

Develop Python scripts using the requests library to perform RESTCONF operations:

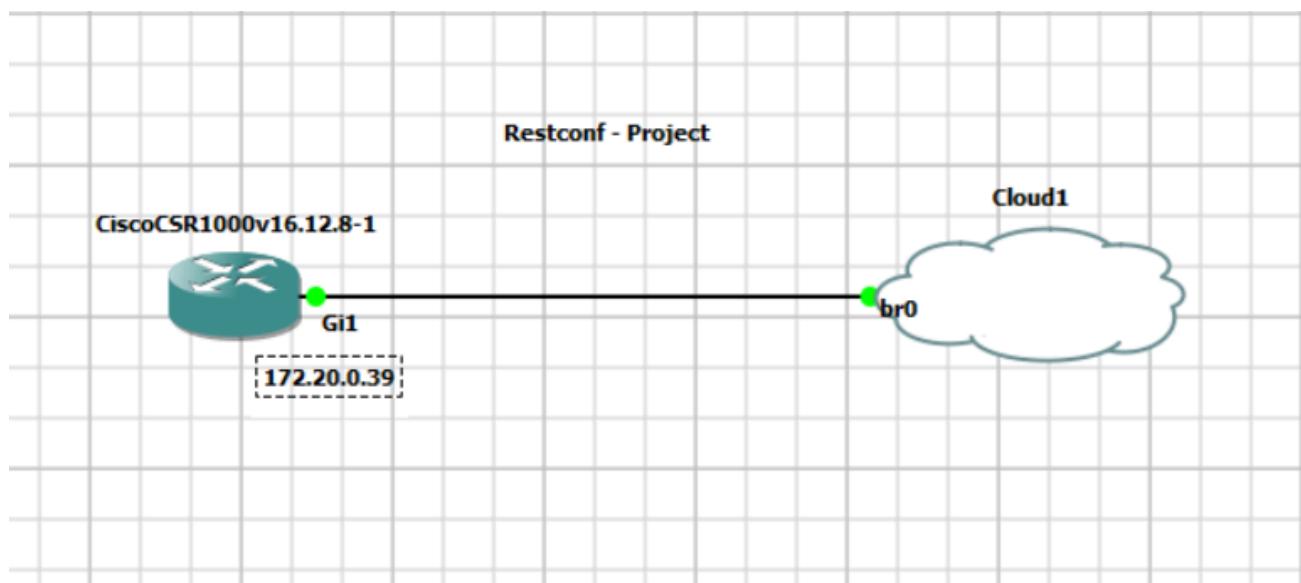
GET : Retrieves configuration and operational data.

POST : Create new resources.

PUT : Updates existing resources.

DELETE : Remove resources.

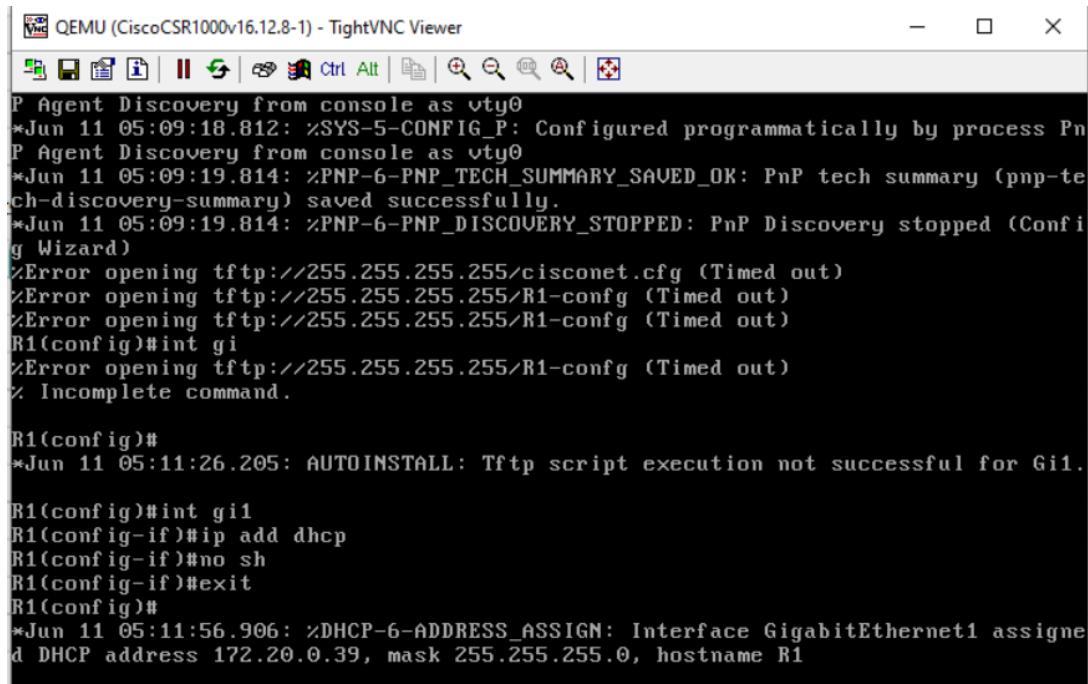
RESCONF Topology



Changing hostname

```
QEMU (CiscoCSR1000v16.8-1) - TightVNC Viewer
File | Ctrl Alt | Help | 
%Error opening tftp://255.255.255.255/network-config (Timed out)
Router>enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#hostname R1
R1(config)#
*Jun 11 05:09:10.318: %SMART LIC-6-HOSTNAME_MATCHED_UFI: The host name has been
```

Adding DHCP



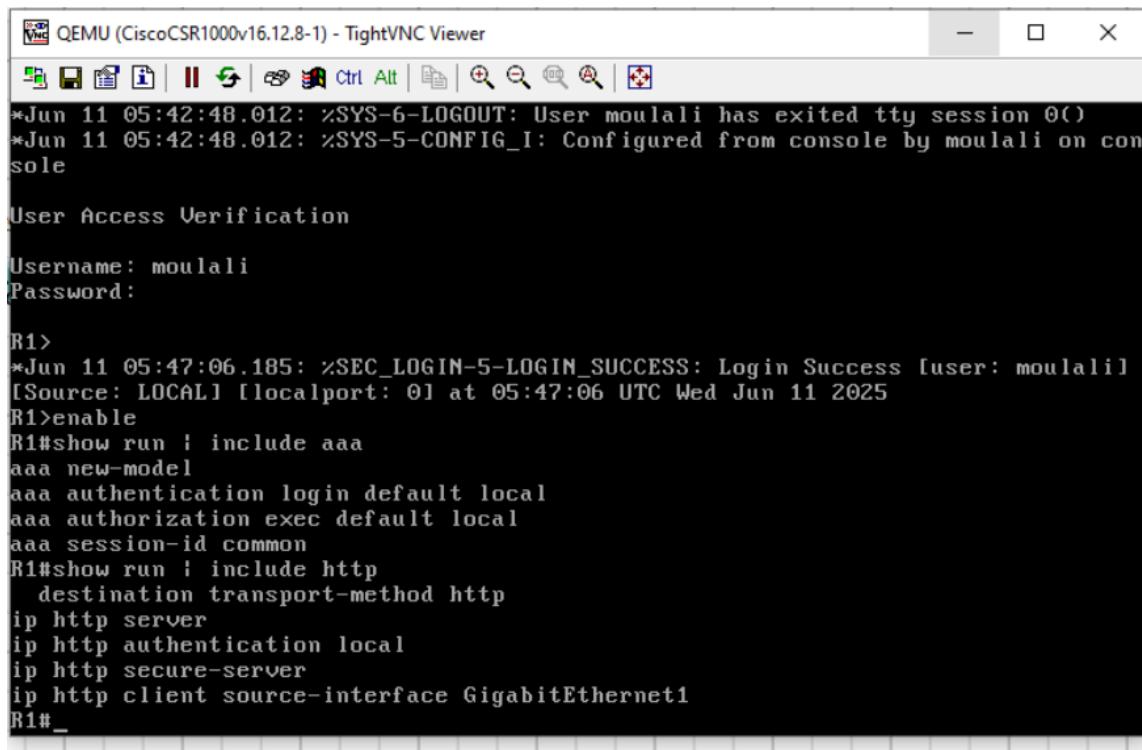
The screenshot shows a terminal session on a Cisco CSR1000v device. The session starts with PnP configuration messages, followed by an attempt to open a TFTP script which fails due to timeout. Then, the user enters configuration mode (R1(config)#) and tries to enable DHCP on interface Gi1, but receives an incomplete command error. Finally, the user configures interface Gi1 with static IP settings and enables DHCP assignment for it.

```
P Agent Discovery from console as vty0
*Jun 11 05:09:18.812: %SYS-5-CONFIG_P: Configured programmatically by process Pn
P Agent Discovery from console as vty0
*Jun 11 05:09:19.814: %PNP-6-PNP_TECH_SUMMARY_SAVED_OK: PnP tech summary (pnp-te
ch-discovery-summary) saved successfully.
*Jun 11 05:09:19.814: %PNP-6-PNP_DISCOVERY_STOPPED: PnP Discovery stopped (Confi
g Wizard)
%Error opening tftp://255.255.255.255/cisconet.cfg (Timed out)
%Error opening tftp://255.255.255.255/R1-config (Timed out)
%Error opening tftp://255.255.255.255/R1-config (Timed out)
R1(config)#int gi
%Error opening tftp://255.255.255.255/R1-config (Timed out)
% Incomplete command.

R1(config)#
*Jun 11 05:11:26.205: AUTOINSTALL: Tftp script execution not successful for Gi1.

R1(config)#int gi1
R1(config-if)#ip add dhcp
R1(config-if)#no sh
R1(config-if)#exit
R1(config)#
*Jun 11 05:11:56.906: %DHCP-6-ADDRESS_ASSIGN: Interface GigabitEthernet1 assigne
d DHCP address 172.20.0.39, mask 255.255.255.0, hostname R1
```

Show run | include aaa and show run | include http



The screenshot shows a terminal session on a Cisco CSR1000v device. It starts with a user logging out and another user (moulali) logging in. The user then runs 'show run | include aaa' which displays AAA configuration including new-model, authentication, authorization, and session-id settings. Next, the user runs 'show run | include http' which shows the configuration for an HTTP server on interface GigabitEthernet1.

```
*Jun 11 05:42:48.012: %SYS-6-LOGOUT: User moulali has exited tty session 0()
*Jun 11 05:42:48.012: %SYS-5-CONFIG_I: Configured from console by moulali on con
sole

User Access Verification

Username: moulali
Password:

R1>
*Jun 11 05:47:06.185: %SEC_LOGIN-5-LOGIN_SUCCESS: Login Success [user: moulali]
[Source: LOCAL] [localport: 0] at 05:47:06 UTC Wed Jun 11 2025
R1>enable
R1#show run | include aaa
aaa new-model
aaa authentication login default local
aaa authorization exec default local
aaa session-id common
R1#show run | include http
    destination transport-method http
    ip http server
    ip http authentication local
    ip http secure-server
    ip http client source-interface GigabitEthernet1
R1#_
```

Show ip int brief

The screenshot shows a terminal window titled "QEMU (CiscoCSR1000v16.12.8-1) - TightVNC Viewer". The window contains the following text:

```
onep          For ONEP authorization service
policy-if     For diameter policy interface application.
prepaid        For diameter prepaid services.
radius-proxy   For proxying radius packets
reverse-access For reverse access connections
reverse-telnet For reverse telnet connections, old command
subscriber-service For iEdge subscriber services (UPDM etc)
suppress       Do not send author request for a specific type of user.
template       Enable template authorization

R1(config)#aaa authorization exec default local
R1(config)#ip http server
R1(config)#ip http secure-server
R1(config)#ip http authentication local
R1(config)#do wr
Building configuration...
[OK]
R1(config)#do show ip int brief
Interface      IP-Address      OK? Method Status      Protocol
GigabitEthernet1 172.20.0.39    YES  DHCP   up        up
GigabitEthernet2 unassigned     YES  unset  down     down
GigabitEthernet3 unassigned     YES  unset  down     down
GigabitEthernet4 unassigned     YES  unset  down     down
R1(config)#_
```

Show run

The screenshot shows a terminal window titled "QEMU (CiscoCSR1000v16.12.8-1) - TightVNC Viewer". The window contains the following text:

```
no ip address
negotiation auto
no mop enabled
no mop sysid
!
ip forward-protocol nd
ip http server
ip http authentication local
ip http secure-server
ip http client source-interface GigabitEthernet1
!
!
!
!
control-plane
!
!
!
--More--
```

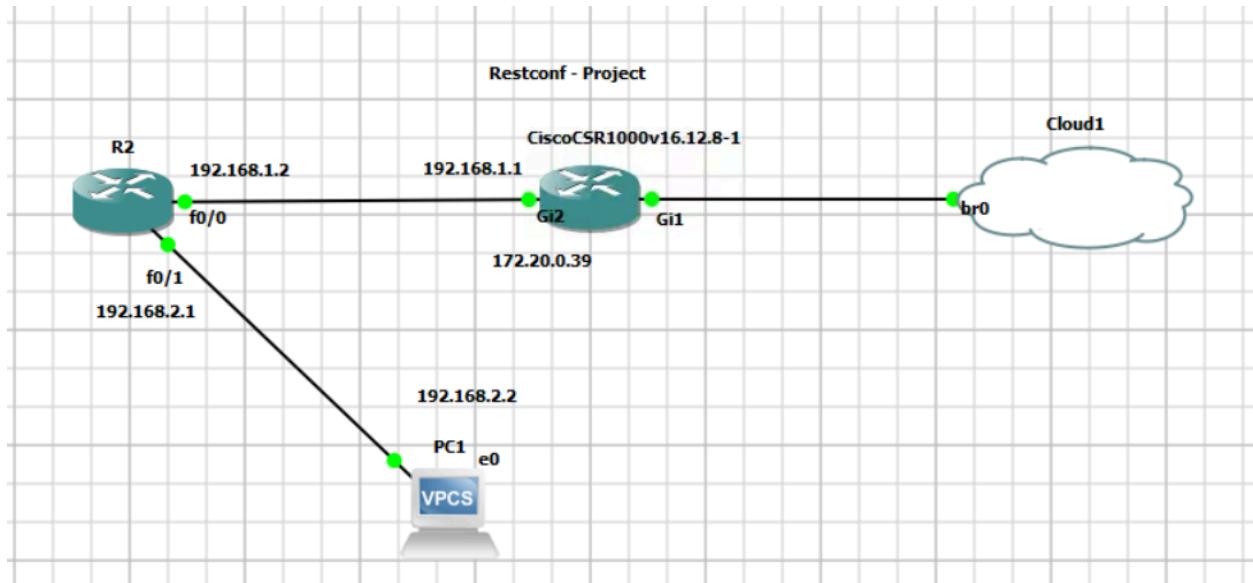
Adding restconf to the end of configuration

```
User Access Verification

Username: moulali
Password:

R1>en
*Jun 11 10:04:44.465: %SEC_LOGIN-5-LOGIN_SUCCESS: Login Success [user: moulali]
[Source: LOCAL] [localport: 0] at 10:04:44 UTC Wed Jun 11 2025
R1#config t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#restconf
R1(config)#
*Jun 11 10:05:03.183: %PSD_MOD-5-DMI_NOTIFY_RESTCONF_START: R0/0: psd: PSD/DMI:
restconf server has been notified to start
R1(config)#do wr
Building configuration...
-
```

Updated Topology



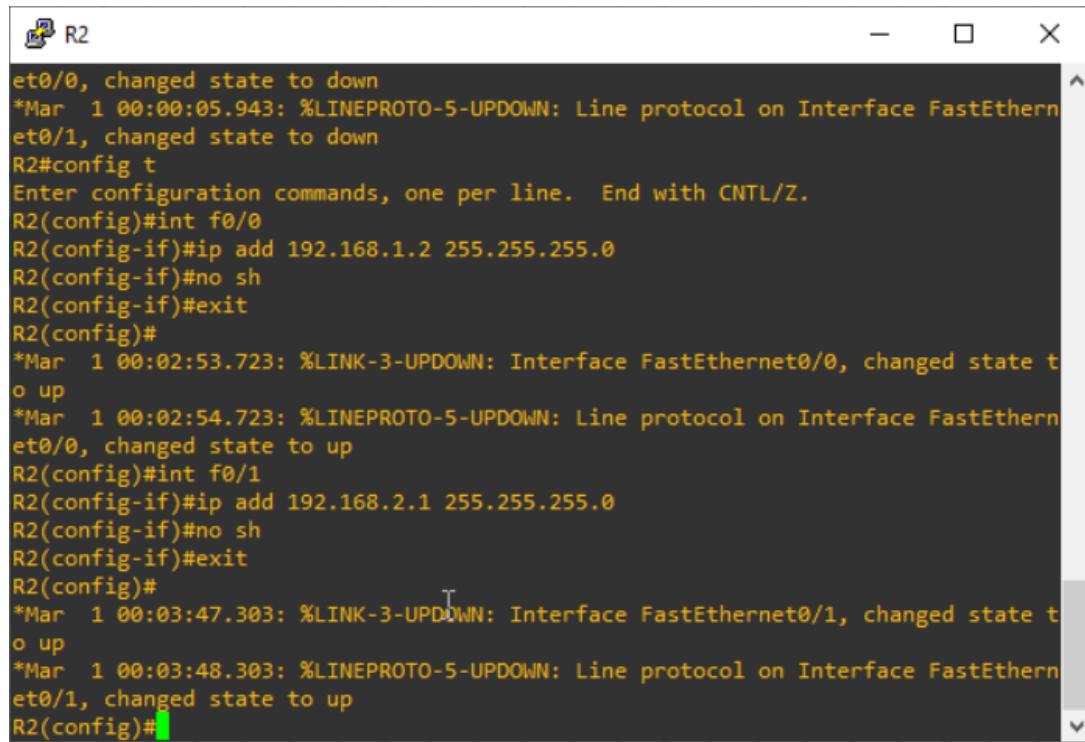
R2 Configurations

```
User Access Verification

Username: moulali
Password:

R1>
*Jun 11 15:32:06.226: %SEC_LOGIN-5-LOGIN_SUCCESS: Login Success [user: moulali]
[Source: LOCAL] [localport: 0] at 15:32:06 UTC Wed Jun 11 2025
R1>enable
R1#config t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#int gi2
R1(config-if)#ip add 192.168.1.1 255.255.255.0
R1(config-if)#no sh
R1(config-if)#exit
R1(config)#_
```

Adding ip address



The terminal window shows the configuration of two interfaces on router R2. It starts with a log message indicating the state transition of interface et0/0 from down to up. Then, it enters configuration mode (config t) and configures interface f0/0 with IP 192.168.1.2. After exiting configuration mode, it shows another log message for interface et0/0 transitioning back to up. It then configures interface f0/1 with IP 192.168.2.1. Finally, it exits configuration mode again.

```
R2
et0/0, changed state to down
*Mar 1 00:00:05.943: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0 changed state to up
R2#config t
Enter configuration commands, one per line. End with CNTL/Z.
R2(config)#int f0/0
R2(config-if)#ip add 192.168.1.2 255.255.255.0
R2(config-if)#no sh
R2(config-if)#exit
R2(config)#
*Mar 1 00:02:53.723: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
et0/1, changed state to down
R2(config)#int f0/1
R2(config-if)#ip add 192.168.2.1 255.255.255.0
R2(config-if)#no sh
R2(config-if)#exit
R2(config)#
*Mar 1 00:03:47.303: %LINK-3-UPDOWN: Interface FastEthernet0/1, changed state to up
et0/0, changed state to up
*Mar 1 00:03:48.303: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0 changed state to up
R2(config)#

```

PC ip address

```
PC1 - PuTTY
Welcome to Virtual PC Simulator, version 0.8.3
Dedicated to Daling.
Build time: Sep 9 2023 11:15:00
Copyright (c) 2007-2015, Paul Meng (mirnshi@gmail.com)
All rights reserved.

VPCS is free software, distributed under the terms of the "BSD" licence.
Source code and license can be found at vpcs.sf.net.
For more information, please visit wiki.freecode.com.cn.

Press '?' to get help.

Executing the startup file

PC1> ip 192.168.2.2 255.255.255.0 192.168.2.1
Checking for duplicate address...
PC1 : 192.168.2.2 255.255.255.0 gateway 192.168.2.1

PC1>
```

Configuring OSPF

```
R1(config)#  
R1(config)#router ospf 1  
R1(config-router)#  
*Jun 11 15:34:48.603: %OSPF-6-DFT_OPT: Protocol timers for fast convergence are  
Enabled.  
R1(config-router)#network 192.168.1.0 0.0.0.255 area 0  
R1(config-router)#exit  
R1(config)#  
  
et0/1, changed state to up  
R2(config)#router ospf 1  
R2(config-router)#network 192.168.1.0 0.0.0.255 area 0  
R2(config-router)#network 192.168  
*Mar 1 00:09:42.135: %OSPF-5-ADJCHG: Process 1, Nbr 192.168.1.1 on FastEthernet  
0/0 from LOADING to FULL, Loading Done  
R2(config-router)#network 192.168.2.0 0.0.0.255 area 0  
R2(config-router)#exit  
R2(config)#
```

ping

```
PC1> ping 192.168.1.1

84 bytes from 192.168.1.1 icmp_seq=1 ttl=254 time=20.083 ms
84 bytes from 192.168.1.1 icmp_seq=2 ttl=254 time=19.583 ms
84 bytes from 192.168.1.1 icmp_seq=3 ttl=254 time=10.919 ms
84 bytes from 192.168.1.1 icmp_seq=4 ttl=254 time=16.580 ms
84 bytes from 192.168.1.1 icmp_seq=5 ttl=254 time=16.079 ms

PC1> █
```

Show ip int brief

```
% Type 'show ?' for a list of subcommands
R1#show ip int brief
Interface          IP-Address      OK? Method Status          Protocol
GigabitEthernet1   172.20.0.39    YES DHCP   up           up
GigabitEthernet2   192.168.1.1    YES manual up           up
GigabitEthernet3   unassigned     YES NURAM  down         down
GigabitEthernet4   unassigned     YES NURAM  down         down
R1#_
```

Show run

```
R2
!
!
!
!
interface FastEthernet0/0
 ip address 192.168.1.2 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet0/1
 ip address 192.168.2.1 255.255.255.0
 duplex auto
 speed auto
!
router ospf 1
 log-adjacency-changes
 network 192.168.1.0 0.0.0.255 area 0
 network 192.168.2.0 0.0.0.255 area 0
!
ip forward-protocol nd
!
!
no ip http server
--More-- █
```

Show ip int brief

```
R2#show run | include ospf
router ospf 1
R2#show ip int brief
Interface          IP-Address      OK? Method Status      Prot
FastEthernet0/0    192.168.1.2    YES manual up
FastEthernet0/1    192.168.2.1    YES manual up
R2#
```

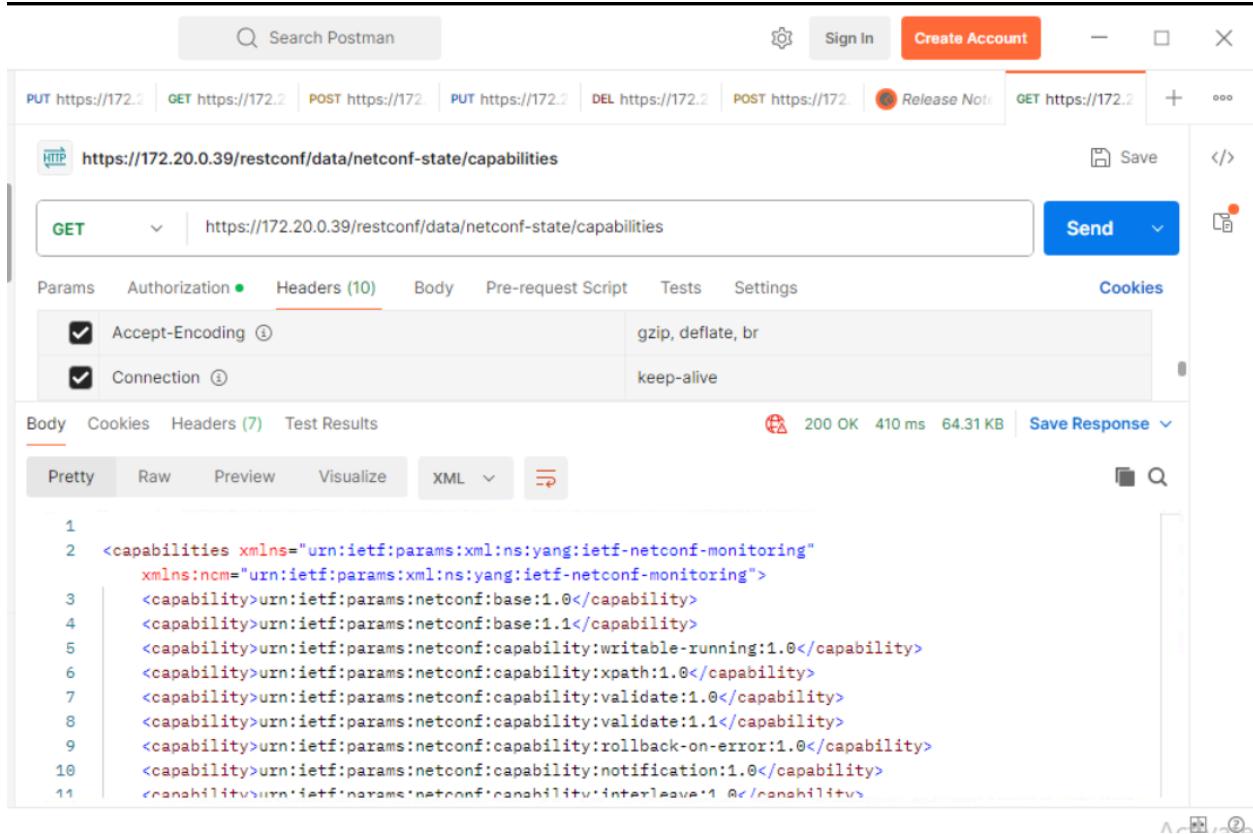
Postman

GET

The screenshot shows the Postman application interface. At the top, there is a search bar and several tabs for different requests. The main request card is set to a GET method and the URL <https://172.20.0.39/restconf>. The 'Authorization' tab is selected, showing 'Basic A...' and fields for 'Username' (mouali) and 'Password' (mouli). Below the request card, the response body is displayed in XML format:

```
1 <restconf xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
2   <data/>
3   <operations/>
4   <yang-library-version>2016-06-21</yang-library-version>
5 </restconf>
```

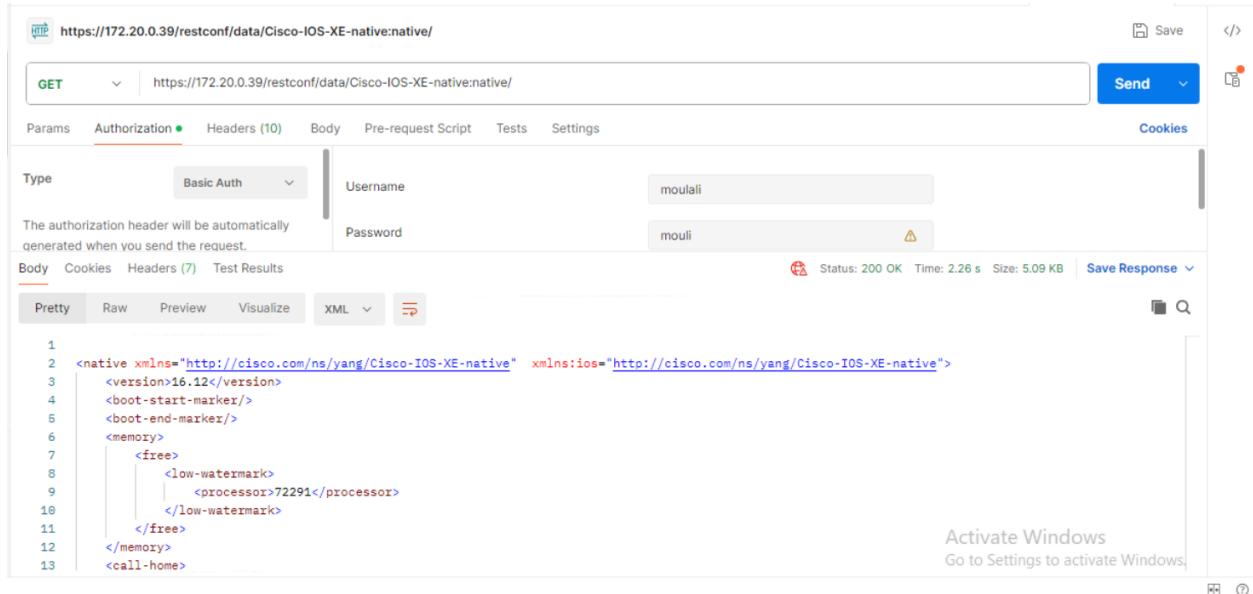
Capabilities



The screenshot shows the Postman interface with a GET request to `https://172.20.0.39/restconf/data/netconf-state/capabilities`. The response status is 200 OK, and the response body is a XML document listing various network monitoring capabilities:

```
1<capabilities xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
2  <capability>urn:ietf:params:netconf:base:1.0</capability>
3  <capability>urn:ietf:params:netconf:base:1.1</capability>
4  <capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
5  <capability>urn:ietf:params:netconf:capability>xpath:1.0</capability>
6  <capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
7  <capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
8  <capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
9  <capability>urn:ietf:params:netconf:capability:notification:1.0</capability>
10 <capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>
11 <capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>
```

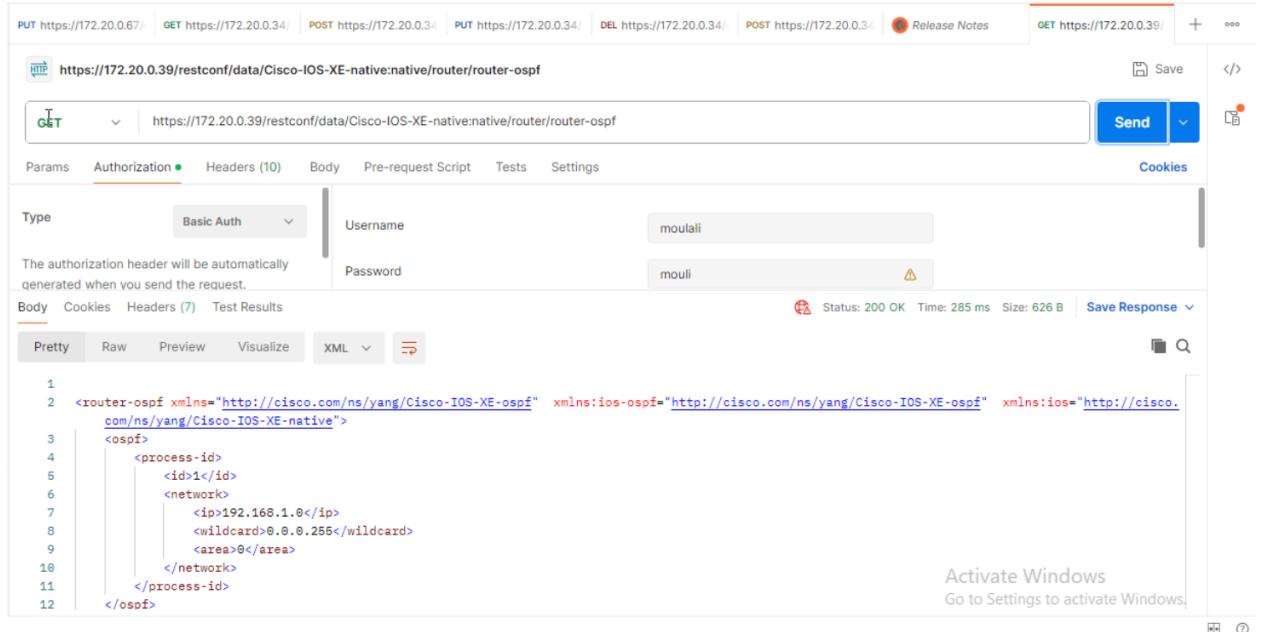
Native config



The screenshot shows the Postman interface with a GET request to `https://172.20.0.39/restconf/data/Cisco-IOS-XE-native:native/`. The response status is 200 OK, and the response body is a XML document showing native configuration details:

```
1<native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native" xmlns:ios="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
2  <version>16.12</version>
3  <boot-start-marker/>
4  <boot-end-marker/>
5  <memory>
6    <free>
7      <low-watermark>
8        <processor>72291</processor>
9        </low-watermark>
10   </free>
11   </memory>
12   <call-home>
```

Get OSPF



https://172.20.0.39/restconf/data/Cisco-IOS-XE-native:native/router/router-ospf

GET https://172.20.0.39/restconf/data/Cisco-IOS-XE-native:native/router/router-ospf

Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

Type Basic Auth Username moulali Password mouli

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

Body Cookies Headers (7) Test Results

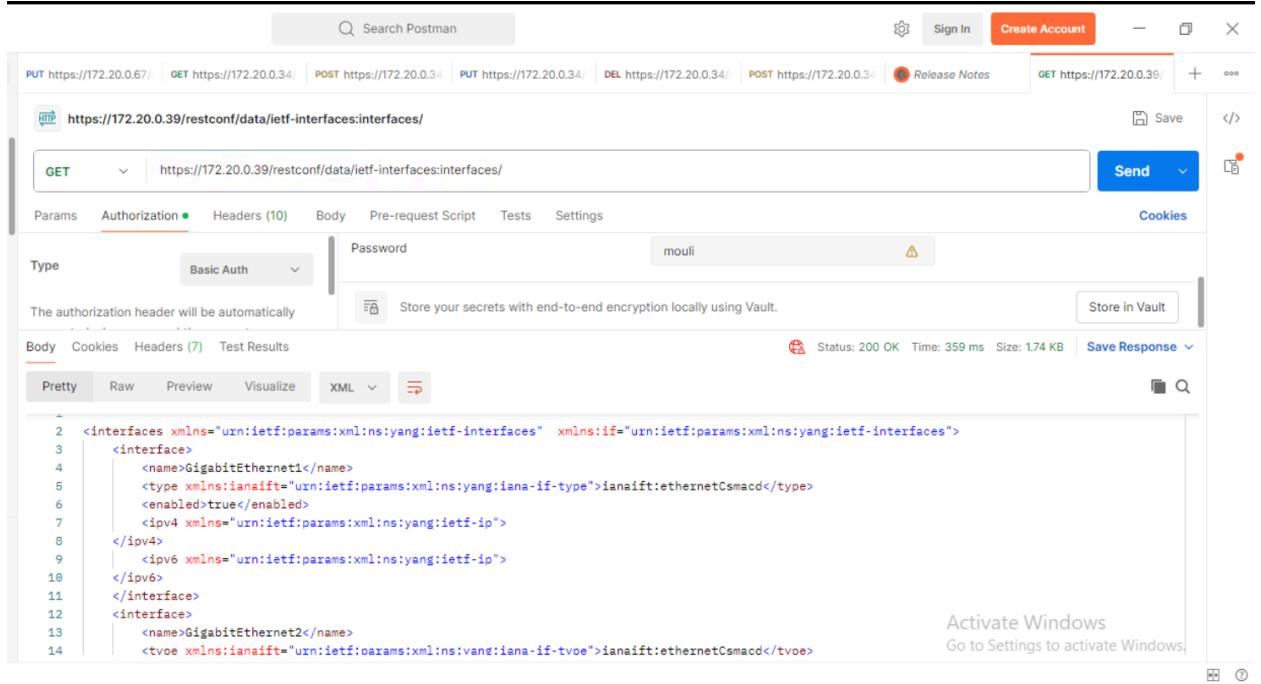
Pretty Raw Preview Visualize XML

```
1 <router-ospf xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ospf" xmlns:ios-ospf="http://cisco.com/ns/yang/Cisco-IOS-XE-ospf" xmlns:ios="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
2   <ospf>
3     <process-id>
4       <id>1</id>
5       <network>
6         <ip>192.168.1.0</ip>
7         <wildcard>0.0.0.255</wildcard>
8         <area>0</area>
9       </network>
10      </process-id>
11    </ospf>
```

Status: 200 OK Time: 285 ms Size: 626 B Save Response

Activate Windows
Go to Settings to activate Windows.

Interfaces



https://172.20.0.39/restconf/data/ietf-interfaces:interfaces

GET https://172.20.0.39/restconf/data/ietf-interfaces:interfaces/

Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

Type Basic Auth Password moulali

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

Body Cookies Headers (7) Test Results

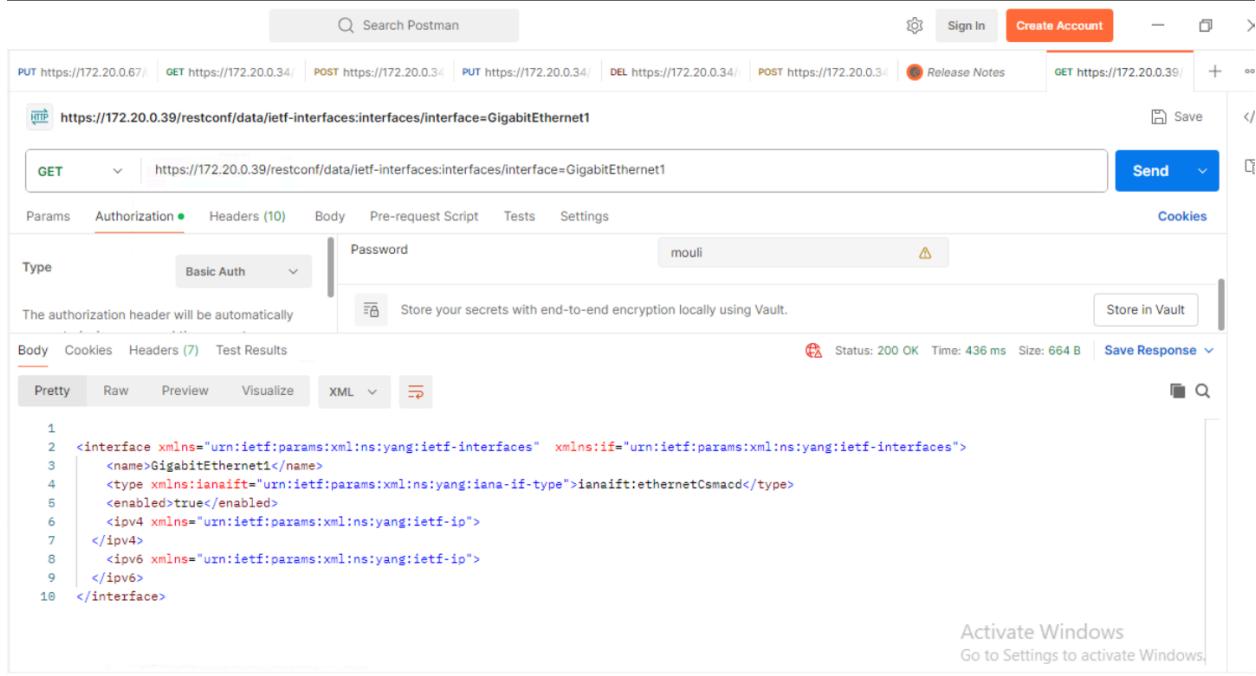
Pretty Raw Preview Visualize XML

```
1 <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces" xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">
2   <interface>
3     <name>GigabitEthernet1</name>
4     <type xmlns:ianaIfType="urn:ietf:params:xml:ns:yang:iana-if-type">ianaIfType:ethernetCsmacd</type>
5     <enabled>true</enabled>
6     <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
7       <ipv4>
8       <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
9         <ipv6>
10        <interface>
11          <name>GigabitEthernet2</name>
12          <tvvoe xmlns:ianaIfType="urn:ietf:params:xml:ns:yang:iana-if-type">ianaIfType:ethernetCsmacd</tvvoe>
```

Status: 200 OK Time: 359 ms Size: 1.74 KB Save Response

Activate Windows
Go to Settings to activate Windows.

Interface = GigabitEthernet1



Search Postman

PUT https://172.20.0.67/ | GET https://172.20.0.34/ | POST https://172.20.0.34/ | PUT https://172.20.0.34/ | DEL https://172.20.0.34/ | POST https://172.20.0.34/ | Release Notes | GET https://172.20.0.39/ | + | [Create Account](#)

https://172.20.0.39/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet1

GET https://172.20.0.39/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet1

Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

Type Basic Auth Password mouli

The authorization header will be automatically stored in Vault. Store in Vault

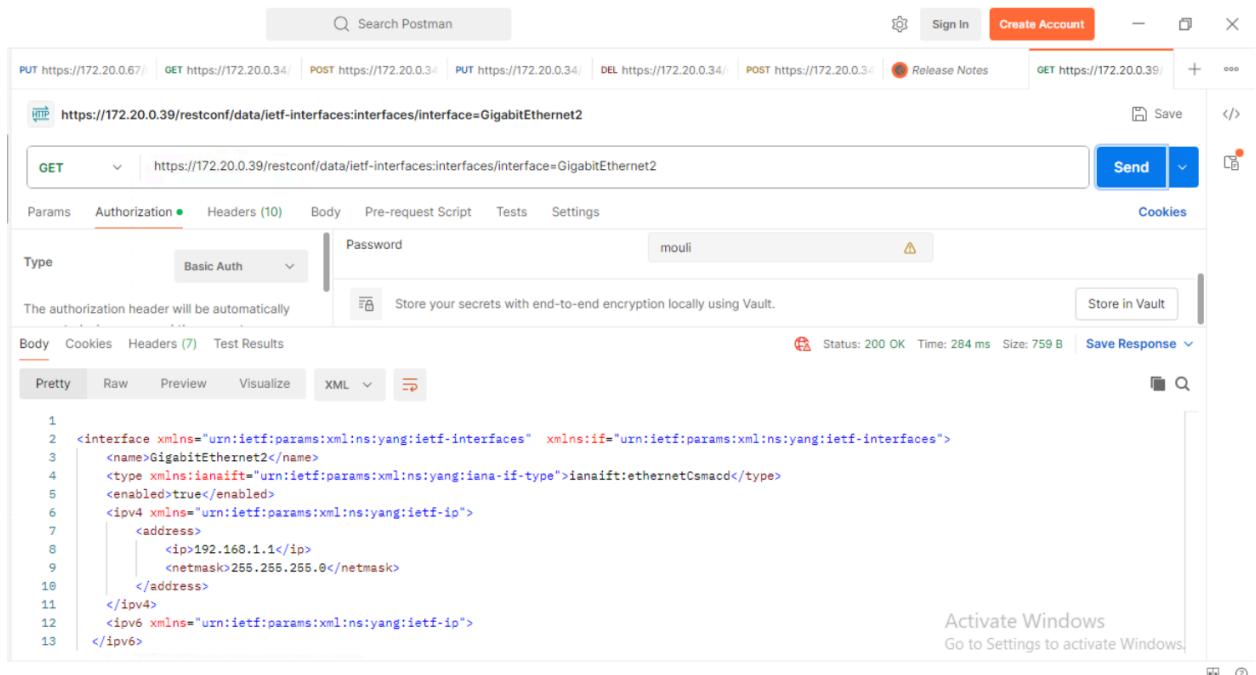
Pretty Raw Preview Visualize XML

1 <interface xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces" xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">
2 <name>GigabitEthernet1</name>
3 <type xmlns:ianaIf="urn:ietf:params:xml:ns:yang:iana-if-type">ianaIf:ethernetCsmacd</type>
4 <enabled>true</enabled>
5 <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
6 </ipv4>
7 <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
8 </ipv6>
9 </interface>

Status: 200 OK Time: 436 ms Size: 664 B Save Response

Activate Windows
Go to Settings to activate Windows.

Interface = GigabitEthernet2



Search Postman

PUT https://172.20.0.67/ | GET https://172.20.0.34/ | POST https://172.20.0.34/ | PUT https://172.20.0.34/ | DEL https://172.20.0.34/ | POST https://172.20.0.34/ | Release Notes | GET https://172.20.0.39/ | + | [Create Account](#)

https://172.20.0.39/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet2

GET https://172.20.0.39/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet2

Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

Type Basic Auth Password mouli

The authorization header will be automatically stored in Vault. Store in Vault

Pretty Raw Preview Visualize XML

1 <interface xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces" xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">
2 <name>GigabitEthernet2</name>
3 <type xmlns:ianaIf="urn:ietf:params:xml:ns:yang:iana-if-type">ianaIf:ethernetCsmacd</type>
4 <enabled>true</enabled>
5 <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
6 <address>
7 <ip>192.168.1.1</ip>
8 <netmask>255.255.255.0</netmask>
9 </address>
10 </ipv4>
11 <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
12 </ipv6>

Status: 200 OK Time: 284 ms Size: 759 B Save Response

Activate Windows
Go to Settings to activate Windows.

POST - Adding loopback

The screenshot shows a Postman interface with the following details:

- Request URL:** https://172.20.0.39/restconf/data/ietf-interfaces:interfaces/
- Method:** POST
- Body:** JSON (selected)
- JSON Body Content:**

```
1 {
2     "ietf-interfaces:interface": {
3         "name": "Loopback24",
4         "description": "NMS_RESTCONF",
5         "type": "iana-if-type:softwareLoopback",
6         "enabled": true,
7         "ietf-ip:ipv4": [
8             {
9                 "address": [
10                    {
11                        "ip": "24.24.24.24",
12                        "netmask": "255.255.255.255"
13                    }
14                ]
15            }
16        ]
17    }
18 }
```
- Status:** 201 Created
- Time:** 1548 ms
- Size:** 396 B

Output

The screenshot shows a terminal window titled "QEMU (CiscoCSR1000v16.12.8-1) - TightVNC Viewer". The output of the configuration command is as follows:

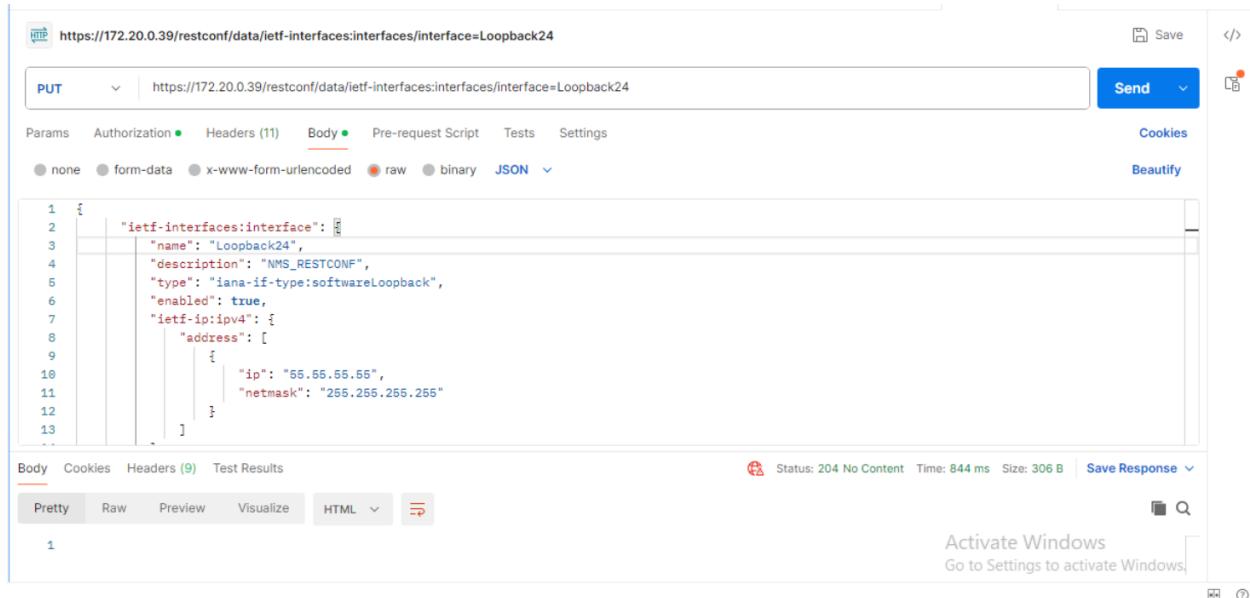
```
icated successfully from 172.20.0.43:0 for rest over http. External groups: PRI V15
*Jun 11 16:08:37.471: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback24
, changed state to up
*Jun 11 16:08:37.634: %SYS-5-CONFIG_P: Configured programmatically by process io
sp_vty_100001_dmi_nesd from console as NETCONF on vty32131
*Jun 11 16:08:37.636: %DMI-5-CONFIG_I: R0/0: nesd: Configured from NETCONF/RESTC
ONF by moulali, transaction-id 69

User Access Verification

Username: moulali
Password:

R1>enab
*Jun 11 16:09:09.194: %SEC_LOGIN-5-LOGIN_SUCCESS: Login Success [user: moulali]
[Source: LOCAL] [localport: 0] at 16:09:09 UTC Wed Jun 11 2025le
R1#show ip int brief
Interface          IP-Address      OK? Method Status      Protocol
GigabitEthernet1   172.20.0.39    YES  DHCP   up       up
GigabitEthernet2   192.168.1.1   YES  manual  up       up
GigabitEthernet3   unassigned     YES  NURAM  down     down
GigabitEthernet4   unassigned     YES  NURAM  down     down
Loopback24        24.24.24.24  YES  other   up       up
R1#
```

PUT - Overwriting Loopback

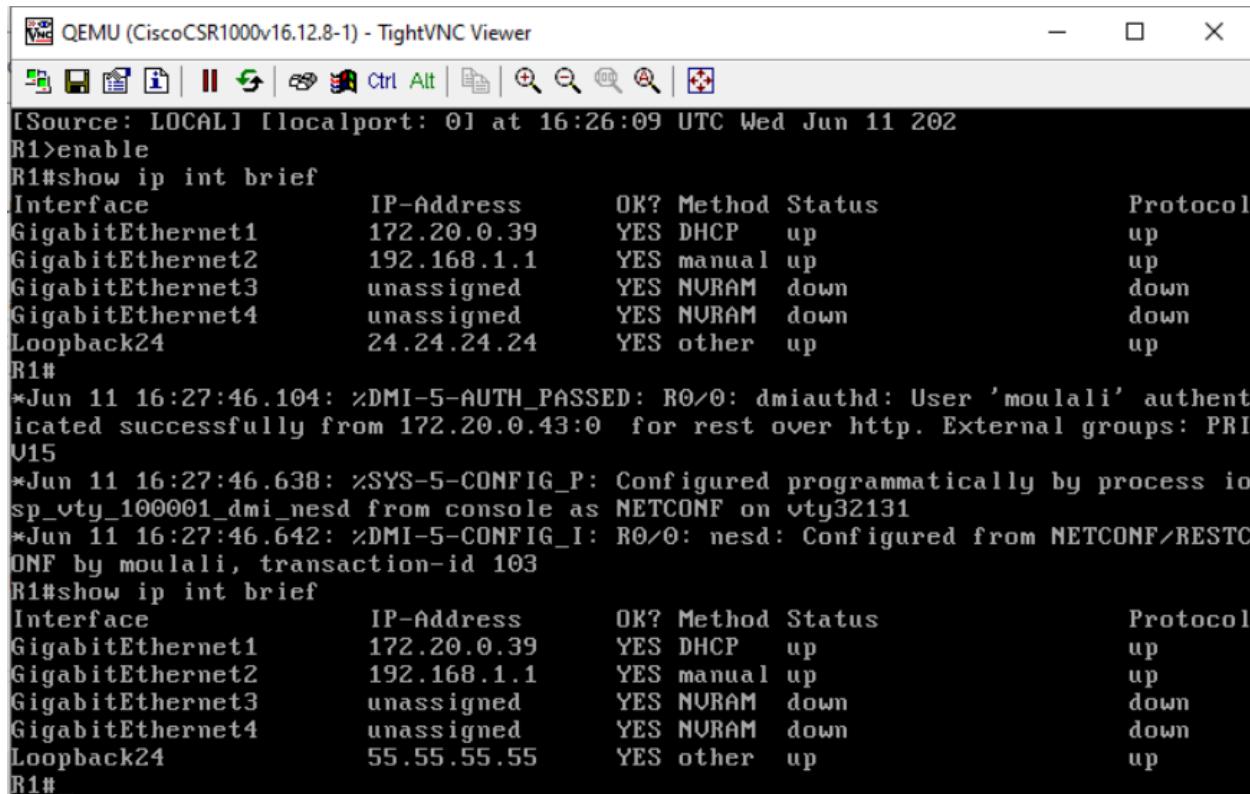


The screenshot shows a POST request in Postman to the URL `https://172.20.0.39/restconf/dataietf-interfaces:interfaces/interface=Loopback24`. The request body is a JSON object:

```
1 {  
2     "ietf-interfaces:interface": [  
3         {"name": "Loopback24",  
4             "description": "NMS_RESTCONF",  
5             "type": "iana-if-type:softwareLoopback",  
6             "enabled": true,  
7             "ietf-ip:ipv4": [  
8                 {"address": [  
9                     {"ip": "66.66.66.66",  
10                     "netmask": "255.255.255.255"  
11                 ]  
12             ]  
13         ]  
14     ]  
15 }
```

The response status is 204 No Content, time 844 ms, size 306 B.

output



```
[Source: LOCAL] [localport: 0] at 16:26:09 UTC Wed Jun 11 2022  
R1>enable  
R1#show ip int brief  
Interface          IP-Address      OK? Method Status      Protocol  
GigabitEthernet1  172.20.0.39    YES DHCP   up           up  
GigabitEthernet2  192.168.1.1   YES manual up           up  
GigabitEthernet3  unassigned     YES NVRAM  down          down  
GigabitEthernet4  unassigned     YES NVRAM  down          down  
Loopback24        24.24.24.24  YES other  up           up  
R1#  
*Jun 11 16:27:46.104: %DMI-5-AUTH_PASSED: R0/0: dmiauthd: User 'moulali' authenticated successfully from 172.20.0.43:0 for rest over http. External groups: PRI_U15  
*Jun 11 16:27:46.638: %SYS-5-CONFIG_P: Configured programmatically by process iosp_vty_100001_dmi_nesd from console as NETCONF on vty32131  
*Jun 11 16:27:46.642: %DMI-5-CONFIG_I: R0/0: nesd: Configured from NETCONF/RESTCONF by moulali, transaction-id 103  
R1#show ip int brief  
Interface          IP-Address      OK? Method Status      Protocol  
GigabitEthernet1  172.20.0.39    YES DHCP   up           up  
GigabitEthernet2  192.168.1.1   YES manual up           up  
GigabitEthernet3  unassigned     YES NVRAM  down          down  
GigabitEthernet4  unassigned     YES NVRAM  down          down  
Loopback24        55.55.55.55   YES other  up           up  
R1#
```

Delete

Removing Loopback

The screenshot shows a POSTMAN interface. The URL is `https://172.20.0.39/restconf/data/ietf-interfaces:interfaces/interface=Loopback24`. The method is set to `DELETE`. The Body tab contains the following JSON payload:

```
1 {
  "ietf-interfaces:interface": [
    {
      "name": "Loopback24",
      "description": "NMS_RESTCONF",
      "type": "iana-if-type:softwareLoopback",
      "enabled": true,
      "ietf-ip:ipv4": [
        {
          "address": [
            {
              "ip": "66.66.66.66",
              "netmask": "255.255.255.255"
            }
          ]
        }
      ]
    }
  ]
}
```

The Response section shows a status of `204 No Content`. Below the interface list, there is a terminal window showing the configuration of the loopback interface.

Output

The screenshot shows a terminal window titled "QEMU (CiscoCSR1000v16.12.8-1) - TightVNC Viewer". The terminal output shows the configuration of the Loopback24 interface:

```
*Jun 11 16:43:16.870: %SYS-5-CONFIG_P: Configured programmatically by process io_sp_vty_100001_dmi_nesd from console as NETCONF on vty32131
*Jun 11 16:43:16.873: %DMI-5-CONFIG_I: R0/0: nesd: Configured from NETCONF/RESTCONF by moulali, transaction-id 113
*Jun 11 16:43:18.772: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback24
, changed state to down
*Jun 11 16:43:18.773: %LINK-5-CHANGED: Interface Loopback24, changed state to administratively down
```

The terminal also shows user access verification:

```
User Access Verification
Username: moulali
Password:
```

And the output of the `show ip int brief` command:

| Interface | IP-Address | OK? | Method | Status | Protocol |
|------------------|-------------|-----|--------|--------|----------|
| GigabitEthernet1 | 172.20.0.39 | YES | DHCP | up | up |
| GigabitEthernet2 | 192.168.1.1 | YES | manual | up | up |
| GigabitEthernet3 | unassigned | YES | NVRAM | down | down |
| GigabitEthernet4 | unassigned | YES | NVRAM | down | down |

Python scripts

Pip install requests

The screenshot shows the Visual Studio Code interface. The left sidebar displays a project structure with 'RESTCONF_PROJECT' and '.venv'. The main editor area shows the code for 'headers and authentication.py':

```
import requests
import json
from requests.auth import HTTPBasicAuth

# Replace these variables with your actual device credentials
host = "https://172.20.0.39"
username = "moulli"
password = "moulli"

headers = {
    "Content-Type": "application/yang-data+json",
    "Accept": "application/yang-data+json"
}

# Disable SSL warnings (use only in dev/test environments)
requests.packages.urllib3.disable_warnings()
```

The terminal at the bottom shows the command 'pip install requests' being run:

```
PS C:\Users\Administrator\Desktop\RESTCONF_PROJECT> pip install requests
Collecting requests
  Downloading requests-2.32.4-py3-none-any.whl.metadata (4.9 kB)
Collecting charset_normalizer-3.4.2-cp313-cp313-win_amd64.whl.metadata (36 kB)
Collecting idna[4,>=2.5] (from requests)
  Downloading idna-3.10-py3-none-any.whl.metadata (10 kB)
Collecting urllib3[>2.4.0-py3-none-any.whl.metadata (6.5 kB)
Collecting certifi[>=2017.4.17] (from requests)
  Downloading certifi-2025.4.26-py3-none-any.whl.metadata (2.5 kB)
Collecting requests[>=2.32.4-py3-none-any.whl (64 kB)
Collecting charset_normalizer[>=3.4.2-cp313-cp313-win_amd64.whl (105 kB)
```

Headers and authentication file

The screenshot shows the Visual Studio Code interface. The left sidebar displays a project structure with 'RESTCONF_PROJECT' and '.venv'. The main editor area shows the same code for 'headers and authentication.py' as in the previous screenshot.

```
import requests
import json
from requests.auth import HTTPBasicAuth

# Replace these variables with your actual device credentials
host = "https://172.20.0.39"
username = "moulli"
password = "moulli"

headers = {
    "Content-Type": "application/yang-data+json",
    "Accept": "application/yang-data+json"
}

# Disable SSL warnings (use only in dev/test environments)
requests.packages.urllib3.disable_warnings()
```

The terminal at the bottom shows the command being run:

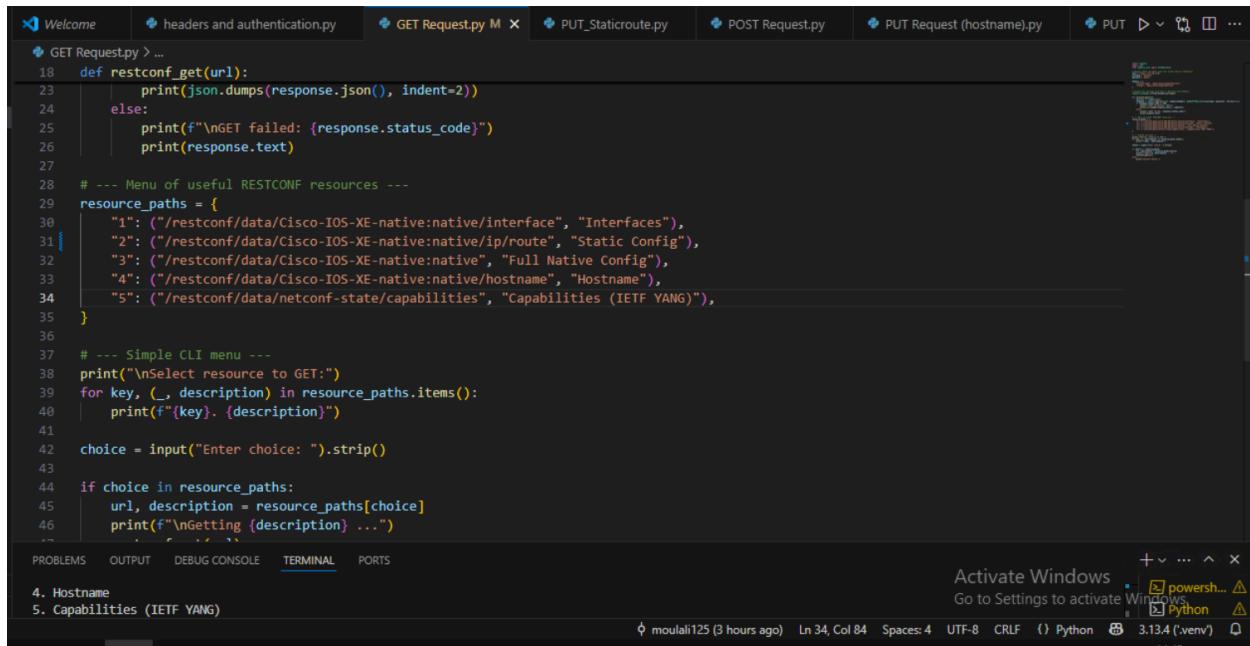
```
PS C:\Users\Administrator\Desktop\RESTCONF_PROJECT> & C:/Users/Administrator/Desktop/RESTCONF_PROJECT/.venv/Scripts/python.exe "c:/Users/Administrator/Desktop/RESTCONF_PROJECT/headers_and_authentication.py"
```

Output from the terminal:

```
PS C:\Users\Administrator\Desktop\RESTCONF_PROJECT>
```

Get Request

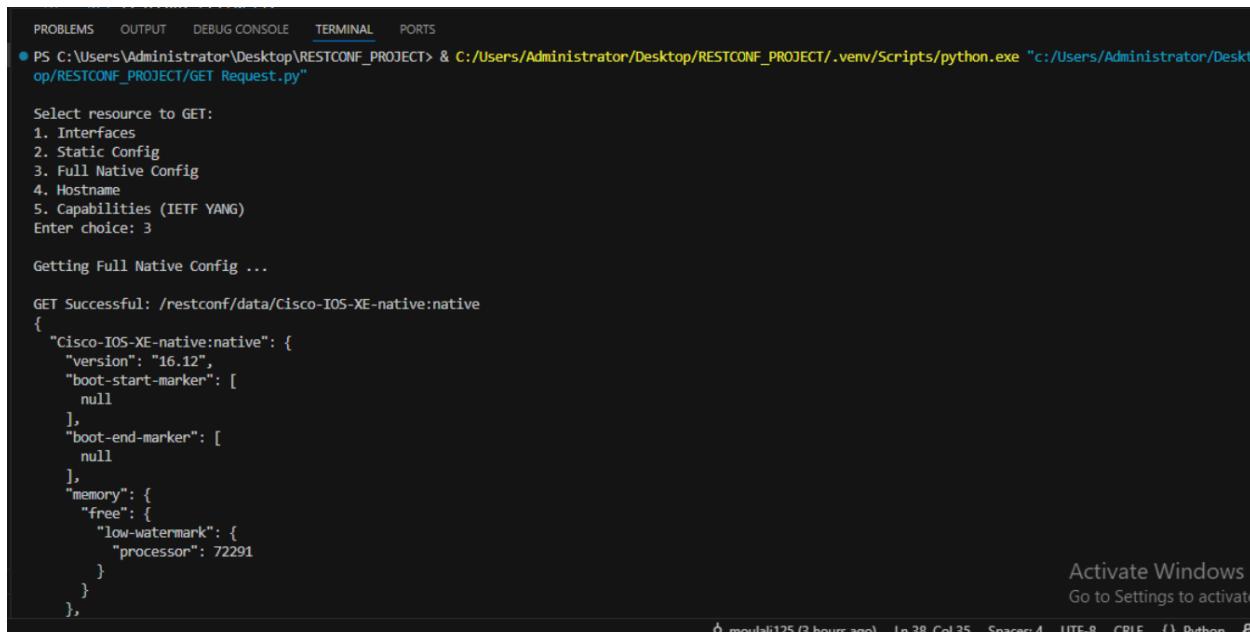
Interfaces, Static config, Full native config, Hostname and Capabilities



The screenshot shows a VS Code interface with a Python script named `GET Request.py`. The code implements a simple CLI menu to select a RESTCONF resource to GET. It defines a dictionary of resource paths and descriptions, prints a menu, and then reads user input to make the selected GET request. A terminal tab is open, showing the execution of the script and its output.

```
18     def restconf_get(url):
19         print(json.dumps(response.json(), indent=2))
20     else:
21         print(f"\nGET failed: {response.status_code}")
22         print(response.text)
23
24     # --- Menu of useful RESTCONF resources ---
25     resource_paths = {
26         "1": ("/restconf/data/Cisco-IOS-XE-native:native/interface", "Interfaces"),
27         "2": ("/restconf/data/Cisco-IOS-XE-native:native/ip/route", "Static Config"),
28         "3": ("/restconf/data/Cisco-IOS-XE-native:native", "Full Native Config"),
29         "4": ("/restconf/data/Cisco-IOS-XE-native:native/hostname", "Hostname"),
30         "5": ("/restconf/data/netconf-state/capabilities", "Capabilities (IETF YANG)"),
31     }
32
33     # --- Simple CLI menu ---
34     print("\nSelect resource to GET:")
35     for key, (url, description) in resource_paths.items():
36         print(f"{key}. {description}")
37
38     choice = input("Enter choice: ").strip()
39
40     if choice in resource_paths:
41         url, description = resource_paths[choice]
42         print(f"\nGetting {description} ...")
43
44     else:
45         print("Invalid choice")
46
47
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Activate Windows powershell... Go to Settings to activate Windows PowerShell
4. Hostname
5. Capabilities (IETF YANG)
```

Terminal Output



The screenshot shows a terminal window with the following session:

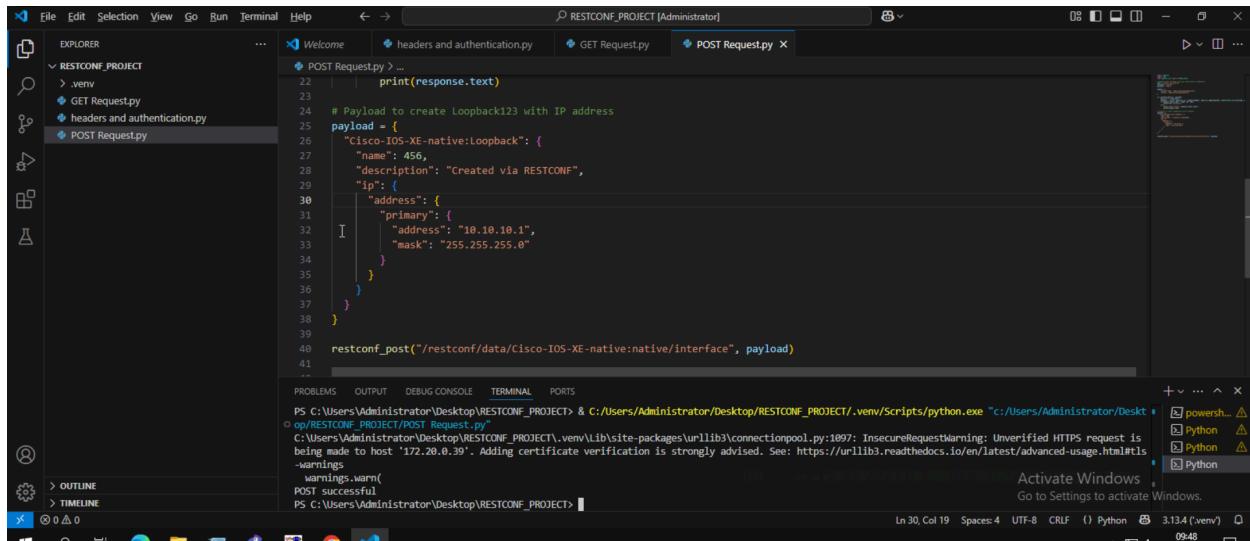
```
P5 C:/Users/Administrator/Desktop/RESTCONF_PROJECT> & C:/Users/Administrator/Desktop/RESTCONF_PROJECT/.venv/Scripts/python.exe "c:/Users/Administrator/Desktop/RESTCONF_PROJECT/GET Request.py"

Select resource to GET:
1. Interfaces
2. Static Config
3. Full Native Config
4. Hostname
5. Capabilities (IETF YANG)
Enter choice: 3

Getting Full Native Config ...

GET Successful: /restconf/data/Cisco-IOS-XE-native:native
{
  "Cisco-IOS-XE-native:native": {
    "version": "16.12",
    "boot-start-marker": [
      null
    ],
    "boot-end-marker": [
      null
    ],
    "memory": {
      "free": {
        "low-watermark": {
          "processor": 72291
        }
      }
    },
    "null"
  }
}
```

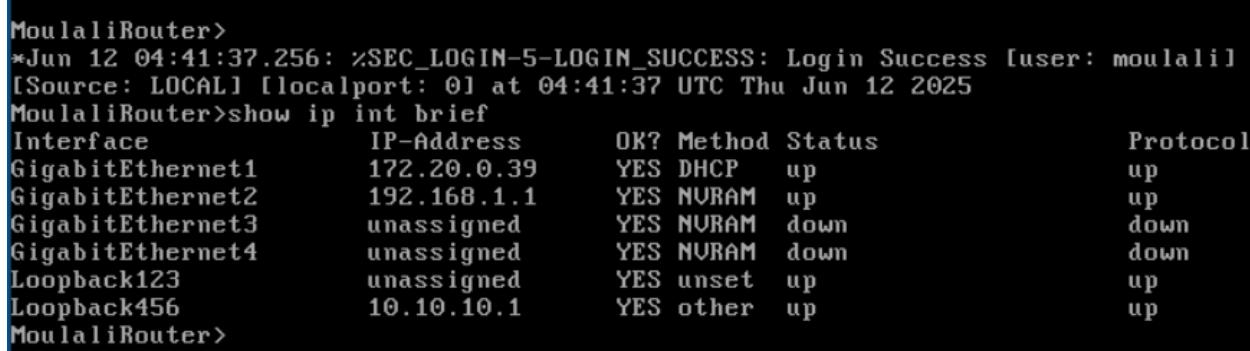
POST Request Adding Loopback



The screenshot shows the Visual Studio Code interface with a Python project named "RESTCONF_PROJECT". The "POST Request.py" file is open in the editor, containing code to create a Loopback interface. The code uses the `restconf_post` function to send a POST request to the RESTCONF endpoint for creating an interface. The payload is defined as a dictionary with "name": "456" and "description": "created via RESTCONF". The "ip": field contains a dictionary with "address": {"primary": {"address": "10.10.10.1", "mask": "255.255.255.0"}}, indicating a primary IP address of 10.10.10.1 with a subnet mask of 255.255.255.0.

```
POST Request.py
...
22     print(response.text)
23
24     # Payload to create Loopback123 with IP address
25     payload = {
26         "Cisco-IOS-XE-native:loopback": {
27             "name": 456,
28             "description": "created via RESTCONF",
29             "ip": {
30                 "address": {
31                     "primary": {
32                         "address": "10.10.10.1",
33                         "mask": "255.255.255.0"
34                     }
35                 }
36             }
37         }
38     }
39
40     restconf_post("/restconf/data/Cisco-IOS-XE-native:native/interface", payload)
41 
```

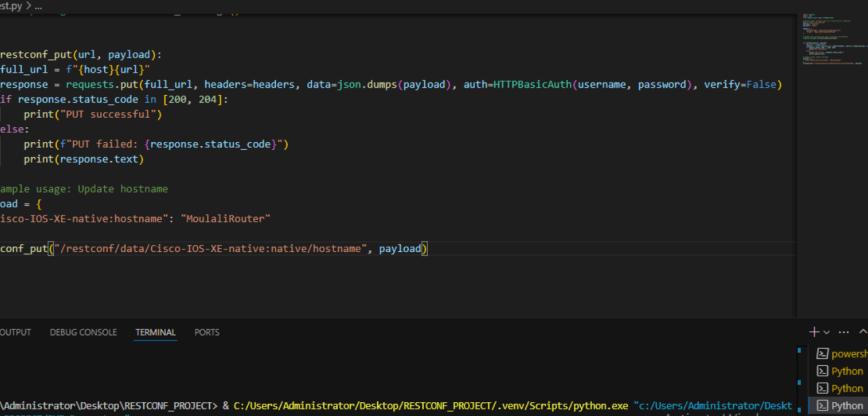
Output



The terminal output shows the user logging in successfully as "moulali" on the router. Then, the command `show ip int brief` is run, displaying the current interface configuration. A new interface, "Loopback123", has been added with the IP address 10.10.10.1.

```
MoulaliRouter>
*Jun 12 04:41:37.256: %SEC_LOGIN-5-LOGIN_SUCCESS: Login Success [user: moulali]
[Source: LOCAL] [localport: 0] at 04:41:37 UTC Thu Jun 12 2025
MoulaliRouter>show ip int brief
Interface          IP-Address      OK? Method Status        Protocol
GigabitEthernet1   172.20.0.39    YES  DHCP   up           up
GigabitEthernet2   192.168.1.1    YES  NURAM  up           up
GigabitEthernet3   unassigned     YES  NURAM  down         down
GigabitEthernet4   unassigned     YES  NURAM  down         down
Loopback123        unassigned     YES  unset   up           up
Loopback456        10.10.10.1    YES  other   up           up
MoulaliRouter>
```

Put Changing hostname



```
def restconf_put(url, payload):
    full_url = f'{host}{url}'
    response = requests.put(full_url, headers=headers, data=json.dumps(payload), auth=HTTPBasicAuth(username, password), verify=False)
    if response.status_code in [200, 204]:
        print("PUT successful")
    else:
        print(f"PUT failed: {response.status_code}")
        print(response.text)

# Example usage: Update hostname
payload = {
    "Cisco-IOS-XE-native:hostname": "MoualiRouter"
}
restconf_put("/restconf/data/Cisco-IOS-XE-native:hostname", payload)
```

PS C:\Users\Administrator\Desktop\RESTCONF_PROJECT> & C:/Users/Administrator/Desktop/RESTCONF_PROJECT/.venv/Scripts/python.exe "c:/Users/Administrator/Desktop/RESTCONF_PROJECT/PUT Request.py"
PUT successful
PS C:\Users\Administrator\Desktop\RESTCONF_PROJECT> []

Output

```
User Access Verification

Username: moulali
Password:

MoulaliRouter>
*Jun 12 04:41:37.256: %SEC_LOGIN-5-LOGIN_SUCCESS: Login Success [user: moulali]
[Source: LOCAL] [localport: 0] at 04:41:37 UTC Thu Jun 12 2025
```

Overwriting loopback interface



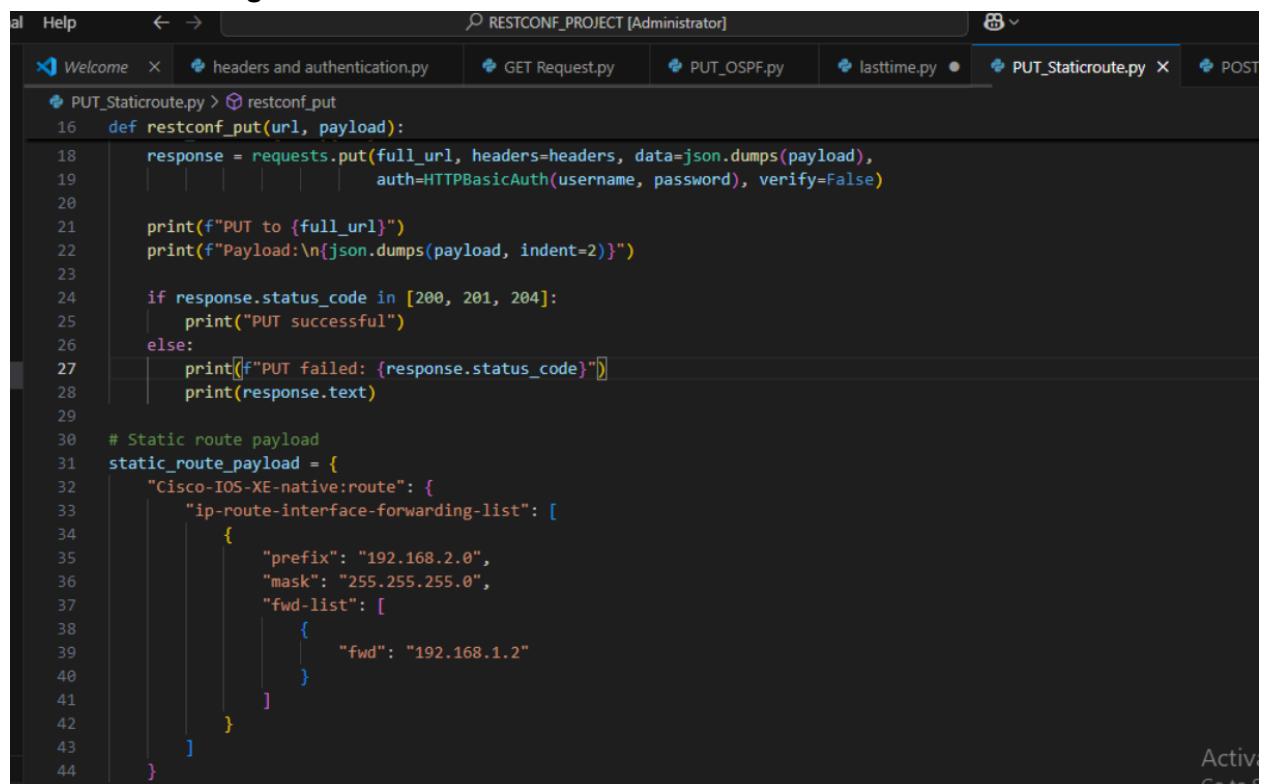
The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows a project named "RESCONF_PROJECT" containing files: .venv, GET Request.py, headers and authentication.py, POST Request.py, PUT Request 2.py, and PUT Request.py.
- Code Editor:** Displays Python code for updating a Loopback interface IP address. The code uses the `restconf` module to make a PUT request to the Cisco IOS-XE native configuration API.
- Terminal:** Shows command-line output from running the script, indicating a successful update of the IP address.
- Status Bar:** Shows the current file is "PUT Request 2.py", with 43 lines of code, 43 characters per line, 79 total characters, 4 spaces, and CRLF line endings.

Output

```
MoulaLiRouter#show ip int brief
Interface          IP-Address      OK? Method Status      Protocol
GigabitEthernet1   172.20.0.39    YES  DHCP   up        up
GigabitEthernet2   192.168.1.1    YES  NVRAM  up        up
GigabitEthernet3   unassigned     YES  NVRAM  down     down
GigabitEthernet4   unassigned     YES  NVRAM  down     down
Loopback123        unassigned     YES  unset   up        up
Loopback456        10.10.10.2    YES  other   up        up
MoulaLiRouter#
```

Static route configuration



The screenshot shows a code editor window with a tab bar at the top labeled "RESTCONF_PROJECT [Administrator]". The main area contains Python code for a "PUT_Staticroute.py" file. The code defines a function "restconf_put(url, payload)" that sends a PUT request to a specified URL with a JSON payload. It prints the full URL and the payload, then checks the response status code. If it's 200, 201, or 204, it prints "PUT successful"; otherwise, it prints "PUT failed" and the response text. The code then creates a static route payload for Cisco IOS XE native routing, defining an interface forwarding list with a single entry for prefix 192.168.2.0 and mask 255.255.255.0, pointing to fwd-list 192.168.1.2.

```
al Help ← → RESTCONF_PROJECT [Administrator]
Welcome × headers and authentication.py GET Request.py PUT_OSPF.py lasttime.py PUT_Staticroute.py POST
◆ PUT_Staticroute.py > restconf.put
16 def restconf_put(url, payload):
17     response = requests.put(full_url, headers=headers, data=json.dumps(payload),
18                             auth=HTTPBasicAuth(username, password), verify=False)
19
20     print(f"PUT to {full_url}")
21     print(f"Payload:\n{json.dumps(payload, indent=2)}")
22
23     if response.status_code in [200, 201, 204]:
24         print("PUT successful")
25     else:
26         print(f"PUT failed: {response.status_code}")
27         print(response.text)
28
29 # Static route payload
30 static_route_payload = {
31     "Cisco-IOS-XE-native:route": [
32         "ip-route-interface-forwarding-list": [
33             {
34                 "prefix": "192.168.2.0",
35                 "mask": "255.255.255.0",
36                 "fwd-list": [
37                     {
38                         "fwd": "192.168.1.2"
39                     }
40                 ]
41             }
42         ]
43     }
44 }
```

Terminal Output

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

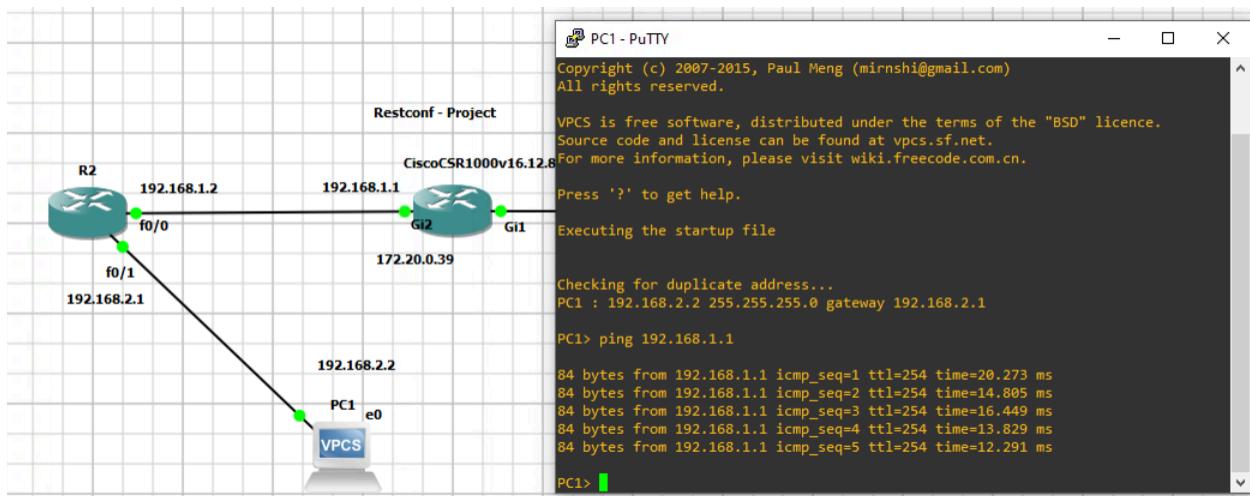
● p/RESCONFIG_PROJECT/PUT_Staticroute.py
PUT to https://172.20.0.39/restconf/data/Cisco-IOS-XE-native:native/ip/route
Payload:
{
    "Cisco-IOS-XE-native:route": [
        {
            "ip-route-interface-forwarding-list": [
                {
                    "prefix": "192.168.2.0",
                    "mask": "255.255.255.0",
                    "fwd-list": [
                        {
                            "fwd": "192.168.1.2"
                        }
                    ]
                }
            ]
        },
        {
            "fwd-list": [
                {
                    "fwd": "192.168.1.2"
                }
            ]
        }
    ],
    {
        "fwd": "192.168.1.2"
    }
]
```

Router output

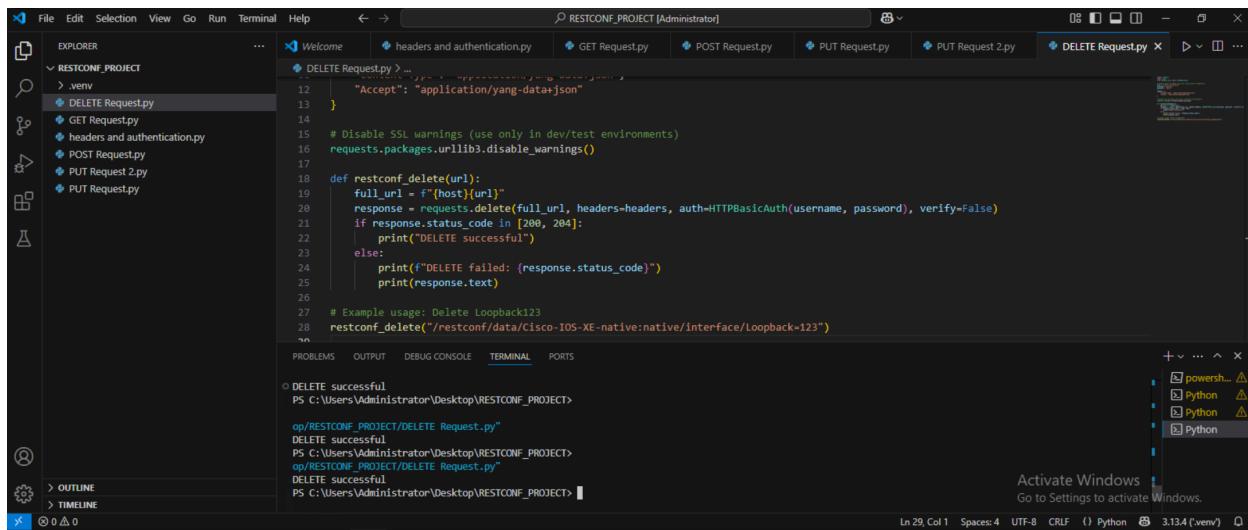
```
VNC QEMU (CiscoCSR1000v16.12.8-1) - TightVNC Viewer
[Icons] [File] [Edit] [Ctrl] [Alt] [Search] [Help] | [Close]

no ip address
negotiation auto
no mop enabled
no mop sysid
!
ip forward-protocol nd
ip http server
ip http authentication local
ip http secure-server
ip http client source-interface GigabitEthernet1
!
ip route 192.168.2.0 255.255.255.0 192.168.1.2
!
!
control-plane
!
--More--
```

ping



Delete



The screenshot shows the VS Code interface with a Python project named 'RESCONF_PROJECT'. The 'DELETE Request.py' file is open in the editor. The code implements a DELETE request to a RESTCONF endpoint, handling responses and printing status messages. The terminal below shows the execution of the script, which successfully deletes an interface.

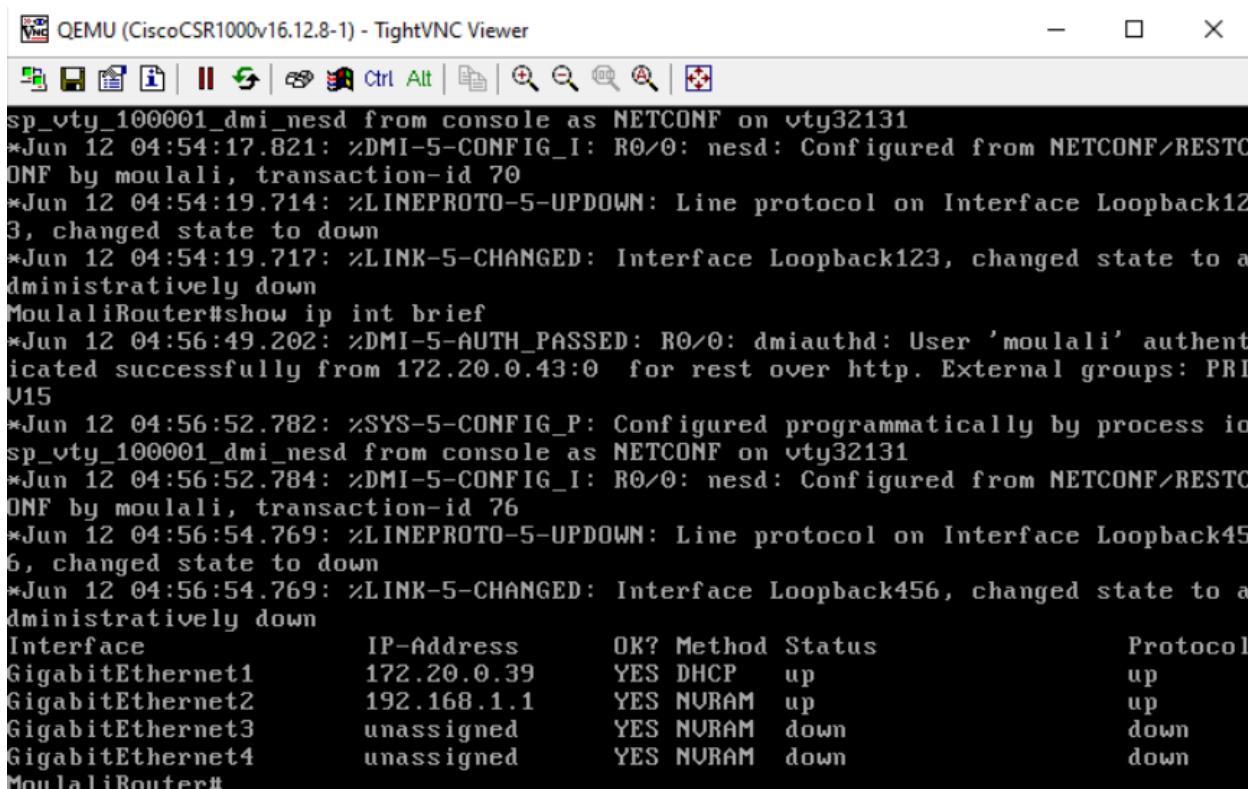
```
12     "Accept": "application/yang-data+json"
13 }
14
15 # Disable SSL warnings (use only in dev/test environments)
16 requests.packages.urllib3.disable_warnings()
17
18 def restconf_delete(url):
19     full_url = f'{host}{url}'
20     response = requests.delete(full_url, headers=headers, auth=HTTPBasicAuth(username, password), verify=False)
21     if response.status_code in [200, 204]:
22         print("DELETE successful")
23     else:
24         print(f"DELETE failed: {response.status_code}")
25         print(response.text)
26
27 # Example usage: Delete Loopback123
28 restconf_delete('/restconf/data/Cisco-IOS-XE-native:native/interface/Loopback-123')
29
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
DELETE successful
PS C:\Users\Administrator\Desktop\RESCONF_PROJECT>
op/RESCONF_PROJECT/DELETE_Request.py*
DELETE successful
PS C:\Users\Administrator\Desktop\RESCONF_PROJECT>
op/RESCONF_PROJECT/DELETE_Request.py*
DELETE successful
PS C:\Users\Administrator\Desktop\RESCONF_PROJECT>
```

Ln 29, Col 1 Spaces: 4 UTF-8 CRLF Python 3.13.4 (.venv)

Output



The screenshot shows the QEMU TightVNC Viewer window displaying the Cisco CSR1000v terminal session. The user has run several commands to configure and verify the state of interfaces. The output includes logs of configuration changes, interface status updates, and a detailed table of interface configurations.

```
sp_vty_100001_dmi_nesd from console as NETCONF on vty32131
*Jun 12 04:54:17.821: %DMI-5-CONFIG_I: R0/0: nesd: Configured from NETCONF/RESTCONF by moulali, transaction-id 70
*Jun 12 04:54:19.714: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback123, changed state to down
*Jun 12 04:54:19.717: %LINK-5-CHANGED: Interface Loopback123, changed state to administratively down
MoulaliRouter#show ip int brief
*Jun 12 04:56:49.202: %DMI-5-AUTH_PASSED: R0/0: dmiauthd: User 'moulali' authenticated successfully from 172.20.0.43:0 for rest over http. External groups: PRI_V15
*Jun 12 04:56:52.782: %SYS-5-CONFIG_P: Configured programmatically by process io
sp_vty_100001_dmi_nesd from console as NETCONF on vty32131
*Jun 12 04:56:52.784: %DMI-5-CONFIG_I: R0/0: nesd: Configured from NETCONF/RESTCONF by moulali, transaction-id 76
*Jun 12 04:56:54.769: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback456, changed state to down
*Jun 12 04:56:54.769: %LINK-5-CHANGED: Interface Loopback456, changed state to administratively down
Interface          IP-Address      OK? Method Status      Protocol
GigabitEthernet1   172.20.0.39    YES  DHCP   up           up
GigabitEthernet2   192.168.1.1    YES  NVRAM  up           up
GigabitEthernet3   unassigned     YES  NVRAM  down        down
GigabitEthernet4   unassigned     YES  NVRAM  down        down
MoulaliRouter#
```