Lab – RESTCONF with Postman

Objectives

Part 1: Setup HTTP Headers in Postman

Background / Scenario

RESTCONF is a RESTful API interface that provides a programmatic interface for accessing data defined in YANG device models.

In this lab, you will learn how to interact with the RESTCONF interface using the Postman application to retrieve the device's configuration, update and create new interfaces and retrieve operation data.

Required Resources

- Postman
- Access to a router with the IOS XE operating system version 16.6 or higher.

Instructions

Part 1: Setup HTTP headers and Authentication in Postman

In this part, you will setup the HTTP Headers in Postman to work with RESTCONF REST API interfaces.

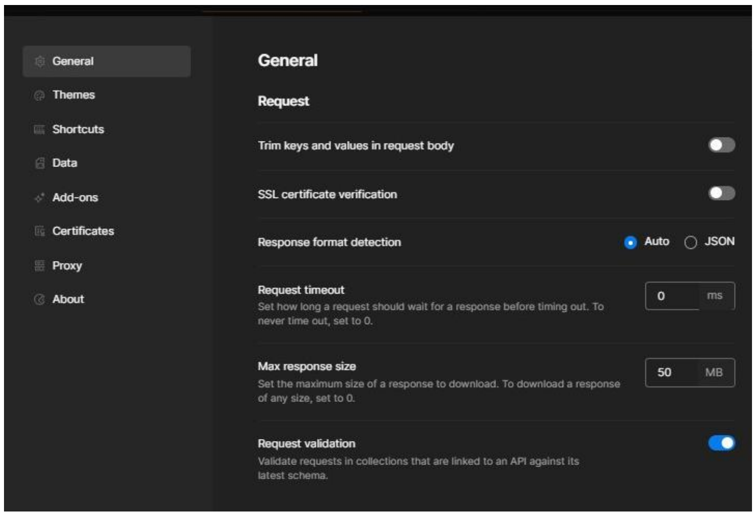
Step 1: Configure Postman.

Configure a Postman setting.

a. Open Postman and create a new tab if necessary.

b. Click **File > Settings**.

c. Under the **General** tab, set the **SSL certificate verification** to **OFF**. Close the **Settings** dialog box.



General

Request

Trim keys and values in request body

SSL certificate verification

Response format detection: Auto / JSON

Request timeout: 0 ms
Set how long a request should wait for a response before timing out. To never time out, set to 0.

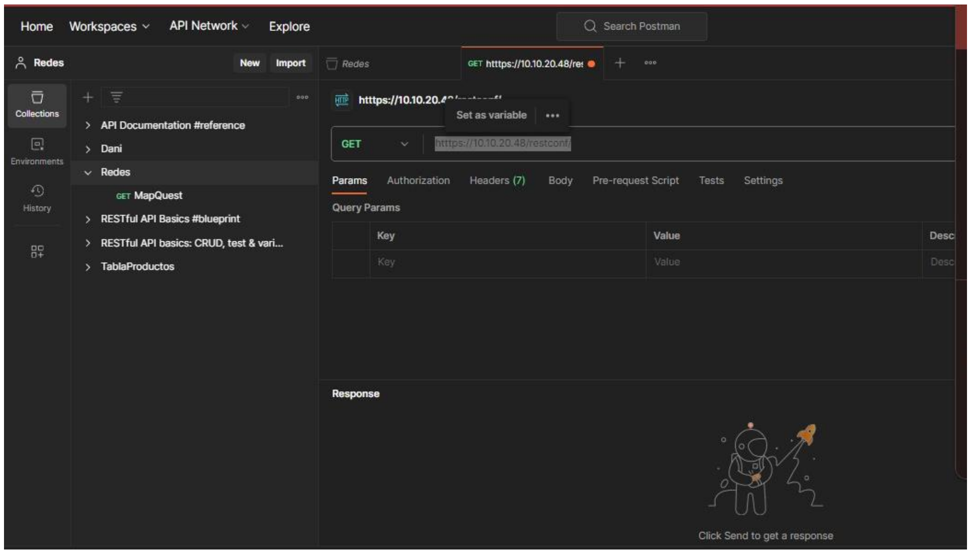
Max response size: 50 MB
Set the maximum size of a response to download. To download a response of any size, set to 0.

Request validation
Validate requests in collections that are linked to an API against its latest schema.

Step 2: Select the method and enter the required URL.

a. Next to the URL box, select the request method to be **GET**.

b. Enter the URL for API endpoint: https://192.168.56.101/restconf/ (adjust the IP address to match the router's current address)

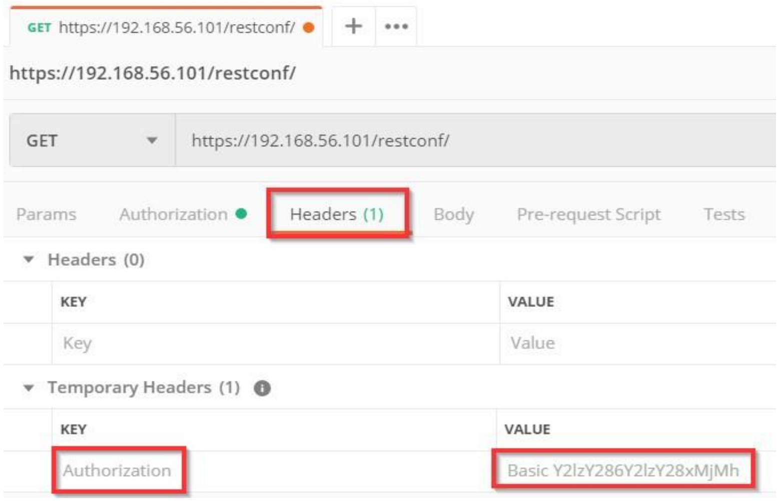


Step 3: Enter the authentication header information.

a. Below the URL input field, select **Authorization**.

b. Select the type **Basic Auth** and enter the username and password for authenticating to the RESTCONF API service: `cisco cisco123!`

c. Click on the **Preview Request** to add the Authentication header to the Headers.



GET https://192.168.56.101/restconf/

https://192.168.56.101/restconf/

GET https://192.168.56.101/restconf/

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

Headers (0)

KEY	VALUE
Key	Value

Temporary Headers (1)

KEY	VALUE
Authorization	Basic Y2lzY286Y2lzY28xMjMh

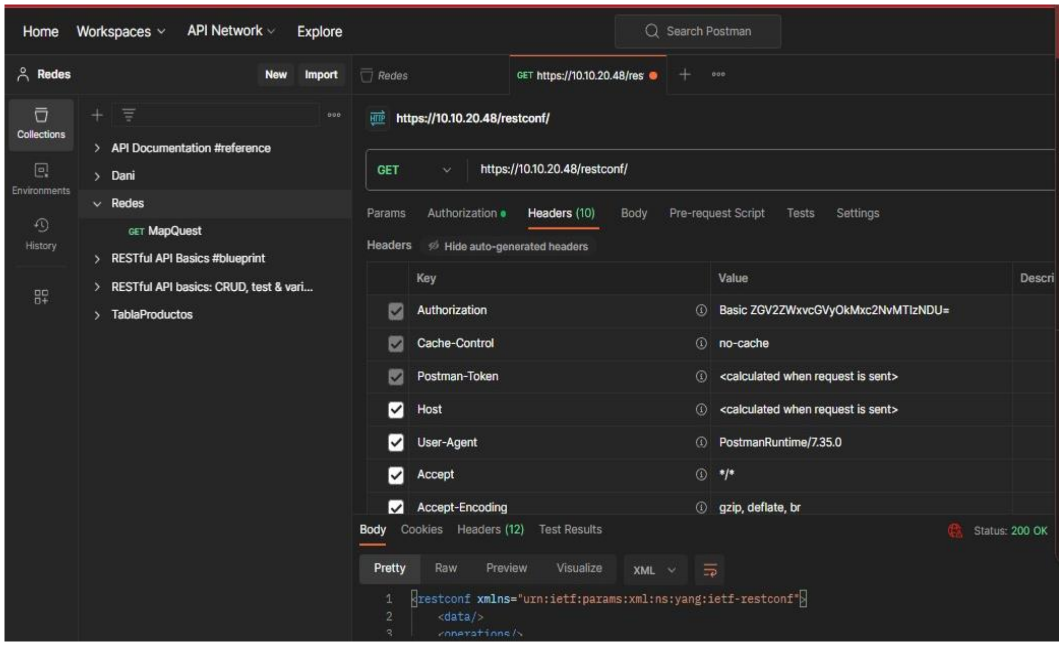
Step 4: Enter the additional Headers.

a. Next, below the URL input field, select **Headers**.

b. Under **Headers**, click the **New key** field under the **Key** column and enter **Content-Type** to define what is the encoding of data that are sent with Postman in the REST API request.

c. Next again under **Headers**, click the **New key** field under the **Key** column and enter **Accept** to define in what encoding you wish to receive back the data replies from the REST API request.

d. In the value columns, enter `application/yang-data+json` when sending or receiving data that are in JSON encoding. When sending or receiving data in XML encoding (default), use `application/yangdata+xml`.

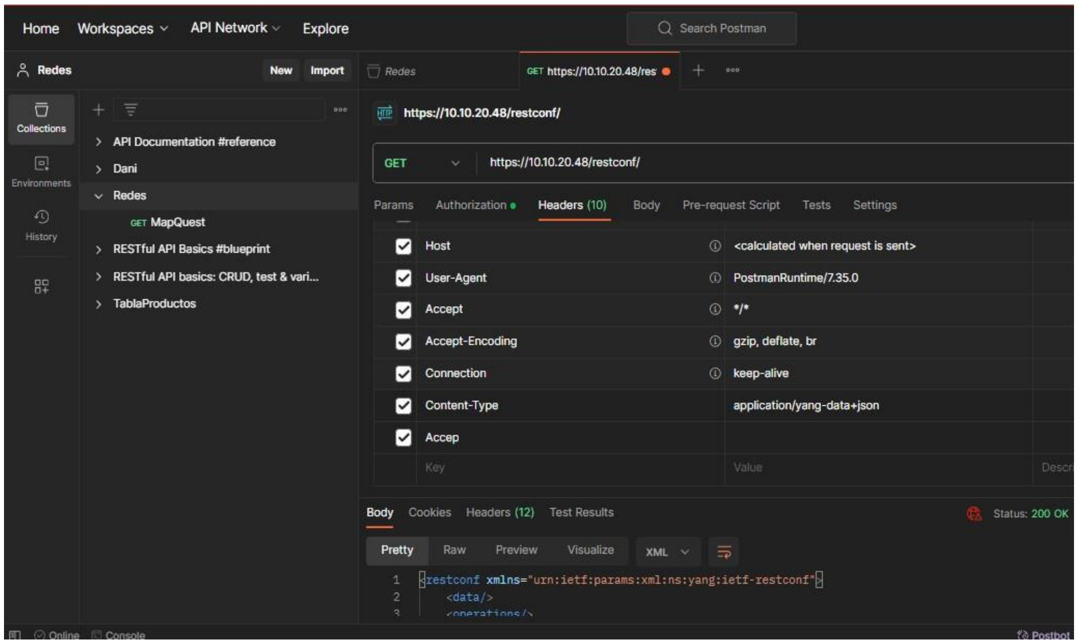


https://10.10.20.48/restconf/

GET https://10.10.20.48/restconf/

	Key	Value	Descri
✓	Authorization	Basic ZGV2ZWxvcGVyOkMxc2NvMTIzNDU=	
✓	Cache-Control	no-cache	
✓	Postman-Token	<calculated when request is sent>	
✓	Host	<calculated when request is sent>	
✓	User-Agent	PostmanRuntime/7.35.0	
✓	Accept	*/*	
✓	Accept-Encoding	gzip, deflate, br	

```
<restconf xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
    <data/>
    <operations/>
```



https://10.10.20.48/restconf/

GET https://10.10.20.48/restconf/

	Key	Value	Descri
✓	Host	<calculated when request is sent>	
✓	User-Agent	PostmanRuntime/7.35.0	
✓	Accept	*/*	
✓	Accept-Encoding	gzip, deflate, br	
✓	Connection	keep-alive	
✓	Content-Type	application/yang-data+json	
✓	Accep		
	Key	Value	Descri

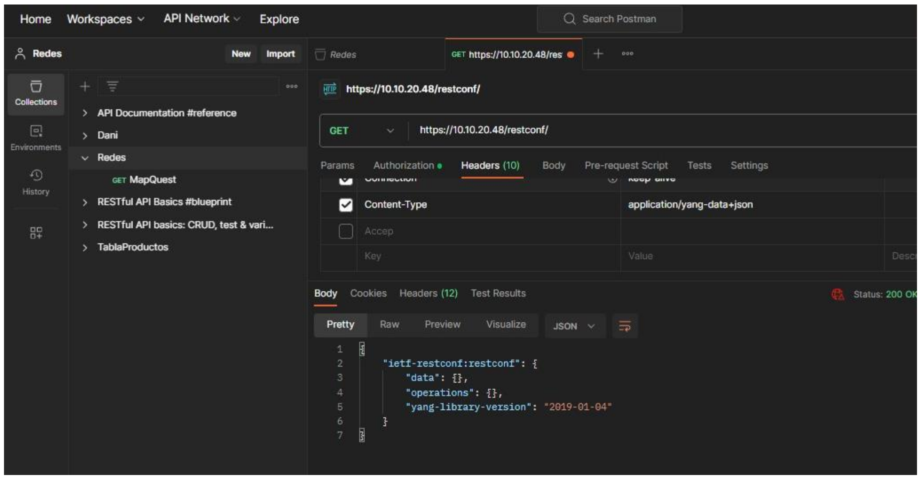
```
<restconf xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
    <data/>
    <operations/>
```

Step 5: Send the request.

a. Click **Send**.

b. After a brief delay, you should see the response appear below the request information. Scroll down, if necessary to view the response data.

c. When the **Accept** header was set to `application/yang-data+json` the response consists of JSON encoded data.



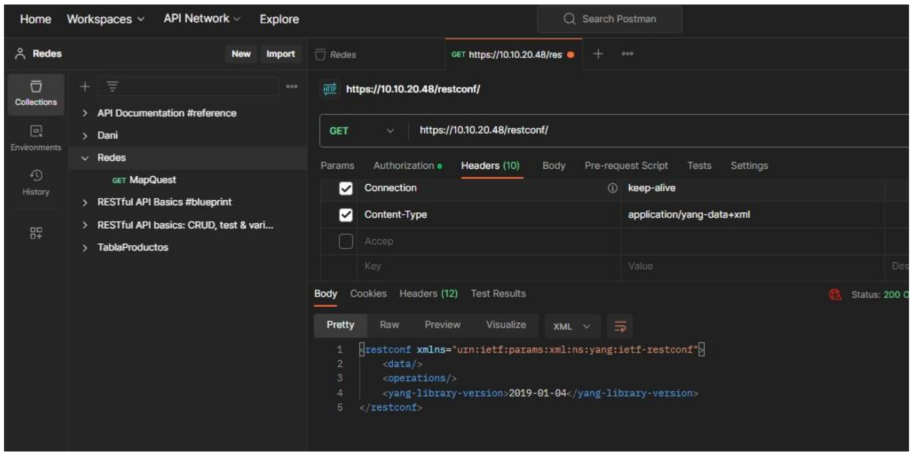
GET https://10.10.20.48/restconf/

	Key	Value
✓	Content-Type	application/yang-data+json
	Accep	

Status: 200 OK

```
{
    "ietf-restconf:restconf": {
        "data": {},
        "operations": {},
        "yang-library-version": "2019-01-04"
    }
}
```

d. Uncheck the **Accept** header, or change it to `application/yang-data+xml` (default behavior when no Accept header is defined) and click on **Send**. Now the response consists of XML encoded data:



GET https://10.10.20.48/restconf/

	Key	Value
✓	Connection	keep-alive
✓	Content-Type	application/yang-data+xml
	Accep	

```
<restconf xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
    <data/>
    <operations/>
    <yang-library-version>2019-01-04</yang-library-version>
</restconf>
```

Note: If there is an error, check the Status of the request. Check the meaning of the status code. 200 means success. A 404 error may mean that the URL was entered incorrectly. A 401 or 403 error could indicate a problem with the authentication, so check that the credentials are entered correctly.

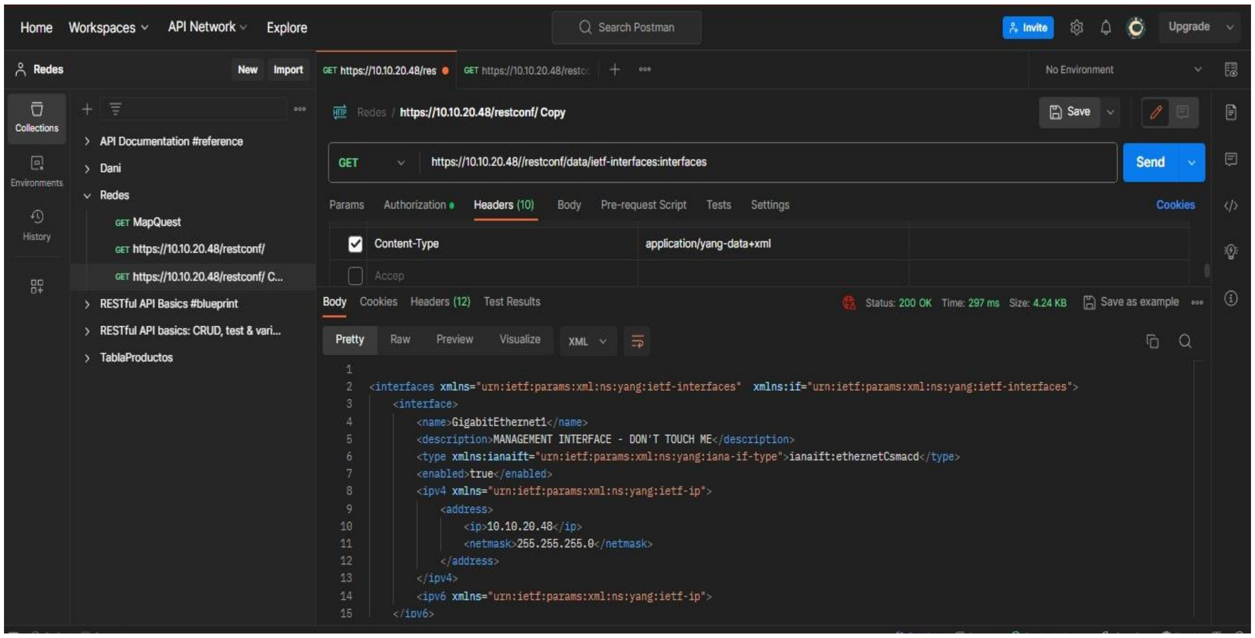
Part 2: Retrieve Interface Configuration

Step 1: Retrieve YANG data in XML.

a. Duplicate the existing REST API request in Postman by right clicking on the request and selecting Duplicate Tab to create a new request with prefilled existing Headers:

b. In the URL for the API endpoint, enter after the host and port portion: `/restconf/data/ietf-interfaces:interfaces` to retrieve data defined in the **ietf-interfaces** YANG model in the **interfaces** container.

c. Click **Send** to send the RESTCONF API request. With unchecked **Accept** header or the **Accept** header set to `application/yang-data+xml`, the result YANG data are returned in XML encoding.



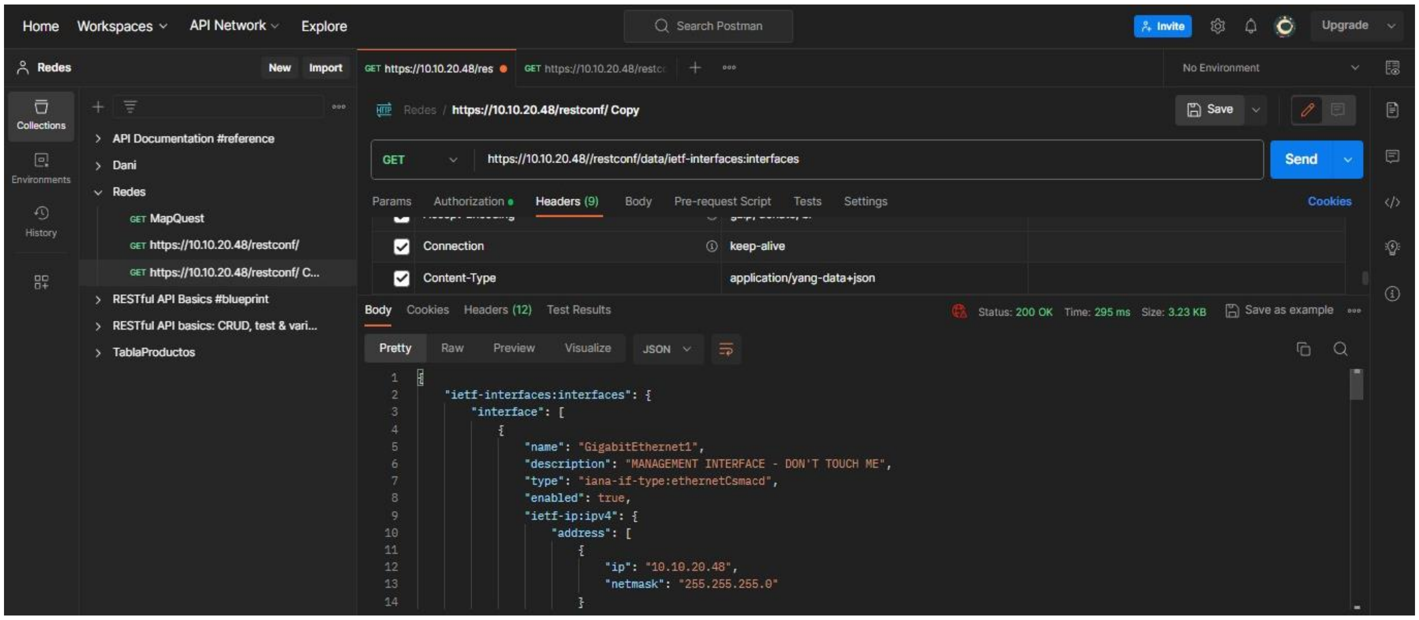
GET https://10.10.20.48//restconf/data/ietf-interfaces:interfaces

✓ Content-Type	application/yang-data+xml
Accep	

Status: 200 OK Time: 297 ms Size: 4.24 KB

```
<interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"  xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">
    <interface>
        <name>GigabitEthernet1</name>
        <description>MANAGEMENT INTERFACE - DON'T TOUCH ME</description>
        <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
            <address>
                <ip>10.10.20.48</ip>
                <netmask>255.255.255.0</netmask>
            </address>
        </ipv4>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
    </ipv6>
```

d. Check the Accept header and set it to `application/yang-data+json` to receive YANG data formatted in JSON encoding.



GET https://10.10.20.48//restconf/data/ietf-interfaces:interfaces

Connection	keep-alive
Content-Type	application/yang-data+json

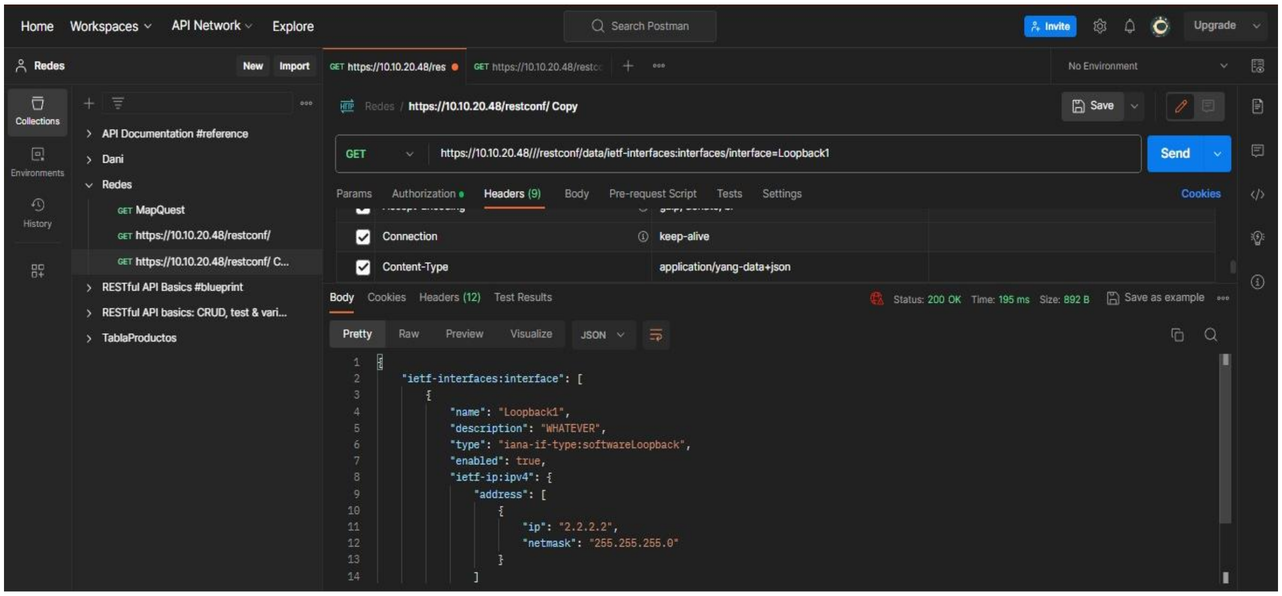
Status: 200 OK Time: 295 ms Size: 3.23 KB

```
{
    "ietf-interfaces:interfaces": {
        "interface": [
            {
                "name": "GigabitEthernet1",
                "description": "MANAGEMENT INTERFACE - DON'T TOUCH ME",
                "type": "iana-if-type:ethernetCsmacd",
                "enabled": true,
                "ietf-ip:ipv4": {
                    "address": [
                        {
                            "ip": "10.10.20.48",
                            "netmask": "255.255.255.0"
                        }
```

Step 2: Retrieve information about a single interface

a. In the URL for the API endpoint, enter after the host and port portion: `/restconf/data/ietf-interfaces:interfaces/interface=Loopback1` to retrieve data for the **Loopback1** interface only.

b. Click **Send** …



GET https://10.10.20.48///restconf/data/ietf-interfaces:interfaces/interface=Loopback1

Connection	keep-alive
Content-Type	application/yang-data+json

Status: 200 OK Time: 195 ms Size: 892 B

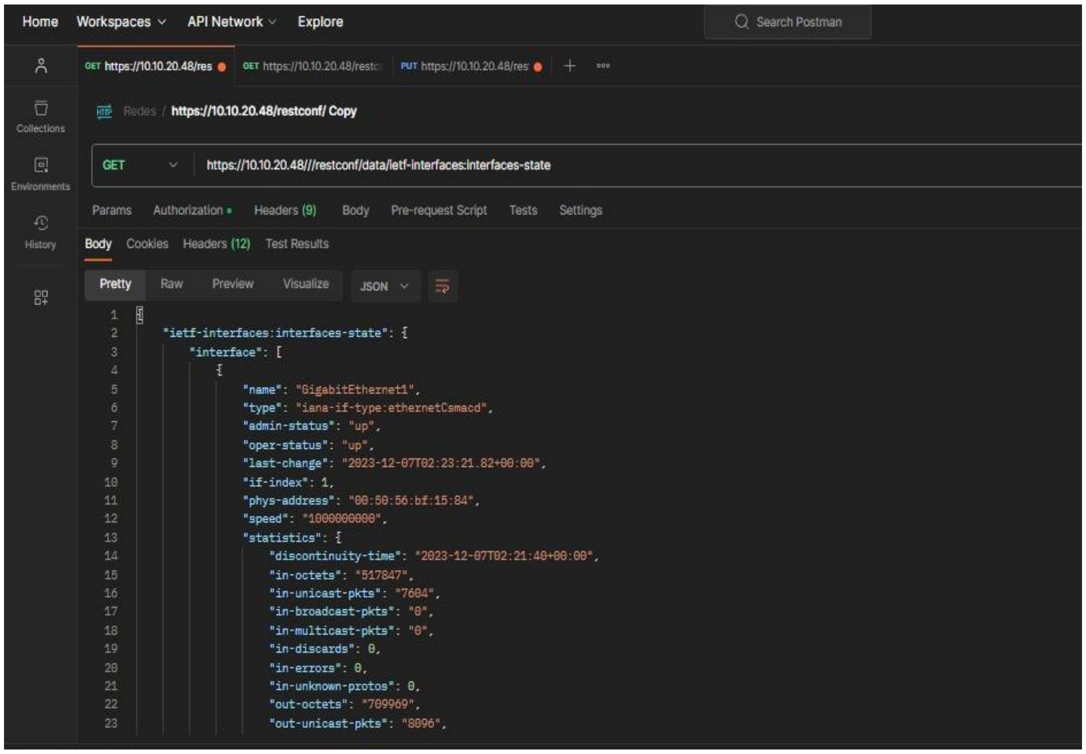
```
{
    "ietf-interfaces:interface": [
        {
            "name": "Loopback1",
            "description": "WHATEVER",
            "type": "iana-if-type:softwareLoopback",
            "enabled": true,
            "ietf-ip:ipv4": {
                "address": [
                    {
                        "ip": "2.2.2.2",
                        "netmask": "255.255.255.0"
                    }
                ]
```

Step 3: Retrieve operation data

a. Change the RESTCONF API Endpoint to retrieve operation and statistical data about the interfaces. You can use the “pyang” tool to discover what container in the **ietf-interfaces** YANG model includes the operational and stats data.

b. What is the new URL?

https://10.10.20.48/restconf/data/ietf-interfaces:interfaces-state



GET https://10.10.20.48///restconf/data/ietf-interfaces:interfaces-state

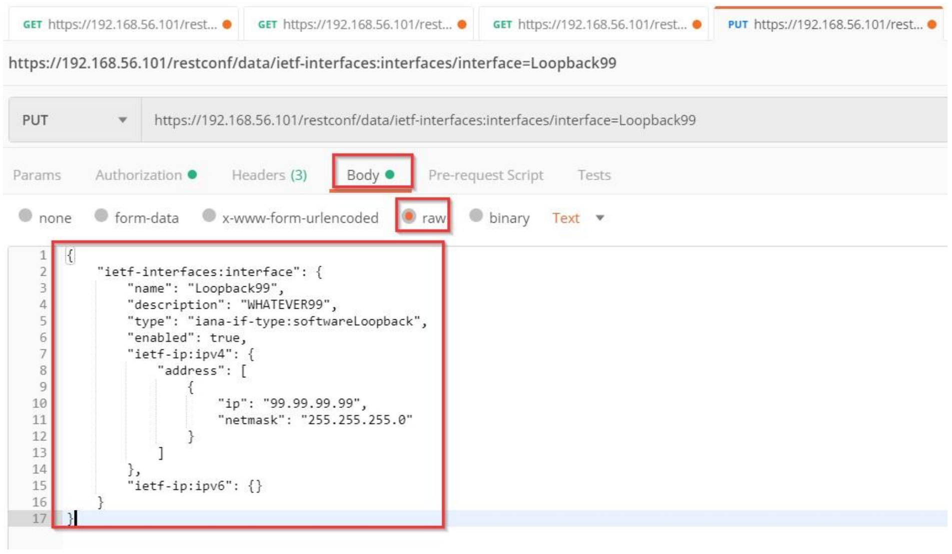
```
{
    "ietf-interfaces:interfaces-state": {
        "interface": [
            {
                "name": "GigabitEthernet1",
                "type": "iana-if-type:ethernetCsmacd",
                "admin-status": "up",
                "oper-status": "up",
                "last-change": "2023-12-07T02:23:21.82+00:00",
                "if-index": 1,
                "phys-address": "00:50:56:bf:15:84",
                "speed": "1000000000",
                "statistics": {
                    "discontinuity-time": "2023-12-07T02:21:40+00:00",
                    "in-octets": "517847",
                    "in-unicast-pkts": "7604",
                    "in-broadcast-pkts": "0",
                    "in-multicast-pkts": "0",
                    "in-discards": 0,
                    "in-errors": 0,
                    "in-unknown-protos": 0,
                    "out-octets": "709969",
                    "out-unicast-pkts": "8096",
```

Part 3: Update Interface Configuration

a. Duplicate the existing REST API request in Postman by right clicking on the request and selecting Duplicate Tab to create a new request with prefilled existing Headers.

b. In the URL for the API endpoint, change the method from **GET** to **PUT** and enter after the host and port portion: `/restconf/data/ietf-interfaces:interfaces/interface=Loopback99` to create a new interface **Loopback99**.

c. In the Body part of the request, enter the YANG data you want to apply to the device's configuration (copy, paste and customize the JSON data from the previous GET request on Loopback1 – change the interface name and IP address).



https://192.168.56.101/restconf/data/ietf-interfaces:interfaces/interface=Loopback99

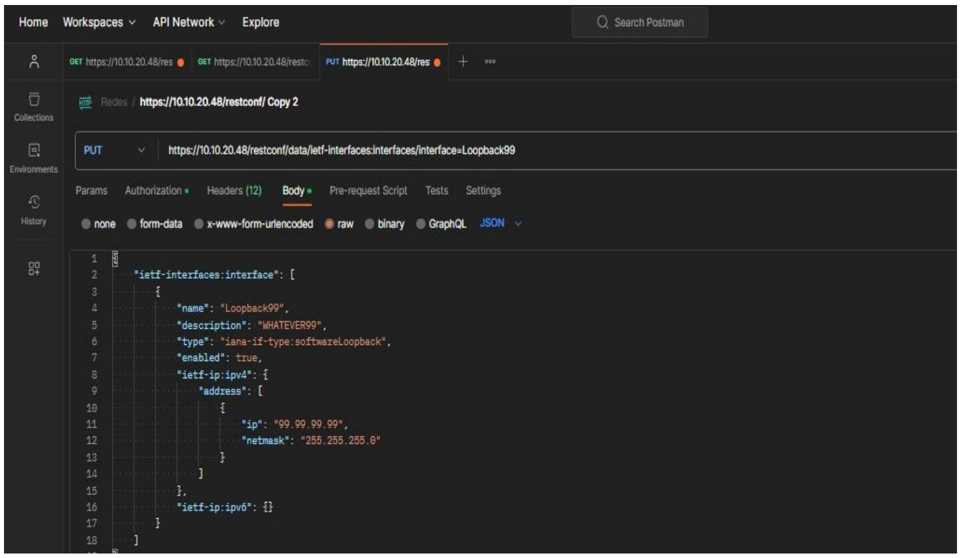
PUT https://192.168.56.101/restconf/data/ietf-interfaces:interfaces/interface=Loopback99

Body: raw, Text

```
{
    "ietf-interfaces:interface": {
        "name": "Loopback99",
        "description": "WHATEVER99",
        "type": "iana-if-type:softwareLoopback",
        "enabled": true,
        "ietf-ip:ipv4": {
            "address": [
                {
                    "ip": "99.99.99.99",
                    "netmask": "255.255.255.0"
                }
            ]
        },
        "ietf-ip:ipv6": {}
    }
}
```

d. Click **Send** to create a new interface Loopback99 with the IP address 99.99.99.99/24.

e. In the Output (scroll down), you should see the HTTP Error code **201 Created** indicating that the new interface has been created:

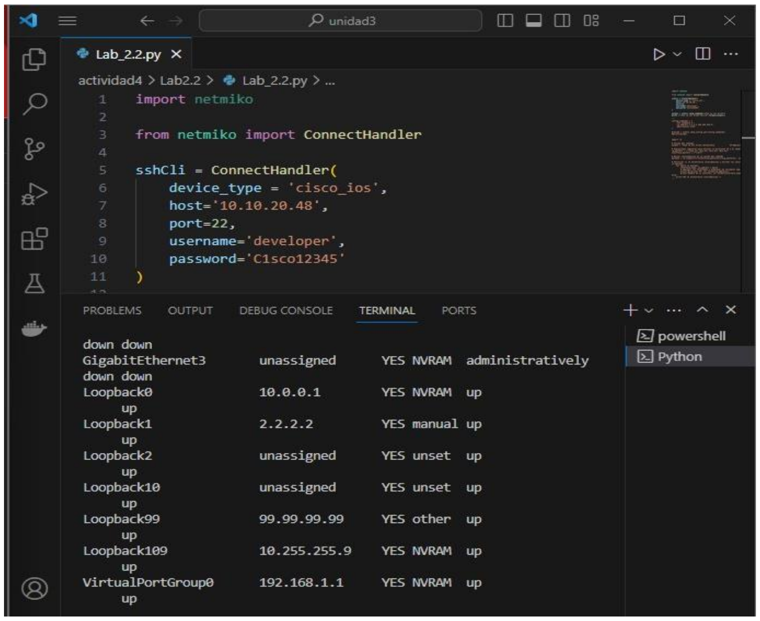


Redes / https://10.10.20.48/restconf/ Copy 2

PUT https://10.10.20.48/restconf/data/ietf-interfaces:interfaces/interface=Loopback99

```
{
    "ietf-interfaces:interface": [
        {
            "name": "Loopback99",
            "description": "WHATEVER99",
            "type": "iana-if-type:softwareLoopback",
            "enabled": true,
            "ietf-ip:ipv4": {
                "address": [
                    {
                        "ip": "99.99.99.99",
                        "netmask": "255.255.255.0"
                    }
                ]
            },
            "ietf-ip:ipv6": {}
        }
    ]
```

f. Verify using the IOS CLI that the new Loopback99 interface has been created (sh ip int brief).



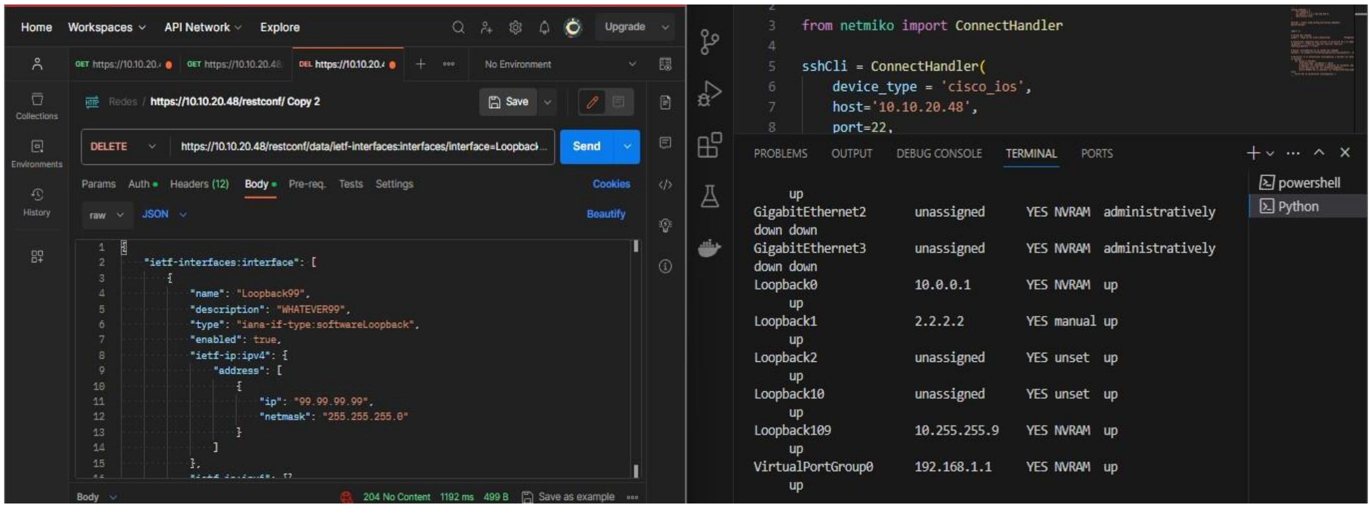
```python
import netmiko

from netmiko import ConnectHandler

sshCli = ConnectHandler(
    device_type = 'cisco_ios',
    host='10.10.20.48',
    port=22,
    username='developer',
    password='C1sco12345'
)
```

```
down down
GigabitEthernet3       unassigned       YES NVRAM  administratively
down down
Loopback0              10.0.0.1         YES NVRAM  up
   up
Loopback1              2.2.2.2          YES manual up
   up
Loopback2              unassigned       YES unset  up
   up
Loopback10             unassigned       YES unset  up
   up
Loopback99             99.99.99.99      YES other  up
   up
Loopback109            10.255.255.9     YES NVRAM  up
   up
VirtualPortGroup0      192.168.1.1      YES NVRAM  up
   up
```

g. Update the RESTCONF API request to delete the interface **Loopback99**.



Redes / https://10.10.20.48/restconf/ Copy 2

DELETE https://10.10.20.48/restconf/data/ietf-interfaces:interfaces/interface=Loopback…

```json
{
    "ietf-interfaces:interface": [
        {
            "name": "Loopback99",
            "description": "WHATEVER99",
            "type": "iana-if-type:softwareLoopback",
            "enabled": true,
            "ietf-ip:ipv4": {
                "address": [
                    {
                        "ip": "99.99.99.99",
                        "netmask": "255.255.255.0"
                    }
                ]
            },
```

204 No Content 1192 ms 499 B

```python
from netmiko import ConnectHandler

sshCli = ConnectHandler(
    device_type = 'cisco_ios',
    host='10.10.20.48',
    port=22,
```

```
   up
GigabitEthernet2       unassigned       YES NVRAM  administratively
down down
GigabitEthernet3       unassigned       YES NVRAM  administratively
down down
Loopback0              10.0.0.1         YES NVRAM  up
   up
Loopback1              2.2.2.2          YES manual up
   up
Loopback2              unassigned       YES unset  up
   up
Loopback10             unassigned       YES unset  up
   up
Loopback109            10.255.255.9     YES NVRAM  up
   up
VirtualPortGroup0      192.168.1.1      YES NVRAM  up
   up
```

h. What changes were needed to be made in the RESTCONF request to delete the interface?

Todos lo parametros que se utilizaron para crear la loopback99.