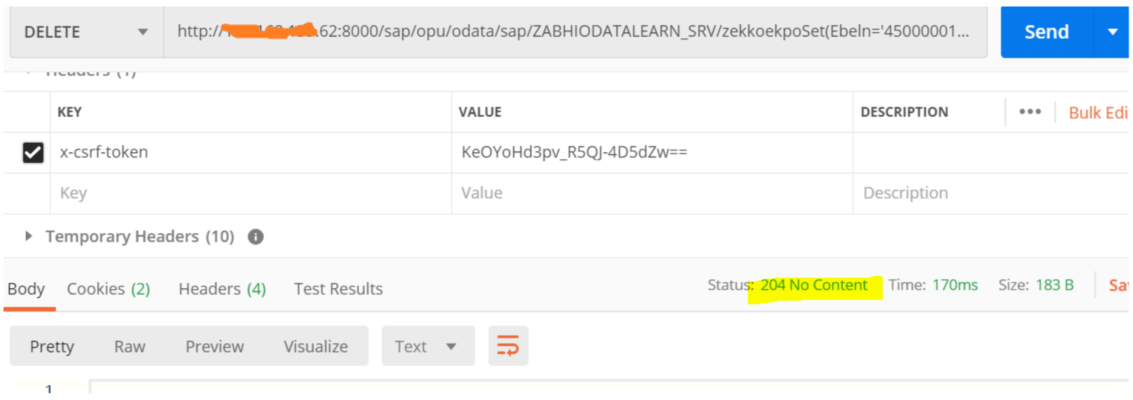
PUT http://[redacted]62:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/zekkoekpoSet(Ebeln='45000001...

```
        <d:Bsart>NB</d:Bsart>
        <d:Meins>PC</d:Meins>
      </m:properties>
    </content>
</entry>
```

Status: 204 No Content Time: 139ms Size: 183 B

DELETE Request

- Create a new request, same steps as before.
- Change the type to **DELETE**, input the URL, and add the authorization details.
- Add the X-CSRF-TOKEN fetched from a GET request.
- Press Send. A **204 HTTP** Status code indicates that delete was successful.
- Verify the details in the system.



DELETE http://[redacted]62:8000/sap/opu/odata/sap/ZABHIODATALEARN_SRV/zekkoekpoSet(Ebeln='45000001...

	KEY	VALUE	DESCRIPTION
✓	x-csrf-token	KeOYoHd3pv_R5QJ-4D5dZw==	
	Key	Value	Description

Temporary Headers (10)

Status: 204 No Content Time: 170ms Size: 183 B

Conclusion

- The document shows testing all the four CRUD methods through Postman Tool that is a more user-friendly way of testing OData services.
- Again, Postman is a highly efficient tool that allows for collaboration with multiple team members. You can research more on this based on your need.
- Hope this document was useful to you!

Links:

Download Postman

1. https://www.postman.com/
2. https://www.postman.com/product/api-client

X-CSRF-TOKEN

1. https://help.sap.com/saphelp_gateway20sp12/helpdata/en/e6/cae27d5e8d4996add4067280c8714e/frameset.ht...

Tags:

gateway OData TesingtOdataService