

Discovery 10: Use NSO RESTCONF API with Postman



Introduction

In this activity, you will learn how to use the NSO RESTCONF API. After completing this activity, you will be able to meet these objectives:

- Learn how to use Postman
- Obtain device information using the GET method
- List service instances, using the GET method
- Create service instances, using the POST method
- Delete service instances, using the DELETE method

Required Resources

The following resources and equipment are required for completing the activities in this lab guide:

- PC or laptop with a web browser
- Access to the internet

Command Syntax Reference

This lab guide uses the following conventions for command syntax:

Formatting	Description and Examples
show running config	Commands in steps use this formatting.
<i>Example</i>	Type show running config
<i>Example</i>	Use the name command.
<code>show running config</code>	Commands in CLI outputs and configurations use this formatting.
highlight	CLI output that is important is highlighted.
<i>Example</i>	<pre>student@student-vm:~\$ ncs --version 5.8.2.1</pre>
<i>Example</i>	Save your current configuration as the default startup config . <pre>Router Name# copy running startup</pre>
brackets ([])	Indicates the optional element. You can choose one of the options.
<i>Example:</i>	<pre>(config-if)# frame-relay lmi-type {ansi cisco q933a}</pre>
<i>italics font</i>	Arguments for which you supply values.
<i>Example</i>	Open file ip tcp window-size bytes
angle brackets (<>)	In contexts that do not allow italics, arguments for which you supply values are enclosed in angle brackets [<>]. Do not type the brackets when entering the command.
<i>Example</i>	If the command syntax is ping <ip_address> , you enter ping 192.32.10.12
string	A nonquoted set of characters. Type the characters as-is.
<i>Example</i>	(config)# hostname MyRouter
vertical line ()	Indicates that you enter one of the choices. The vertical line separates choices. Do not type the vertical line when entering the command.
<i>Example</i>	If the command syntax is show ip route arp , you enter either show ip route or show ip arp , but not both.

Command List

The following are the most common commands that you will need:

Linux shell:

Command	Comment
source /home/student/nso-5.8/ncsrc	Source NSO environmental variables.
ls ll	Display contents of the current directory.
cd	Move directly to user home directory.
cd ..	Exit the current directory.
cd test	Move into folder "test," which is a subfolder of the current directory.
cd /home/student/test	Move into folder "test" by specifying the direct path to it, starting from the root of the directory system.
ncs_cli -Cu admin	Log in to NSO CLI directly from the local server.

NSO CLI:

Command	Comment
switch cli	Change CLI style.
show ?	Display all command options for current mode.
configure	Enter configuration mode.
commit	Commit new configuration (configuration mode only command).
show configuration	Display new configuration that has not yet been committed (configuration mode only command).

Job Aids

The following job aids are available to help you complete the lab activities:

- This lab guide
- Student guide, for general explanations

RESTCONF URI Structure

URIs represent RESTCONF API resources. Take a closer look at the following URI structure and description of every building block:

SCHEME://ADDRESS/ROOT/STORE:[YANG MODULE:]CONTAINER/LEAF[?QUERY]

- SCHEME–HTTPS for encrypted communication and HTTP for cleartext.
- ADDRESS–NSO server IP address or hostname.
- ROOT–The root resource of the RESTCONF API.
- STORE–Datastore resource representation.
- [YANG MODULE:]CONTAINER–Optional YANG module name and container.
- LEAF–Individual element defined inside selected CONTAINER.
- [?QUERY]–Optional query parameters that affect the returned results.

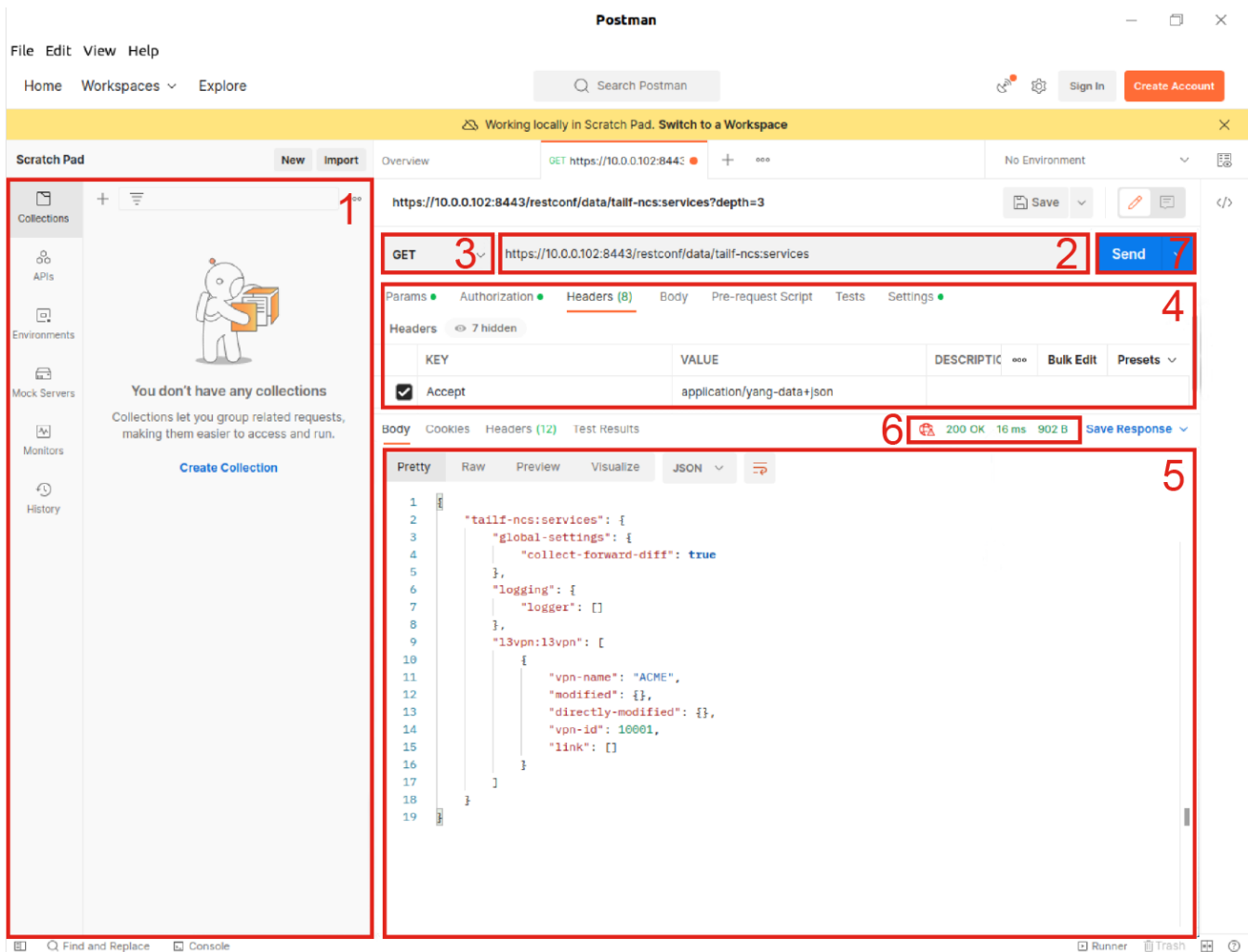
The RESTCONF protocol uses HTTP methods for CRUD operations on the selected resource. The following table shows how some of the RESTCONF operations relate to NETCONF operations:

RESTCONF	NETCONF
GET	<get-config>, <get>
POST	<edit-config> (operation = "create")
PUT	<edit-config> (operation = "replace")
PATCH	<edit-config> (operation = "merge")
DELETE	<edit-config> (operation = "delete")

Postman Interface

Postman is a set of tools designed to help developers, and others explore and develop API interfaces. In this lab, we will use Postman to initiate a RESTCONF request and analyze responses, but there is much more to Postman than simply exploring RESTCONF APIs. This topic will familiarize you with the basic layout and functions of Postman.

1. History and Collections tabs: Historical overview of your recent requests and collections of your saved requests and responses for later reference.
2. Request URL: Target address for this request. It follows the RESTCONF URI structure described earlier.
3. Type of request: Various types of RESTCONF requests will result in various options in the request parameters.
4. Request parameters or headers: (Optional) Request parameters are sent with the request and are used to input information into the request if required. For example, creating a service using a POST request requires all the necessary parameters for a service instance.
5. Response body: Response of the current request.
6. HTTP Response status and code: Example: 200 OK.
7. Send button. Sends the request.



Job Aids for Task 1

Use *https* for the scheme because the server uses TLS for encrypt the communication.

The root of the RESTCONF API resource is located at */restconf* and must be present in every request.

Root resource of the RESTCONF API can be discovered by sending a GET request to the well-known URI at */well-known/host-meta*.

The following table contains credentials for successfully completing this task.

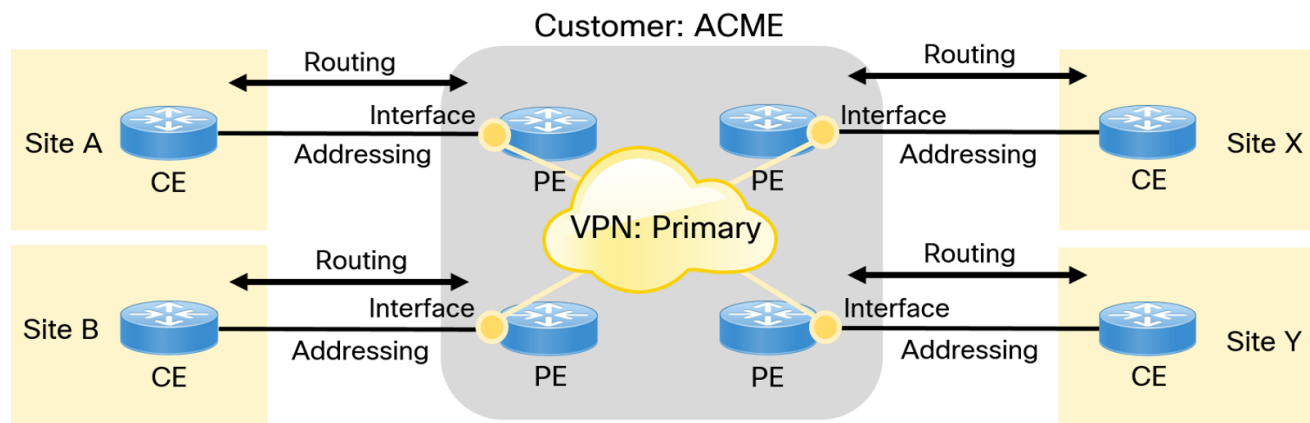
Device	Username	Password
RESTCONF API	admin	admin

The following table contains IP addresses used in this task.

Device	IP	Port
RESTCONF API	10.0.0.102	8443

Job Aids for Task 4

The following data will be required for this task.



The following table describes the l3vpn YANG service model characteristics and requirements. Use this information as a starting point for creating a new service instance.

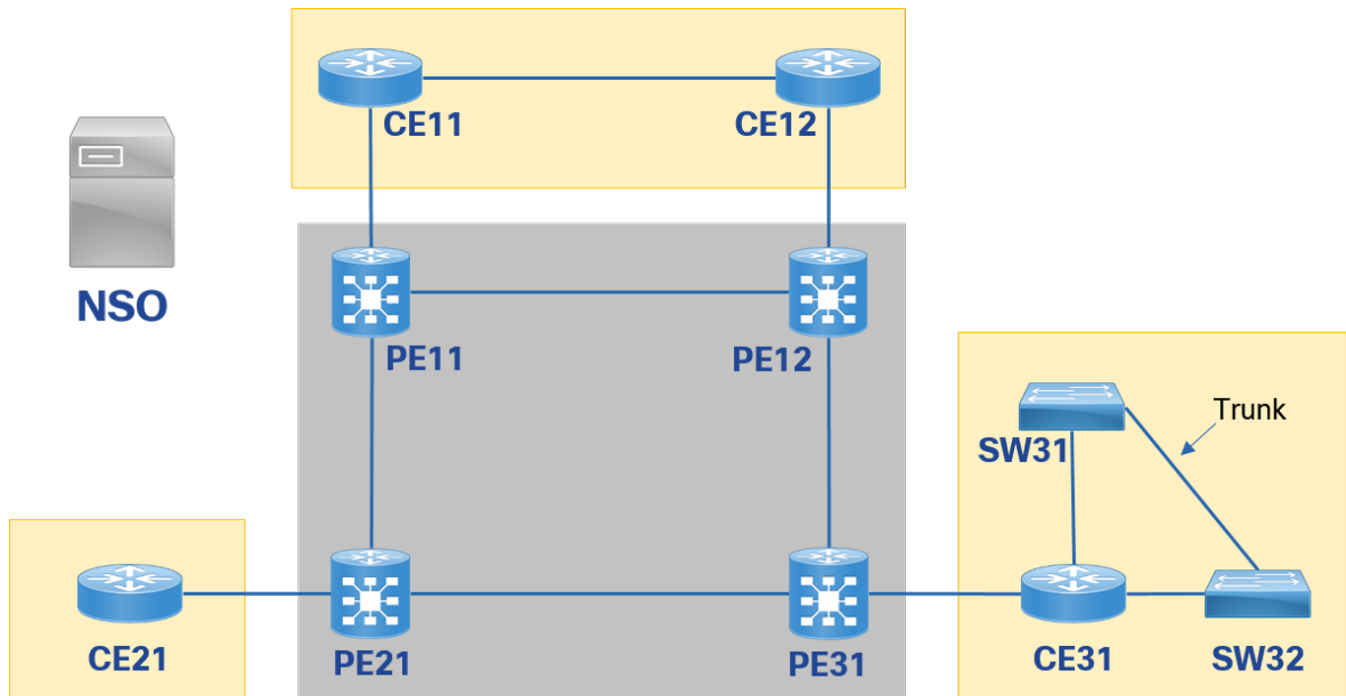
Attribute	Name	Data Type	Restriction	Other
Service name	l3vpn	list	—	—
VPN ID	vpn-id	uint32	Range:10001–19999	key for list <i>l3vpn</i>
VPN name	vpn-name	string	—	—
Customer	customer	leafref	—	reference to NSO customers
Link	link	list	—	—
Link ID	link-id	uint32	Range: 1–255	key for list <i>link</i>
Interface	interface	string	—	<i>pe-device</i> and <i>interface</i> combination must be unique
Routing protocol	routing-protocol	enumeration	Options: <i>bgp</i> <i>static</i>	—
PE device	pe-device	leafref	Only PE devices might be selected	<i>pe-device</i> and <i>interface</i> combination must be unique
Static route	static-route	list	Applicable only when static routing is selected	—
Prefix	prefix	inet:ipv4-address	—	key for list <i>static-route</i>
Mask	mask	union of inet:ipv4-address and inet:ipv6address	—	—

Lab Topology Information

Your lab session is your personal sandbox. Whatever you do in your session will not be reflected in anyone else's session. There are two topologies. The general one is your lab environment, which has your workstation and a Linux server. On the Linux server within that topology is your second topology with your NSO installation, together with numerous Netsim routers and switches that are logically grouped into a network topology. This network will be the one that you will orchestrate with your NSO.

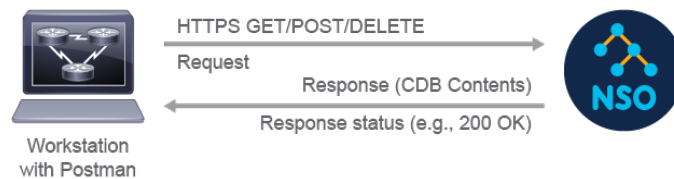
- The network topology is designed to cover both the service provider and enterprise use cases. It is a simulated Netsim network; devices have no control or data plane. Devices will, however, accept or reject a configuration sent by the NSO, just as real devices would.

Topology



Visual Objective

The figure illustrates the topology of this exercise. The goal is to initiate a RESTCONF communication between the client (Postman running on a workstation) and the server (Student-VM).



Task 1: Perform a Basic GET in Postman

The first task requires you to start and test the basic functionality of Postman. You will use a GET method to list the available datastore resources.



The final solutions for all labs, including this one, are located in the ~/solutions directory. You can use them for copy-pasting longer pieces of code and as a reference point for troubleshooting your packages.

Activity

Complete these steps:

Step 1

Connect to the NSO server by clicking on the NSO icon.

Step 2

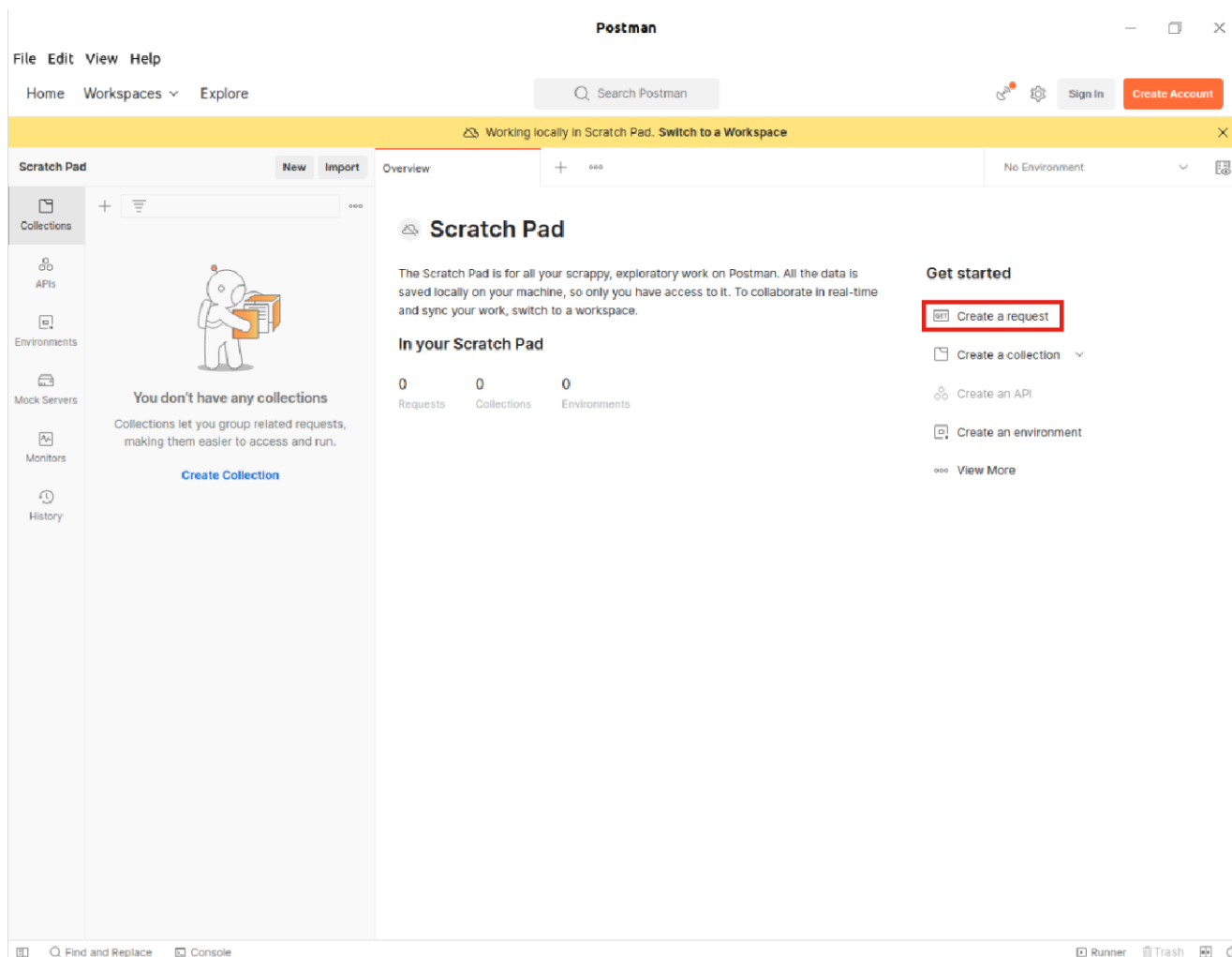
Open the Postman application by clicking the icon in the taskbar.



Step 3

In Postman, create a new request.

Click **Create a request** at the **Get started** section of the application.

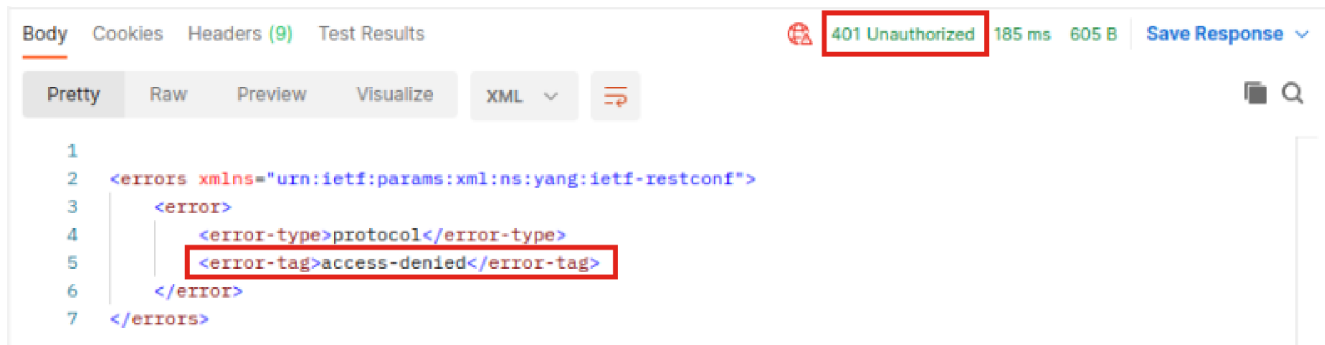
**Step 4**

Make a **GET** request to the local Cisco NSO RESTCONF API.

Choose **GET** as the request type. In the request URL, enter the local Cisco NSO API address **https://10.0.0.102:8443/restconf** and click **Send** or press **Enter**.

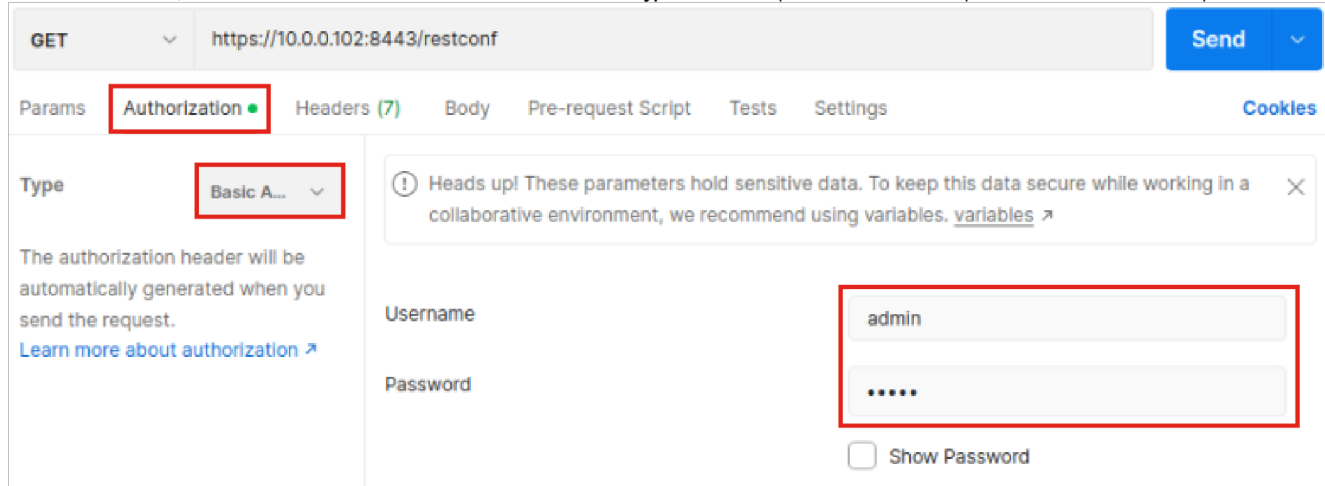


Verify that the RESTCONF request is unsuccessful because no user authorization attaches to the request. In the **HTTP Response status**, observe the **401 Unauthorized** code. In the **Response body** section, observe the **access-denied** message.

**Step 5**

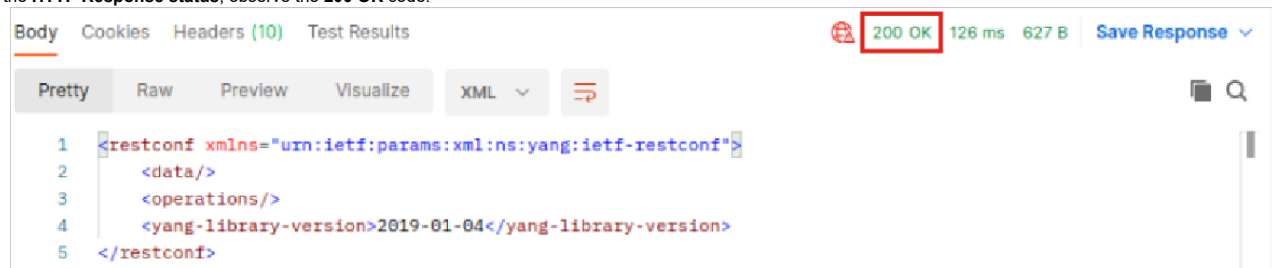
Complete the request with authorization data by using the **admin** username and the **admin** password. Use basic authorization and repeat the GET request.

In the headers section, click the **Authorization** tab and choose **Basic Auth** in the **Type** section. Complete the username and password fields. Click **Send** or press **Enter**.

**Step 6**

Verify that the GET RESTCONF request is successful with a **200 OK** response. The response body includes some basic information about the Cisco NSO instance.

In the **HTTP Response status**, observe the **200 OK** code.



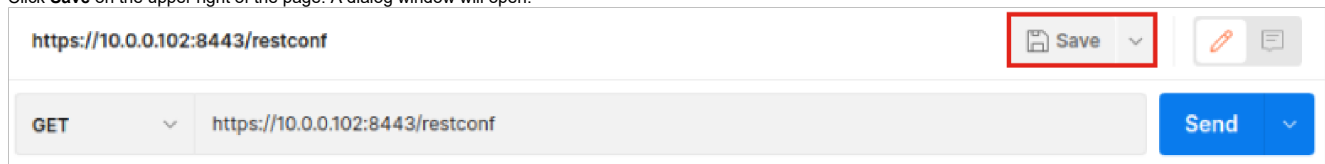
NSO RESTCONF API uses Basic Authentication to secure the communication between the client and the server. Basic access authentication is a method for an HTTP user agent to provide a username and password when making a request.

It is also possible to enable token-based authentication. With token authentication the client sends a **User:Password** to the RESTCONF server, which will perform the authentication and upon success produce a token that the client can use in subsequent requests.

Step 7

Save the request to a new collection.

Click **Save** on the upper right of the page. A dialog window will open.



Name it *Basic request*. You can optionally enter a request description.

SAVE REQUEST


Request name

Basic request

[Add description](#)

Save to

Select a collection/folder



You don't have any collections.
Collections let you group related requests, making them easier to access and run.
[Create a collection](#)

New Collection

Cancel

Save

Create a new collection called *NSO Lab* and click the **Create** button to save.

SAVE REQUEST

Request name

Basic request

Add description

Save to Select a collection/folder

Search for collection or folder



NSO Lab

Create

Cancel

Activity Verification

You have completed this task when you attain this result:

- You received a 200 OK response from the Cisco NSO API when you used Postman to make a GET RESTCONF request.

Task 2: Use the GET Method to Obtain Device Information

This task requires that you obtain a list of all devices and then obtain information about a specific device.

Activity

Complete these steps:

Step 1

Input the correct URI for all devices currently present in the running CDB data store.

- Use the unified datastore by adding `/data` to the base API URL.
- Select all devices by adding the YANG module and the container `/tailf-ncs:devices` to the request URL.

Make sure that the request URL matches the value **https://10.0.0.102:8443/restconf/data/tailf-ncs:devices/**.

Step 2

Click **Send**.

The response should be a **"200 OK"** success message with the following body:

```
<devices xmlns="http://tail-f.com/ns/ncs" xmlns:ncs="http://tail-f.com/ns/ncs">
  <global-settings>
    <trace-dir>./logs</trace-dir>
  </global-settings>
  <authgroups>
    <group>
      <name>default</name>
      <umap>
        <local-user>admin</local-user>
        <remote-name>admin</remote-name>
        <remote-password>$9$NSfg4Fbybm+w/2fcvVIHY8mES6nx5k0s9YPAA67viyo=</remote-password>
      </umap>
      <umap>
        <local-user>oper</local-user>
        <remote-name>oper</remote-name>
        <remote-password>$9$ZY7Iw2y071fQK1TqxsWJIUrwmgNav3vxtv1+2B3bVEo=</remote-password>
      </umap>
    </group>
    <snmp-group>
      ...
    </snmp-group>
  </authgroups>
  <device-group>
    <name>All devices</name>
    <device-group>CE routers</device-group>
    <device-group>PE routers</device-group>
    <device-group>Switches</device-group>
    <member>CE11</member>
    <member>CE12</member>
  </device-group>
</devices>
```

```

    <member>CE21</member>
    ...
  </device-group>
  ...
  <session-pool>
    <pooled-device>
      <device>CE11</device>
      <device-type>cli</device-type>
      <sessions>1</sessions>
      <max-sessions>unlimited</max-sessions>
      <idle-time>30</idle-time>
    </pooled-device>
    <pooled-device>
      <device>CE12</device>
      <device-type>cli</device-type>
      <sessions>1</sessions>
      <max-sessions>unlimited</max-sessions>
      <idle-time>30</idle-time>
    </pooled-device>
    <pooled-device>
      <device>CE21</device>
      <device-type>cli</device-type>
      <sessions>1</sessions>
      <max-sessions>unlimited</max-sessions>
      <idle-time>30</idle-time>
    </pooled-device>
  </session-pool>
</devices>

```

The output is deep and verbose. It contains all the information about device groups, devices, and so on, currently present in the data store of the CDB. It also takes considerable time to get this data from NSO. Currently, you are interested only in the names of configured devices so that you will filter the output by device names.



Besides XML, RESTCONF also supports JSON encoded data. It is considered more human readable and is natively supported in Python so you will change to this type in the Headers area of Postman.

Step 3

Switch to the JSON format.

- Click the **Headers** tab.
- Under KEY, enter **Accept**, and under VALUE enter **application/yang-data+json**.

Step 4

To filter the output by device names, edit the request URL as follows:

- Select the device list by adding **/device**.
- Filter the output based on names by specifying the **?fields=name** query parameter after the device.

Make sure that the request URL matches the value **https://10.0.0.102:8443/restconf/data/taif-ncs:devices/device/?fields=name**.

The screenshot shows the Postman interface for a RESTCONF request. The request method is GET, and the URL is `https://10.0.0.102:8443/restconf/data/taif-ncs:devices/device/?fields=name`. The Headers tab is selected, showing a header with KEY `Accept` and VALUE `application/yang-data+json`. The Body tab is also selected, showing the JSON response in Pretty format. The response is a list of device names: `["taif-ncs:device": [{"name": "CE11"}, {"name": "CE12"}, {"name": "CE21"}, {"name": "CE31"}, {"name": "PE11"}, {"name": "PE12"}, {"name": "PE21"}]]`.

The response was much quicker, and the list of devices is more readable. However, it does not contain information about those devices. After obtaining a list of devices, obtaining information about a specific device is common.



The depth query parameter limits the depth of subtrees that the server returns. Data nodes with a value greater than the depth parameter are not returned in response to a GET request. The value of the depth parameter is either an integer between 1 and 65535 or the string "unbounded." The default value is "unbounded."

Step 5

Create a new request to obtain information about the device PE11.

- Select the device PE11 by altering **/device=PE11** in the request URL.

Make sure that the request URL matches the value **https://10.0.0.102:8443/restconf/data/tailf-ncs:devices/device=PE11**.

Step 6

Click **Send**.

The result should be a **"200 OK"** status with the following body:

```
{
  "tailf-ncs:device": [
    {
      "name": "PE11",
      "address": "127.0.0.1",
      "port": 10022,
      "ssh": {
        "host-key": [
          {
            "algorithm": "ssh-rsa",
            "key-data": "AAAAAB3NzaC1yc2EAAAADAQABAAQDP9GHpS2jFEQzrErWl7+PWY80pz3C+T+dfBgAyHsgc\nDWEqYdCvmE9HctV0169hNquTkQ6ABG1cE5OBQ1ecJKEG5+ViFTDApkXx39DGk9ha98VApch6\nNjyy6+f/xzPf3oTiIOmn6mCWSQVXZ5iTonznWfcam++DGgDuGKTYhP5IdRH1FkXzcuW\nnnJo66PorkNWKln9QGH5p3R28ctGeyOZHPTj1v2A/HKRXDMxn6wUCJ584v5MyAZg4sFT0Zsr\nnscU/fBfid9L5GUwquZtHFZkt5750YJEGyi8LKGZRXaqGdenuRao7Z4AaAL6KWpLkaOB+WJCe\n\nnrZKuBIYEj1XL"
          },
          {
            "algorithm": "ssh-ed25519",
            "key-data": "AAAAAC3NzaC1lZDI1NTE5AAAAICBNTzfDawum7FOMHCFBlEUqDamAuhav3wdAqk\nJbkRCm"
          }
        ]
      },
      "authgroup": "default",
      "device-type": {
        "cli": {
          "ned-id": "cisco-ios-cli-6.85:cisco-ios-cli-6.85"
        }
      },
      ...
      ...
      "service-list": [
        "ncs:services/l3vpn:l3vpn{ACME}"
      ],
      "tailf-ncs:alarms:alarm-summary": {
        "indeterminates": 0,
        "criticals": 0,
        "majors": 0,
        "minors": 0,
        "warnings": 0
      }
    }
  ]
}
```

Step 7

Verify that you have received a **"200 OK"** status, with the response containing the configuration of PE11.

Activity Verification

You have completed this task when you attain this result:

- You have received the PE11 device configuration.

Task 3: Use the GET Method for L3vpn Service Instances with Postman

This task requires you to obtain information about an existing L3vpn service instance created in one of the previous discovery labs. You will obtain the information from the NSO by sending a GET request to a RESTCONF API endpoint.

Activity

Complete these steps:

Step 1

Input the correct URI for all service instances of the L3vpn service.

- Select the running data store by adding **/data** to the base API URL.
- Select all the services by adding **/tailf-ncs:services** to the request URL.
- Select all the L3vpn service instances by adding **/l3vpn:l3vpn** to the request URL.

Make sure that the request URL matches the value **https://10.0.0.102:8443/restconf/data/tailf-ncs:services/l3vpn:l3vpn**.

Step 2

Click **Send**.

The result should be a **"200 OK"** success message similar to the following:

```
{
  "l3vpn:l3vpn": [
    {

```

```

      "vpn-name": "ACME",
      "modified": {
        "devices": [
          "PE11",
          "PE21",
          "PE31"
        ]
      },
      "directly-modified": {
        "devices": [
          "PE11",
          "PE21",
          "PE31"
        ]
      },
      "vpn-id": 10001,
      "link": [
        {
          "link-id": 1,
          "link-name": "Site1",
          "pe-device": "PE11",
          "interface": "0/1",
          "routing-protocol": "bgp"
        },
        {
          "link-id": 2,
          "link-name": "Site2",
          "pe-device": "PE21",
          "interface": "0/0/0/1",
          "routing-protocol": "bgp"
        },
        {
          "link-id": 3,
          "link-name": "Site3",
          "pe-device": "PE31",
          "interface": "0/1",
          "routing-protocol": "static",
          "static-route": [
            {
              "prefix": "192.168.1.0",
              "mask": "255.255.255.0"
            },
            {
              "prefix": "192.168.2.0",
              "mask": "255.255.255.0"
            }
          ]
        }
      ]
    }
  ]
}

```

Step 3

Add the **?depth=2** query parameter to the end of the request URL and click Send.

Make sure that the request URL matches the value **https://10.0.0.102:8443/restconf/data/tailf-ncs:services/l3vpn:l3vpn?depth=2**.

Step 4

Notice the output.

```

{
  "l3vpn:l3vpn": [
    {
      "vpn-name": "ACME",
      "modified": {},
      "directly-modified": {},
      "vpn-id": 10001,
      "link": []
    }
  ]
}

```



All options for formatting, limiting the depth and length of the response, and other details, are specified in the NSO Northbound API documentation.

Activity Verification

You have completed this task when you attain these results:

- You have received the whole configuration for L3vpn service instances.

Task 4: Use the POST Method for L3vpn Service Instances with Postman

This task requires you to create new L3vpn service instances. You will construct a correct body, encode it in JSON, and send it to the RESTCONF API endpoint using a POST method.

Activity

Complete these steps:

Step 1

Gather the required information for creating the payload.

Look at your l3vpn YANG service model definition and review all the required information for this service—the names, valid values, and possible restrictions. Use the following table.

Attribute	Name	Data Type	Restriction	Other
Service name	l3vpn	list	—	—

Attribute	Name	Data Type	Restriction	Other
VPN ID	vpn-id	uint32	Range: 10001–9999	key for list <i>/3vpn</i>
VPN name	vpn-name	string	—	—
Customer	customer	leafref	—	reference to NSO customers
Link	link	list	—	—
Link ID	link-id	uint32	Range: 1–55	key for list <i>/link</i>
Interface	interface	string	—	<i>pe-device</i> and <i>interface</i> combination must be unique
Routing protocol	routing-protocol	enumeration	Options: <i>bgp</i> <i>static</i>	—
PE device	pe-device	leafref	Only PE devices might be selected	<i>pe-device</i> and <i>interface</i> combination must be unique
Static route	static-route	list	Applicable only when static routing is selected	—
Prefix	prefix	inet:ipv4-address	—	key for list <i>static-route</i>
Mask	mask	union of inet:ipv4-address and inet:ipv6address	—	—

To create a new service instance of L3vpn, you need to provide the correct JSON body. You can obtain valid service instance data from an existing service deployment using the NSO CLI or performing a GET request to the service endpoint URI.



You can find this table in the Job Aids.

Step 2

Perform a GET request to obtain information about an existing L3vpn service instance.

- Make sure that you have an **Accept** key in the **Headers** tab, with the value of **application/yang-data+json** to specify output in a JSON format.
- To show only the necessary nodes, you will select them with the **?fields** query in the URI.

Make sure that the request URL matches the value **https://10.0.0.102:8443/restconf/data/tailf-ncs:services/l3vpn:l3vpn?fields=vpn-name;vpn-id;link**.

Step 3

Click **Send**.

The screenshot shows a Postman interface for a GET request to `https://10.0.0.102:8443/restconf/data/tailf-ncs:services/l3vpn:l3vpn?fields=vpn-name,vpn-id,link`. The response status is 200 OK. The response body is displayed in JSON format and is highlighted with a red box. The JSON structure is as follows:

```

1  {
2    "l3vpn:l3vpn": [
3      {
4        "vpn-name": "ACME",
5        "vpn-id": 10001,
6        "link": [
7          {
8            "link-id": 1,
9            "link-name": "Site1",
10           "pe-device": "PE11",
11           "interface": "0/1",
12           "routing-protocol": "bgp"
13         },
14         {
15           "link-id": 2,
16           "link-name": "Site2",
17           "pe-device": "PE21",
18           "interface": "0/0/0/1",
19           "routing-protocol": "bgp"
20         },
21         {
22           "link-id": 3,
23           "link-name": "Site3",
24           "pe-device": "PE31",
25           "interface": "0/0/0/1",
26           "routing-protocol": "bgp"
27         }
28       ]
29     }
30   ]
31 }

```

Step 4

Select the resulting body text and save it in a text editor on your desktop.

Step 5

Modify the output to create a POST request body.

- Leave the top node as **l3vpn:l3vpn**.
- At this time, you will not configure any devices, so keep only the required variables for this service. See the table for the required nodes.
- Change the variable values. Use the table for possible entries.
- Edit the link list to contain only the link-id.

The request body should resemble the following:

```

{
  "l3vpn:l3vpn": [
    {
      "vpn-name": "Test",
      "vpn-id": 10002,
      "link": [
        {
          "link-id": 1
        },
        {
          "link-id": 2
        },
        {
          "link-id": 3
        }
      ]
    }
  ]
}

```

Step 6

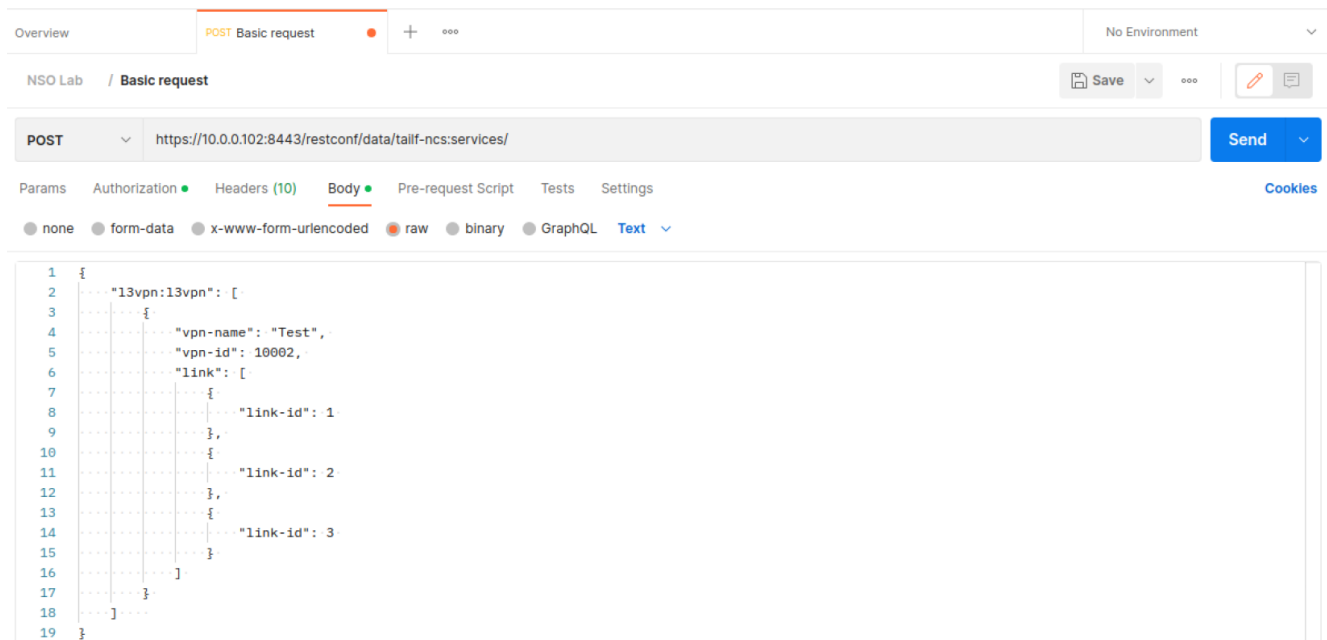
Change the request type in Postman and set it to **POST**.

Step 7

Select the raw format for the request body and paste the new request into the body.

Step 8

Specify the correct path to create a new l3vpn service instance. Select data and services, by adding `/restconf/data/tailf-ncs:services/` to the base API URL. Make sure that the request URL matches the value `https://10.0.0.102:8443/restconf/data/tailf-ncs:services/`.



Overview POST Basic request No Environment

NSO Lab / Basic request Save

POST https://10.0.0.102:8443/restconf/data/tailf-ncs:services/ Send

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

none form-data x-www-form-urlencoded raw binary GraphQL Text

```
1 {
2   "13vpn:13vpn": [
3     {
4       "vpn-name": "Test",
5       "vpn-id": 10002,
6       "link": [
7         {
8           "link-id": 1
9         },
10        {
11          "link-id": 2
12        },
13        {
14          "link-id": 3
15        }
16      ]
17    }
18  ]
19 }
```

Step 9

Click **Send** and view the result.

The result should be a "415 Unsupported Media Type" message with the following error message:

```
"Unsupported media type: text/plain ; Should be one of: application/yang-data+xml, application/yang-data+json."
```

Add a new field in the **Headers** tab. Name it *Content-Type* and set the value to **application/yang-data+json**. This is required because the data is encoded in JSON.

Step 10

Set a new header field called **Content-Type** to **application/yang-data+json**. If a Content-Type field already exists, simply correct its value.

As shown in the error message you received in the previous step, NSO supports data in XML for JSON formats. Further examples of both formats are in the "Northbound APIs" section of the NSO documentation.

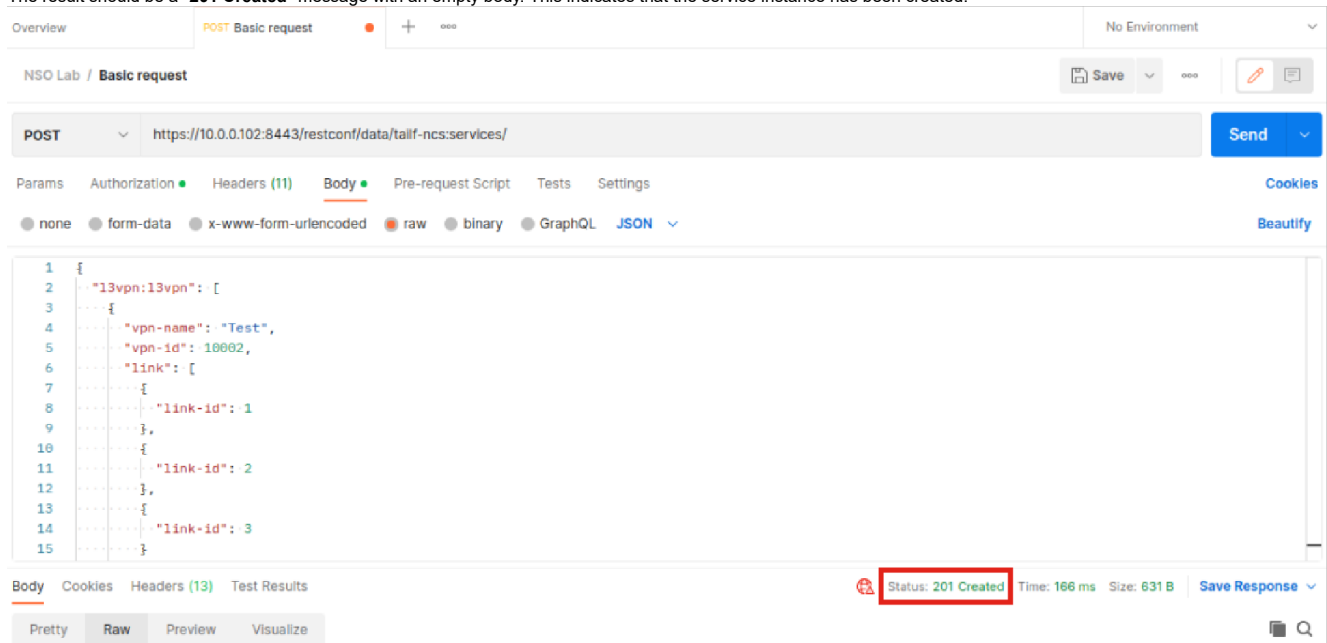
Step 11

Send the request again.

Step 12

View the result.

The result should be a **"201 Created"** message with an empty body. This indicates that the service instance has been created.



Overview POST Basic request No Environment

NSO Lab / Basic request Save

POST https://10.0.0.102:8443/restconf/data/tailf-ncs:services/ Send

Params Authorization Headers (11) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "13vpn:13vpn": [
3     {
4       "vpn-name": "Test",
5       "vpn-id": 10002,
6       "link": [
7         {
8           "link-id": 1
9         },
10        {
11          "link-id": 2
12        },
13        {
14          "link-id": 3
15        }
16      ]
17    }
18  ]
19 }
```

Body Cookies Headers (13) Test Results Status: 201 Created Time: 166 ms Size: 631 B Save Response

Pretty Raw Preview Visualize

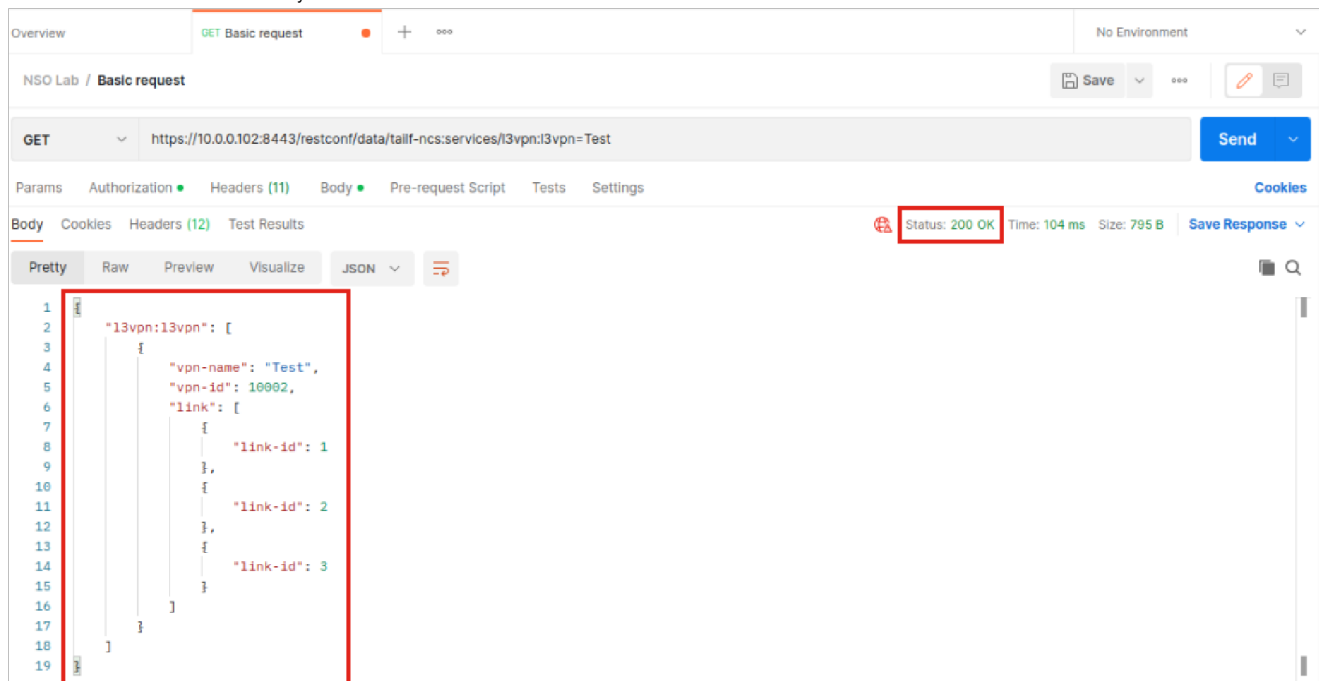
NSO will return a "201 Created" response if it is successful, or a "409 Conflict" status in case an already present element.

Step 13

Verify that the service instance has been created by performing a GET request using what you learned in previous tasks:

- Select only your newly created instance by specifying it after the /l3vpn:l3vpn.
- Make sure that the request URL matches the value **https://10.0.0.102:8443/restconf/data/tailf-ncs:services/l3vpn:l3vpn=Test**.
- Click **Send**.

The result should be visible in the body of Postman.



Step 14

Verify that the service instance has been created using NSO CLI.

- Log in to the Cisco NSO CLI console. (See the previous labs for details.)
- Use a **show** command to display all the l3vpn service instances.
- Verify that your created service instance is present in the output.

```

admin@ncs# show services l3vpn
services l3vpn ACME
modified devices [ PE11 PE21 PE31 ]
directly-modified devices [ PE11 PE21 PE31 ]
services l3vpn Test

```

Activity Verification

You have completed this task when you attain these results:

- You have received a "201 OK" response from NSO when creating a new service instance.
- You can view a new l3vpn service instance using GET RESTCONF API and the Postman app.
- You can view a new l3vpn service instance using the **show services** command in NSO.

Task 5: Use the DELETE method for L3vpn Service Instances with Postman

This task requires you to delete recently created l3vpn service instance by using the RESTCONF DELETE method.

Activity

Complete these steps:

Step 1

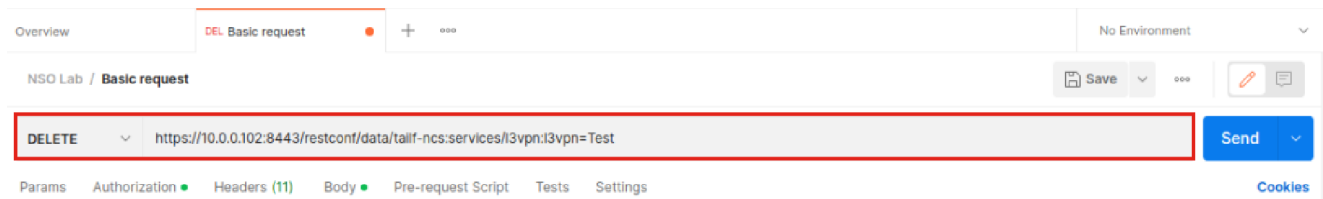
Open a new request tab and set the new request type to **DELETE**.

Step 2

Recreate the URL using the following information:

- Select the data store by adding **/data** to the base RESTCONF API URL.
- Select l3vpn service by adding **/tailf-ncs:services/l3vpn:l3vpn** to the request URL.
- Select the previously created service instance by adding the name of that service instance (or the unique identifier of that service instance as configured in YANG) to the request URL.

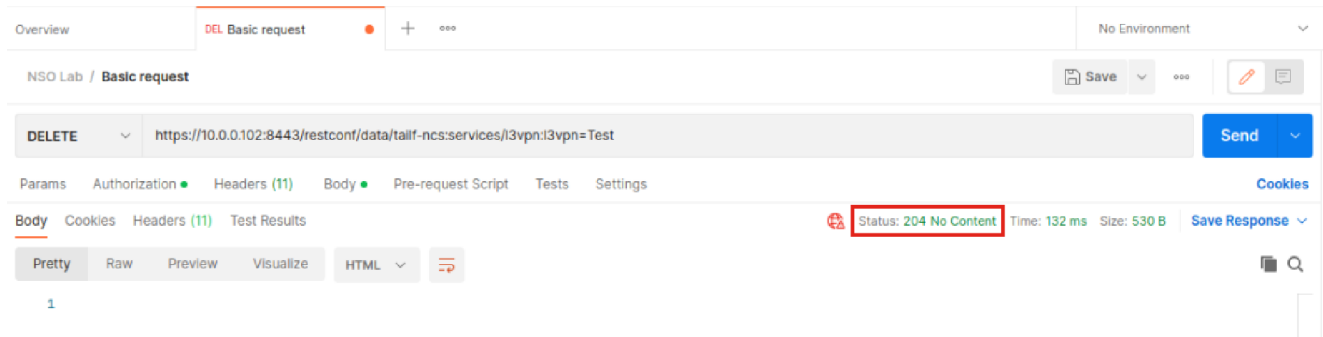
Make sure the request URL matches the value **https://10.0.0.102:8443/restconf/data/tailf-ncs:services/l3vpn:l3vpn=Test**.

**Step 3**

Click **Send**.

Step 4

Verify that the response is a **"204 No Content"** answer with an empty body.

**Step 5**

Verify that the service instance is no longer present with NSO CLI.

```
admin@ncs# show services l3vpn
services l3vpn ACME
modified devices [ PE11 PE21 PE31 ]
directly-modified devices [ PE11 PE21 PE31 ]
```

Activity Verification

You have completed this task when you attain this result:

- Service instance no longer exists in NSO.