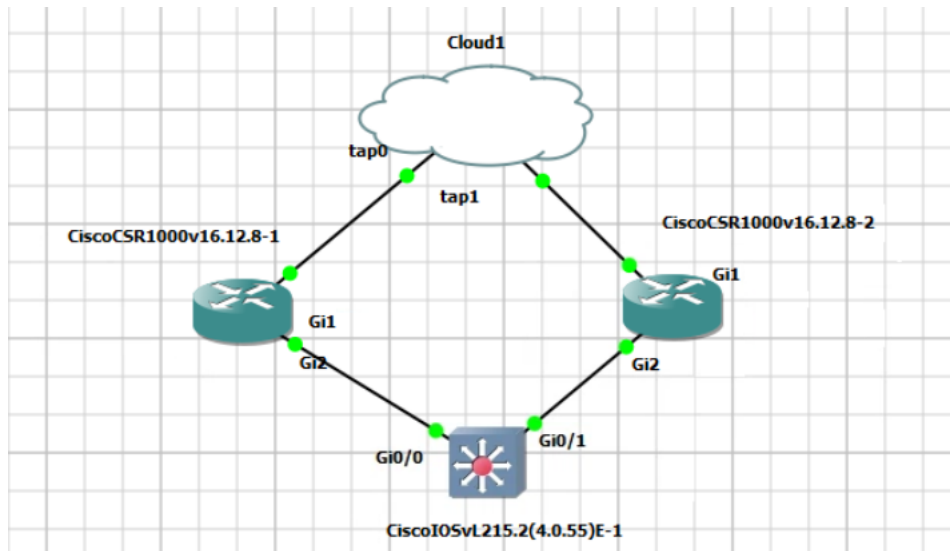


## Enable the NETCONF configuration in router

Build the Network Topology in GNS3 as per below



DHCP will allocate the IP address to the Router

```
Router#show ip int b
Interface                IP-Address      OK? Method Status  Protoco
GigabitEthernet1         172.20.0.67    YES DHCP    up      up
GigabitEthernet2         unassigned      YES unset   down    down
GigabitEthernet3         unassigned      YES unset   down    down
GigabitEthernet4         unassigned      YES unset   down    down
```

## Go to Config Mode:

Conf t

user admin privilege 15 secret cisco123

aaa new-model

aaa authentication login default local

aaa authorization exec default local

Netconf-yang

```

Router(config)#user admin privilege 15 secret cisco123
Router(config)#aaa new-m
Router(config)#aaa new-model
Router(config)#aaa auth
Router(config)#aaa authen
Router(config)#aaa authentication logi
Router(config)#aaa authentication login def
Router(config)#aaa authentication login default loc
Router(config)#aaa authentication login default local
Router(config)#aaa authrori
Router(config)#aaa authrori
Router(config)#aaa authori
Router(config)#aaa authorization ex
Router(config)#aaa authorization exec def
Router(config)#aaa authorization exec default lo
Router(config)#aaa authorization exec default local
Router(config)#net
Router(config)#netco
Router(config)#netconf-y
Router(config)#netconf-yang

```

Show platform software yang-management process

```

Router#show platform software yang-management process
confd          : Running
nesd           : Running
syncfd         : Running
ncsshd         : Running
dmiauthd       : Running
nginx          : Running
ndbmand        : Running
pubd           : Running

```

Go to Command prompt and check the yang connectivity

Ssh admin@IP -p 830 -s netconf -- connect via cmd prompt

```

C:\Users\Administrator>ssh admin@172.20.0.67 -p 830 -s netconf
The authenticity of host '[172.20.0.67]:830 ([172.20.0.67]:830)' can't be established.
RSA key fingerprint is SHA256:MSYPzH3dtKfwdIH5FL1YID1c+uuFEyGdaMwvxAManE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[172.20.0.67]:830' (RSA) to the list of known hosts.
admin@172.20.0.67's password:
Permission denied, please try again.
admin@172.20.0.67's password:
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
<capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
<capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
<capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
<capability>urn:ietf:params:netconf:capability:notification:1.0</capability>
<capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>
<capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=explicit&also-supported=
d</capability>
<capability>urn:ietf:params:netconf:capability:yang-library:1.0?revision=2016-06-21&module-set-id=b1
50b5119785edf</capability>
<capability>http://tail-f.com/ns/netconf/actions/1.0</capability>

```

Add the additional Gi2 interface to router and assign the IP 10.0.0.1

```
Router(config)#int gi 2
Router(config-if)#ip address 10.0.0.1 255.0.0.0
Router(config-if)#no shut
Router(config-if)#
Router(config-if)#
```

Login to YANG Explorer:

<https://172.20.0.11:8088/static/YangExplorer.html> (guest/guest)

YANG Explorer 0.8.0 (Beta)

Build Collections Manage Models

Operations Device Settings

Profile: **ios-xe** [Create device profile](#)

Platform: **csr**

Host: **172.20.0.67** Port: **830**

Username: **admin** Password: **cisco123**

☒ NetConf ☐ RestConf

**RPC** **Python** **YDK** **Capabilities**

Encoding Console

Custom RPC Run Save Clear Copy

Click on capability, router will exchange the capability.

YANG Explorer 0.8.0 (Beta)

Build Collections Manage Models

Operations Device Settings

Profile: **ios-xe** [Create device profile](#)

Platform: **csr**

Host: **172.20.0.67** Port: **830**

Username: **admin** Password: **cisco123**

☒ NetConf ☐ RestConf

**RPC** **Python** **YDK** **Capabilities**

Encoding Console

Custom RPC Run Save Clear Copy

urn:ietf:params:netconf:base:1.0  
urn:ietf:params:netconf:base:1.1  
urn:ietf:params:netconf:capability:interleave:1.0  
urn:ietf:params:netconf:capability:notification:1.0  
urn:ietf:params:netconf:capability:rollback-on-error:1.0  
urn:ietf:params:netconf:capability:validate:1.0  
urn:ietf:params:netconf:capability:validate:1.1  
urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=explicit&also-supported=report-all-tagged  
urn:ietf:params:netconf:capability:writable-running:1.0

Click on Manage model and subscribe the ietf-interface yang model

Yang Explorer 0.8.0 (Beta)

Explorer search Values Operation

Build Collections Manage Models

Workspace Device

Select All Showing 182 models filter

- ☐ ietf-event-notifications@2016-10-27.yang
- ☐ ietf-inet-types@2013-07-15.yang
- ☐ ietf-interfaces-ext.yang
- ☒ ietf-interfaces@2014-05-08.yang [subscribed]
- ☐ ietf-ip@2014-06-16.yang
- ☐ ietf-ipv4-unicast-routing@2015-05-25.yang
- ☐ ietf-ipv6-unicast-routing@2015-05-25.yang
- ☐ ietf-key-chain@2015-02-24.yang
- ☐ ietf-netconf-acm@2012-02-22.yang
- ☐ ietf-netconf-monitoring@2010-10-04.yang
- ☐ ietf-netconf-notifications@2012-02-06.yang
- ☐ ietf-netconf-with-defaults@2011-06-01.yang
- ☐ ietf-netconf@2011-06-01.yang
- ☐ ietf-ospf@2015-03-09.yang
- ☐ ietf-restconf-monitoring@2017-01-26.yang
- ☐ ietf-routing@2015-05-25.yang
- ☐ ietf-yang-library@2016-06-21.yang
- ☐ ietf-yang-push@2016-10-28.yang
- ☐ ietf-yang-smiv2@2012-06-22.yang

Config Oper + Add - Delete Reset Add Subscribe Un-Subscribe Delete Graph

Example 1; Get the Router Interface Name

Yang Explorer 0.8.0 (Beta)

Explorer search Values Operation

Build Collections Manage Models

Operations Device Settings

Profile ios-xe Create device profile

Platform csr

Host 172.20.0.74 Port 830

Username admin Password cisco123

NetConf RestConf RPC Python YDK Capabilities

Encoding Console

```
<interface>
  <name>GigabitEthernet1</name>
</interface>
<interface>
  <name>GigabitEthernet2</name>
</interface>
<interface>
  <name>GigabitEthernet3</name>
</interface>
<interface>
  <name>GigabitEthernet4</name>
</interface>
```

Custom RPC Run Save Clear Copy

## Example 2; Get the Router Interface admin and Operation state

Explorer search

Values

Operation

Build Collections Manage Models

Operations Device Settings

Profile **ios-xe** Create device profile

Platform **csr**

Host 172.20.0.74 Port 830

Username admin Password cisco123

☒ NetConf ☐ RestConf

RPC Python YDK Capabilities

Encoding Console

```
<name>GigabitEthernet3</name>
<admin-status>up</admin-status>
<oper-status>up</oper-status>
</interface>
<interface>
<name>GigabitEthernet4</name>
<admin-status>up</admin-status>
<oper-status>lower-layer-down</oper-status>
</interface>
</interfaces-state>
</data>
</rpc-reply>
```

## Example 3: Run the YANG query and get the Router interface and IP details

Explorer search

Values

Operation

Build Collections Manage Models

Operations Device Settings

Profile **ios-xe** Create device profile

Platform **csr**

Host 172.20.0.67 Port 830

Username admin Password cisco123

☒ NetConf ☐ RestConf

RPC Python YDK Capabilities

Encoding Console

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get>
<filter>
<interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
xmlns:ip="urn:ietf:params:xml:ns:yang:ietf-ip">
<interface>
<name/>
<ip:ipv4>
<ip:address>
<ip:ip/>
</ip:address>
</ip:ipv4>

```

#### Example 4; Run the YANG query and get the Router interface stats

The screenshot displays a network management tool interface with the following components:

- Explorer:** A tree view on the left showing a hierarchy of nodes. The 'interface' node is expanded, showing sub-nodes like 'name', 'type', 'admin-status', 'oper-status', 'last-change', 'if-index', 'phys-address', 'higher-layer-if', 'lower-layer-if', 'speed', and 'statistics'. The 'statistics' node is also expanded, showing sub-nodes like 'discontinuity-time', 'in-octets', 'in-unicast-pkts', 'in-broadcast-pkts', 'in-multicast-pkts', 'in-discards', and 'in-errors'. The 'in-octets' node is selected, and its value is '<get>'.
- Values:** A column in the middle showing the values for the selected nodes. The value for 'in-octets' is '<get>'.
- Operation:** A column on the right showing the operation for the selected nodes. The operation for 'in-octets' is '<get>'.
- Build:** A tab at the top right with sub-tabs 'Operations' and 'Device Settings'. The 'Device Settings' tab is active.
- Device Settings:** A form for configuring device settings. It includes fields for 'Profile' (ios-xe), 'Platform' (csr), 'Host' (172.20.0.74), 'Port' (830), 'Username' (admin), and 'Password' (cisco123). There are also buttons for 'Create device profile', 'RPC', 'Python', 'YDK', and 'Capabilities'.
- Console:** A text area at the bottom right showing the output of the YANG query. The output is a JSON object representing the router interface statistics.

The console output is as follows:

```
</statistics>
</interface>
<interface>
  <name>GigabitEthernet3</name>
  <statistics>
    <in-octets>1351626</in-octets>
    <out-octets>1346434</out-octets>
  </statistics>
</interface>
<interface>
  <name>GigabitEthernet4</name>
  <statistics>
```

### **Other Example**

Run the custom query for below :

- 1: Get the router hostname
- 2: Change the Router hostname
- 3: Change the interface operational status
- 4: Check the IP address
- 5: Assign the Ip address on interface
- 6: configure the loopback IP address
- 7: Attempt to create new loopback interface with same IP address
- 8: Delete the Loopback ip address
- 9: Delete the loopback interface
- 10: Get Complete config
- 11: Get the filtered configuration
- 12: Enable Candidate data store
- 13: make the changes on candidate data store
- 14: Commit the changes on candidate
- 15: Copy configuration from running to candidate
- 16: Close the session

**Attempt to create a new loopback interface with the same IPv4 address and get the Error from router**

Username  Password

☒ NetConf ☐ RestConf

Encoding

```
<nc:error-tag>invalid-value</nc:error-tag>
<nc:error-severity>error</nc:error-severity>
<nc:error-message lang="en" xmlns="http://www.w3.org/XML/1998/
namespace">inconsistent value: Device refused one or more commands</
nc:error-message>
<nc:error-info>
  <severity xmlns="http://cisco.com/yang/cisco-ia">error_cli</
severity>
  <detail xmlns="http://cisco.com/yang/cisco-ia">
    <bad-cli>
      <bad-command>ip address 10.1.1.1 255.255.255.0</bad-command>
    </bad-cli>
  </detail>
</nc:error-info>
</nc:error-tag>
```

**Delete the Loopback IP address**

☒ NetConf ☐ RestConf

Encoding

```
<interface>
  <Loopback>
    <name>4</name>
    <ip>
      <address operation='delete'>
        <primary>
          <address>14.1.1.1</address>
          <mask>255.255.255.0</mask>
        </primary>
      </address>
    </ip>
  </Loopback>
</interface>
```

☒ Custom RPC

**Delete the Loopback interface**

☒ NetConf ☐ RestConf

Encoding

```
<edit-config>
  <config>
    <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
      <interface>
        <Loopback operation='delete'>
          <name>1</name>
        </Loopback>
      </interface>
    </native>
  </config>
</edit-config>
</rpc>
```

☒ Custom RPC



## Assign the IP address on Gi2 interface



## Enable Candidate Data Store Capability

Conf t

user admin privilege 15 secret cisco123

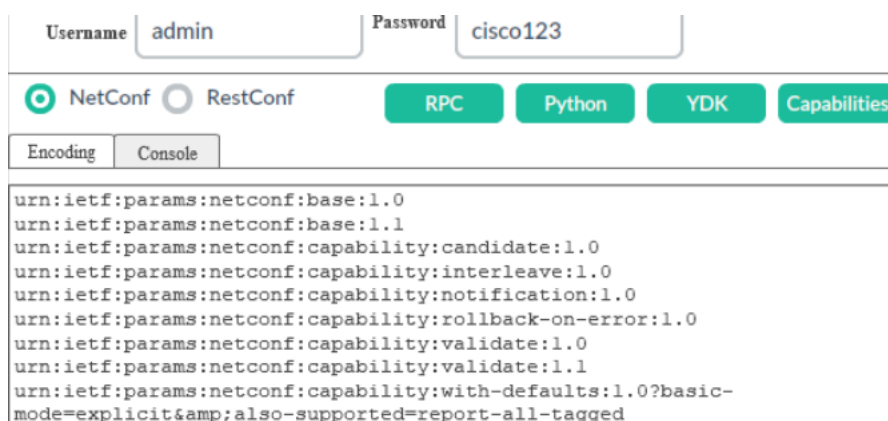
aaa new-model

aaa authentication login default local

aaa authorization exec default local

Netconf-yang feature candidate-datastore

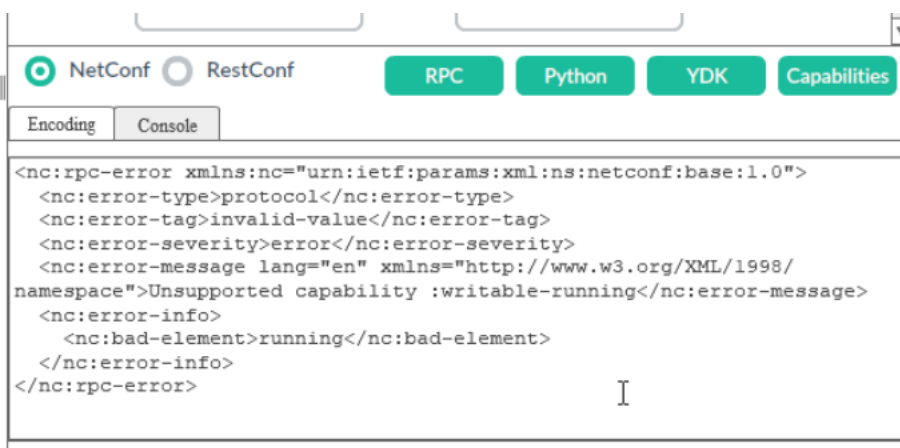
Netconf-yang



## Get the Configuration from Candidate data store



Change the interface status to down (Gi2) in running data store --- **Error as no writable running capability**



## Change the interface status to down (Gi2) in candidate data store



The screenshot shows the NetConf console interface. At the top, there are tabs for 'NetConf' (selected) and 'RestConf'. Below these are buttons for 'RPC', 'Python', 'YDK', and 'Capabilities'. The main area is a text editor with 'Encoding' and 'Console' tabs. The text in the editor is as follows:

```
<candidate/>
</target>
<config>
  <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
    <interface>
      <name>GigabitEthernet2</name>
      <enabled>false</enabled>
    </interface>
  </interfaces>
</config>
</edit-config>
</rpc>
```

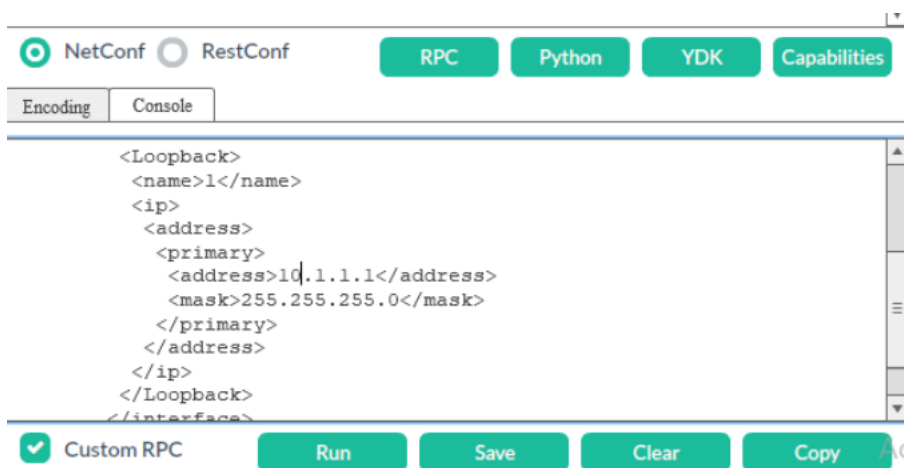
## Commit the changes



The screenshot shows the NetConf console interface. At the top, there are tabs for 'NetConf' (selected) and 'RestConf'. Below these are buttons for 'RPC', 'Python', 'YDK', and 'Capabilities'. The main area is a text editor with 'Encoding' and 'Console' tabs. The text in the editor is as follows:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <commit></commit>
</rpc>
```

## Create the Loopback interface on candidate data store



The screenshot shows the NetConf console interface. At the top, there are tabs for 'NetConf' (selected) and 'RestConf'. Below these are buttons for 'RPC', 'Python', 'YDK', and 'Capabilities'. The main area is a text editor with 'Encoding' and 'Console' tabs. The text in the editor is as follows:

```
<Loopback>
  <name>l</name>
  <ip>
    <address>
      <primary>
        <address>10.1.1.1</address>
        <mask>255.255.255.0</mask>
      </primary>
    </address>
  </ip>
</Loopback>
</interface>
```

At the bottom of the console, there is a 'Custom RPC' checkbox which is checked, and buttons for 'Run', 'Save', 'Clear', and 'Copy'.

## Commit the changes



## Change the router Hostname on candidate data store and commit the change



## Copy the configuration from running data store to candidate data store

**Step 1 :** Create the new Loopback interface on candidate data store

**Step 2 :** Copy the running data store to candidate data store

**Step 3 :** Commit the changes on candidate data store

**Step 4 :** We observe new Loopback interface are not created on running data st



The screenshot shows a web-based NetConf interface. At the top, there are radio buttons for 'NetConf' (selected) and 'RestConf'. To the right are buttons for 'RPC', 'Python', 'YDK', and 'Capabilities'. Below these are tabs for 'Encoding' and 'Console'. The 'Console' tab is active, displaying an XML RPC request in a text area. The request is as follows:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <copy-config>
    <target>
      <candidate/>
    </target>
    <source>
      <running/>
    </source>
  </copy-config>
</rpc>
```

## RestConf:

Enable the router with Restconf capability

Conf t (Config mode)

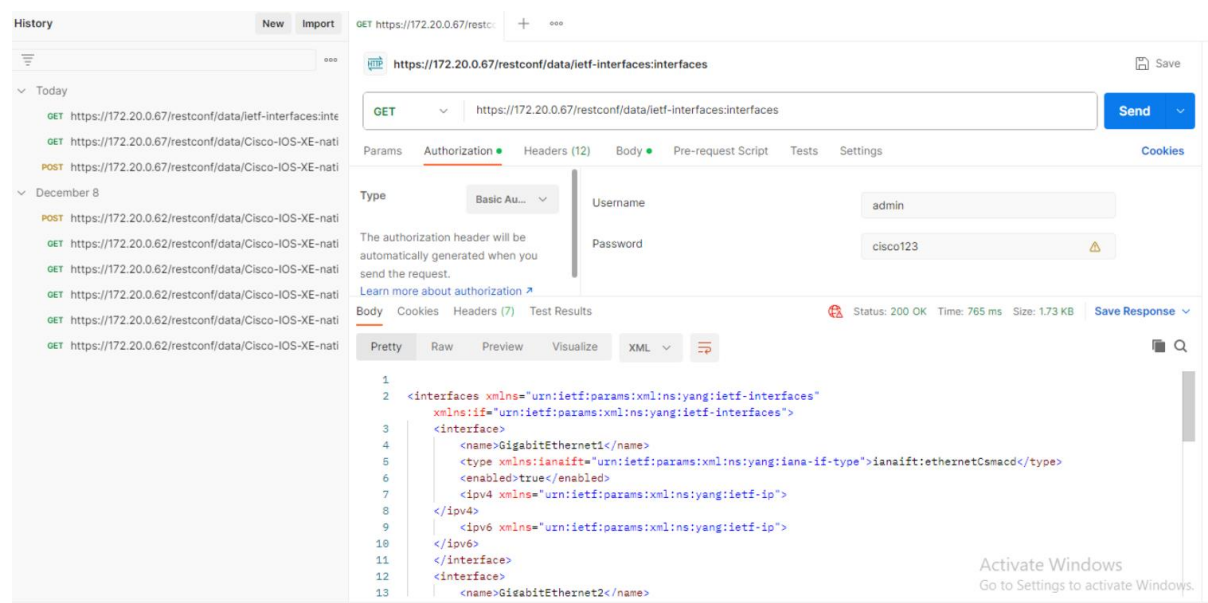
Restconf

Ip http secure-server

## Login to Postman:

Example : Get the Router interface details

**GET : <https://IP address of server/restconf/data/ietf-interfaces:interfaces>**



GET
https://172.20.0.74/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet2
Send

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

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Body
Cookies
Headers (7)
Test Results

Status: 200 OK
Time: 808 ms
Size: 759 B
Save Response

Pretty
Raw
Preview
Visualize
XML

```

1
2 <interface xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
   xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">
3   <name>GigabitEthernet2</name>
4   <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
5   <enabled>true</enabled>
6   <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
7     <address>
8       <ip>192.168.2.1</ip>
9       <netmask>255.255.255.0</netmask>
10    </address>
11  </ipv4>
12  <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
13    </ipv6>

```

## Get Config

GET
https://172.20.0.74/restconf/data/Cisco-IOS-XE-native:native/
Send

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

Type
Basic Auth

The authorization header will be automatically generated when you send the request.  
[Learn more about authorization](#)

Username
admin
Password
cisco123

Body
Cookies
Headers (7)
Test Results

Status: 200 OK
Time: 6.69 s
Size: 6.08 KB
Save Response

Pretty
Raw
Preview
Visualize
XML

```

106
107
108
109
110
111
...

```

## Example :

Create the new Loopback interface on router via CLI

Conf t

Int loopback1

Ip address 11.0.0.1 255.0.0.0

GET : <https://IP address of server/restconf/data/ietf-interfaces:interfaces/interface=Loopback1>

The screenshot displays a REST client interface with the following details:

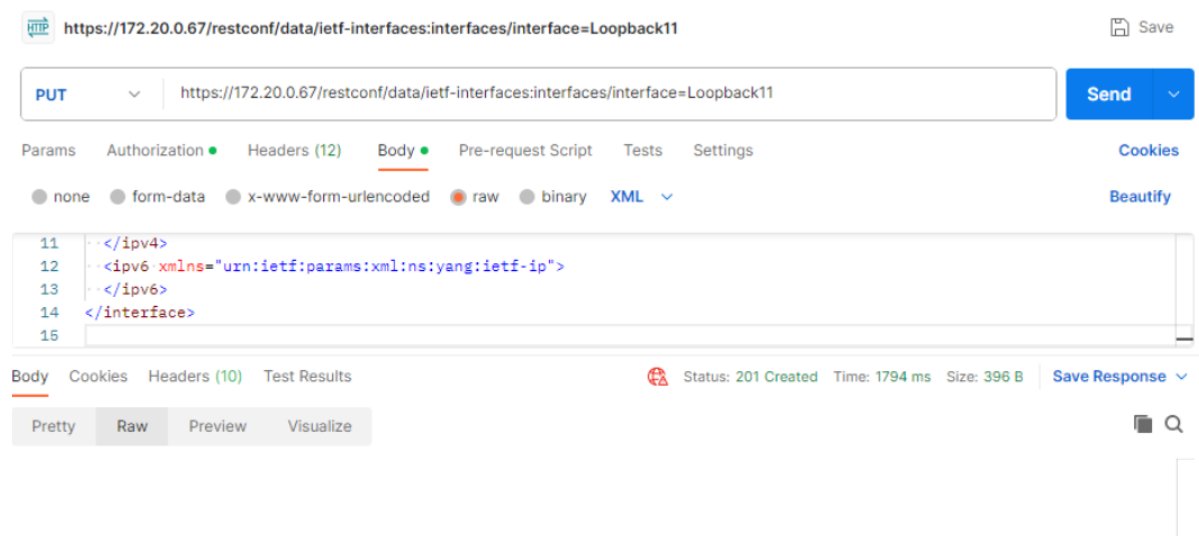
- URL:** `https://172.20.0.67/restconf/data/ietf-interfaces:interfaces/interface=Loopback11`
- Method:** PUT
- Body:** XML content:

```
11 </ipv4>
12 <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
13 </ipv6>
14 </interface>
15
```
- Status:** 201 Created
- Time:** 1794 ms
- Size:** 396 B
- Buttons:** Save, Send, Beautify, Save Response, Pretty, Raw, Preview, Visualize.

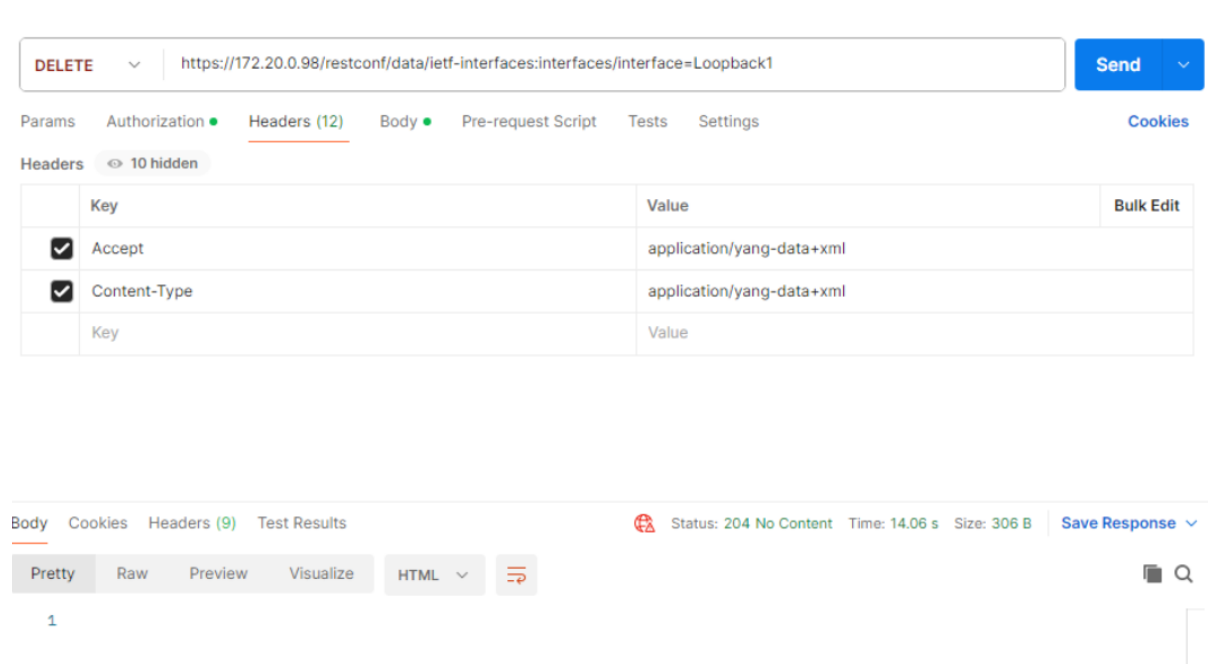


## Example : Create the Loopback interface via Postman

PUT : <https://IP address of server/restconf/data/ietf-interfaces:interfaces/interface=Loopback11>



## Delete the Loopback interface.



## Change the Interface status to down for GigabitEthernet2 interface

The screenshot shows a REST client interface with a PUT request to `https://172.20.0.98/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet2`. The request body is an XML document that sets the `enabled` attribute to `false` for the `GigabitEthernet2` interface. The response status is 204 No Content.

```
PUT https://172.20.0.98/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet2
```

Params Authorization Headers (12) Body Pre-request Script Tests Settings Cookies Beautify

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

```
3 <name>GigabitEthernet2</name>
4 <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
5 <enabled>false</enabled>
6 <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
7   <address>
8     <ip>192.168.1.1</ip>
9     <netmask>255.255.0.0</netmask>
10  </address>
11 </ipv4>
12 <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
13 </ipv6>
14 </interface>
```

Body Cookies Headers (9) Test Results Status: 204 No Content Time: 2.13 s Size: 306 B Save Response

Pretty Raw Preview Visualize HTML

## Change the Interface status to UP for GigabitEthernet2 interface

The screenshot shows a REST client interface with a PUT request to `https://172.20.0.98/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet2`. The request body is an XML document that sets the `enabled` attribute to `true` for the `GigabitEthernet2` interface. The response status is 204 No Content.

```
PUT https://172.20.0.98/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet2
```

Params Authorization Headers (12) Body Pre-request Script Tests Settings Cookies Beautify

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

```
3 <name>GigabitEthernet2</name>
4 <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
5 <enabled>true</enabled>
6 <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
7   <address>
8     <ip>192.168.1.1</ip>
9     <netmask>255.255.0.0</netmask>
10  </address>
11 </ipv4>
12 <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
13 </ipv6>
14 </interface>
```

Body Cookies Headers (9) Test Results Status: 204 No Content Time: 1525 ms Size: 306 B Save Response

Pretty Raw Preview Visualize HTML

## Assign the interface IP address

PUT ▼ <https://172.20.0.98/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet4> Send ▼

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

● none ● form-data ● x-www-form-urlencoded ● raw ● binary XML ▼ Beautify

```
3 <<name>GigabitEthernet4</name>
4 <<type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
5 <<enabled>true</enabled>
6 <<ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
7 <<address>
8 <<ip>172.168.1.1</ip>
9 <<netmask>255.255.0.0</netmask>
10 <</address>
11 <</ipv4>
12 <<ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
13 <</ipv6>
14 <</interface>
```

Body Cookies Headers (9) Test Results 🔗 Status: 204 No Content Time: 1471 ms Size: 306 B Save Response ▼

Pretty Raw Preview Visualize HTML ▼ 🔍

## Remove the Interface IP address

PUT ▼ <https://172.20.0.98/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet4> Send ▼

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

● none ● form-data ● x-www-form-urlencoded ● raw ● binary XML ▼ Beautify

```
1
2 <<interface xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
3   xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">
4   <<name>GigabitEthernet4</name>
5   <<type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
6   <<enabled>false</enabled>
7   <<ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
8   <</ipv4>
9   <<ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
10  <</ipv6>
11 <</interface>
```

Body Cookies Headers (9) Test Results 🔗 Status: 204 No Content Time: 2.92 s Size: 305 B Save Response ▼

Pretty Raw Preview Visualize HTML ▼ 🔍