Discovery 7: Use Cisco NSO RESTCONF API with Postman

Introduction

In this activity, you will learn how to use the Cisco NSO RESTCONF API. Postman is a tool that you will use to list, view, create, and delete service instances in Cisco NSO by sending appropriate RESTCONF request messages.

Scenario

You have been hired as a junior network engineer at a company that uses Cisco NSO to automate device configuration management and network service activation tasks. Your manager requires that you learn how to use RESTCONF API to help the development team integrate Enterprise IT applications with NSO by providing sample RESTCONF API requests and responses for various services.

Activity Objective

To accomplish the task that your manager assigned to you, you will use a Linux-based virtual machine with the NSO software installed on it in your home directory.

After completing this activity, you will be able to meet the following objectives:

- Use Postman to interact with the Cisco NSO RESTCONF API

- List service instances by using the GET method in Postman

- View the service instance configuration by using the GET request in Postman

- Create a service instance by using the POST method in Postman

- Delete service instance by using the DELETE method in Postman

Job Aid

Device Information

Device	Description	IP Address	Credentials
Student	Linux Ubuntu VM	10.10.20.50	developer, C1sco12345
R1 (dist-rtr01)	Virtual Router, IOS XE - Amsterdam-17.3.4	10.10.20.175	cisco, cisco
R2 (dist-rtr02)	Virtual Router, IOS XE - Amsterdam-17.3.4	10.10.20.176	cisco, cisco
R3 (internet-rtr01)	Virtual Router, IOS XE - Amsterdam-17.3.4	172.21.1.181	cisco, cisco

Task 1: Perform a Basic GET Request in Postman

The first task will guide you in the process to start, test, and understand the basic functionalities of Postman.

Postman Interface

Postman is a set of tools that helps developers and others explore and develop API interfaces. In this lab, you will use Postman to initiate a RESTCONF request and analyze responses. However, there is much more to Postman than simply exploring RESTCONF APIs. This task will familiarize you with the basic layout and functions of Postman.
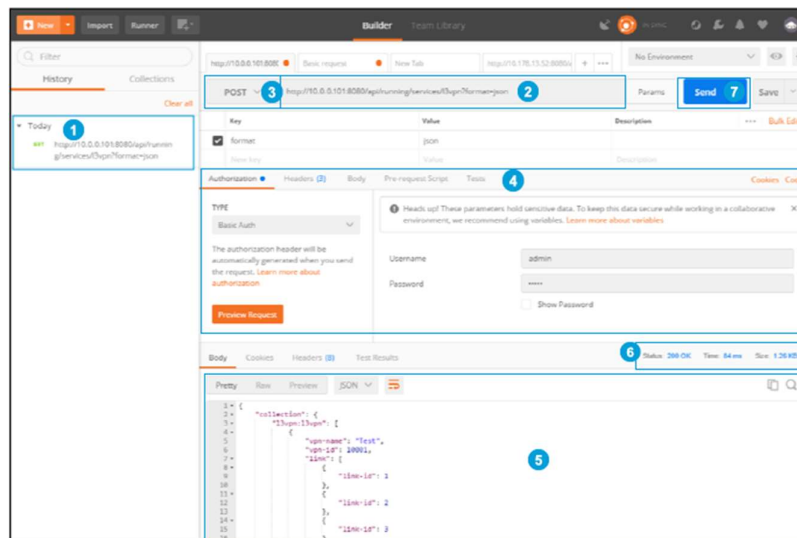
Refer to the figure to familiarize yourself with the Postman GUI and then read the explanation about each numbered element:

1. **History and collections:** 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.

3. **Type of request:** Various types of RESTCONF requests will result in various options in the request parameters. The following is a minimal list of HTTP methods that RESTCONF uses and a description of each operation:

    – **GET:** Reads the specified resource information from a device.

    – **POST:** Creates or performs operations on a device.

    – **DELETE:** Deletes the targeted resource from a device.

4. **Request parameters or headers:** Request parameters transmit with the request and input necessary information into the request. For example, the creation of a service that uses a POST request requires all the necessary parameters for a service instance.

5. **Response body:** Response to the current request.

6. **HTTP Response status and code:** Example: 200 OK.
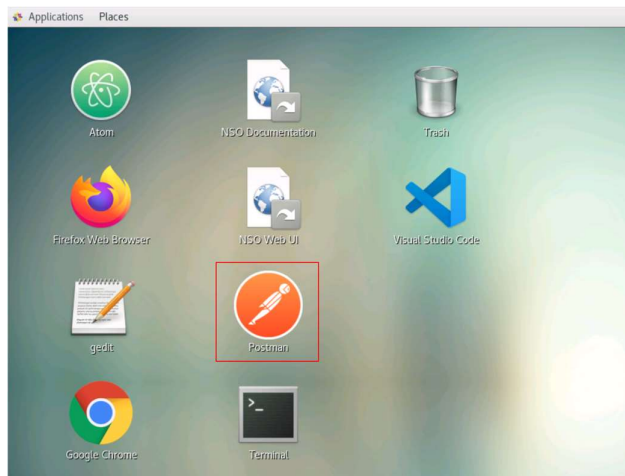
7. **Send button:** Sends the request.

Activity

Step 1   Connect to the Student DevBox.

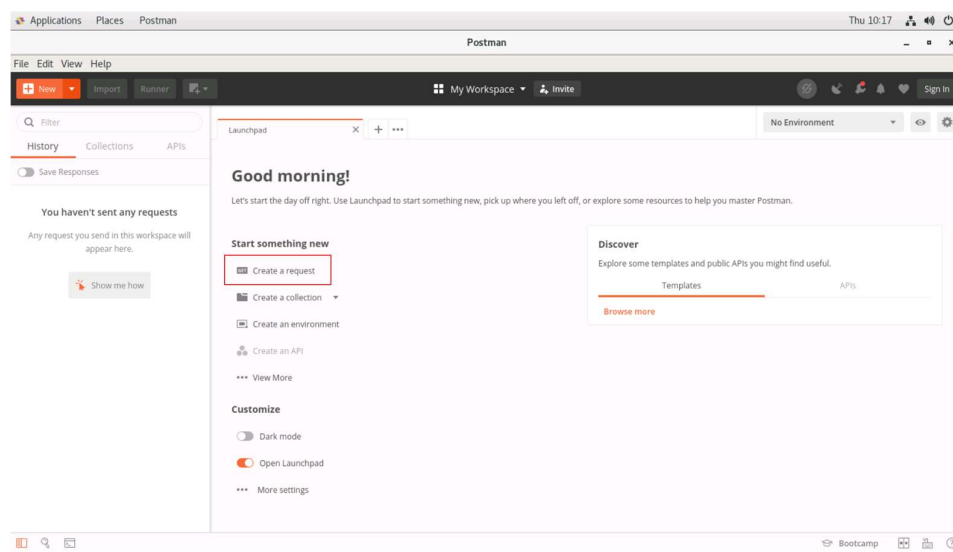On the topology diagram, click the **Student DevBox**.

Step 2   Open the **Postman** application.

Open **Postman** by clicking the icon on the Desktop.
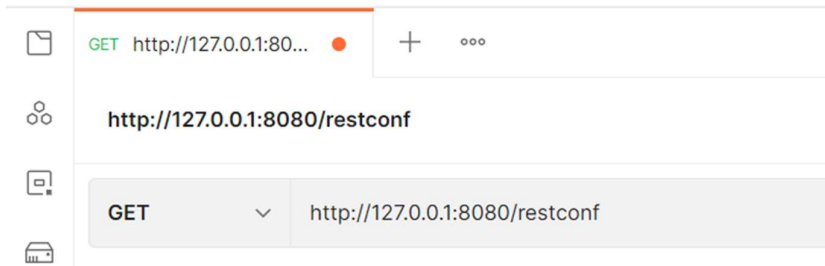

Step 3   In **Postman**, create a new request.

Click **Create a request** at the **Launchpad** section in the center of the screen.

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 **127.0.0.1:8080/restconf/** and click **Send** or press **Enter**.
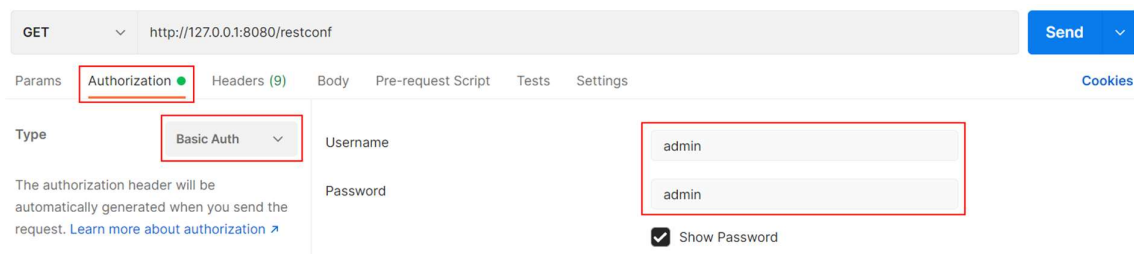

Verify that the RESTCONF request is unsuccessful because no user authorization attaches to the request. In the **HTTP Response status**, observe the **401 Unauthorized** code. In the **Response body** section, observe the **access-denied** message.

Step 5   Complete the request with authorization data by using the username **admin** and the password **admin**. Use basic authorization. Repeat the GET request.

In the **Request parameters** or headers section (below **Type of Request** and **URL**), 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 user name 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.

Activity Verification

You have completed this task when you attain these results:

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

Task 2: Obtain Network Device Information on Cisco NSO

Activity

The goal of this task is that you obtain a list of all devices in the Cisco NSO CDB and then obtain information about specific devices (the R1 router).

Step 1   In Postman, check the list of devices that are currently present in the running CDB datastore of Cisco NSO.

Choose **GET** as the request type. In the request URL, enter the API address **127.0.0.1:8080/restconf/data/tailf-ncs:devices?depth=3** and click **Send** or press **Enter**.
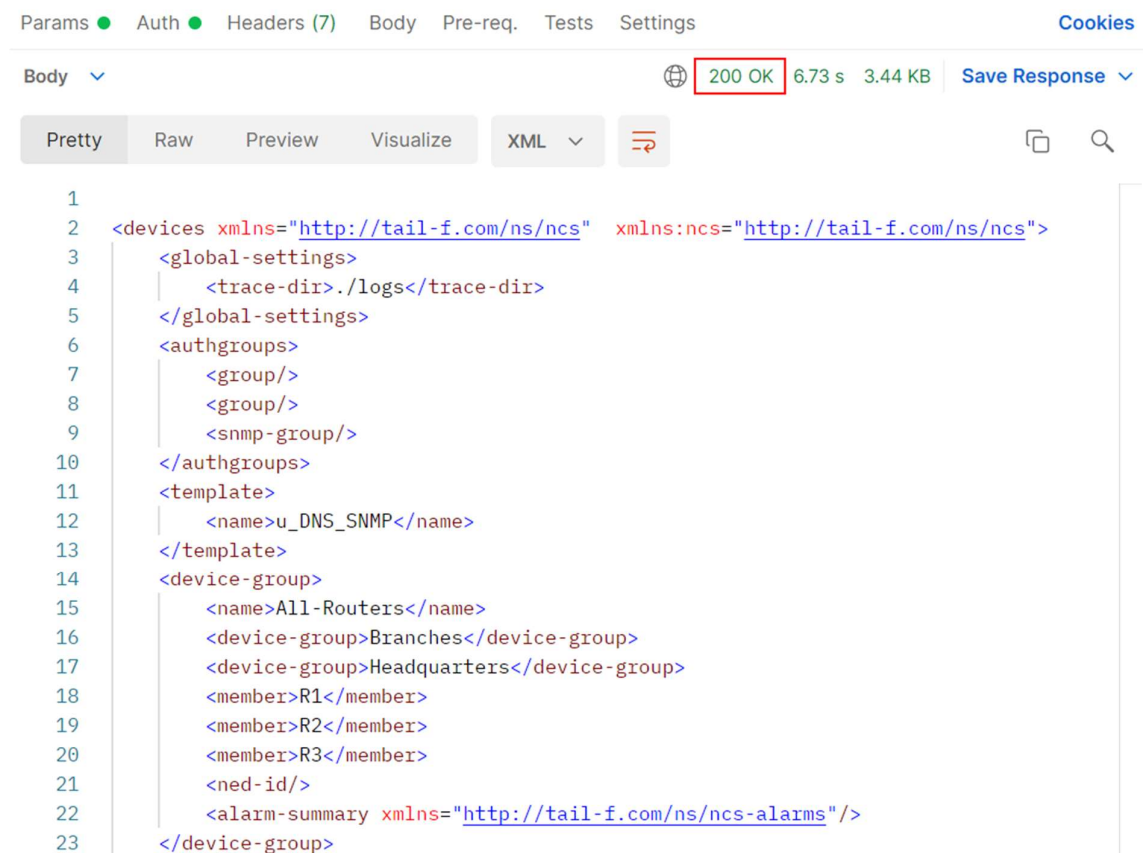
Note	The depth query parameter is used to limit the depth of subtrees that the server returns. Data nodes with a value greater than the depth parameter are not returned in a response to a GET request. The value of the depth parameter is either an integer between 1 and 65535 or the string "unbounded." The default value is "unbounded."

Step 2 Verify that the GET request is successful with a **200 OK** response. The output contains all device names that are currently present in the running datastore of the CDB (**R1, R2**, and **R3**). However, it does not contain detailed information about those devices.

In the **HTTP Response status**, observe the **200 OK** code. In the **Response Body** section, you can see the authgroup, device group, device list, and other information.

```
Params ●  Auth ●  Headers (7)  Body  Pre-req.  Tests  Settings                    Cookies

Body  ⌄                                              ⊕  200 OK  6.73 s  3.44 KB  Save Response ⌄

  Pretty   Raw   Preview   Visualize   XML ⌄   ⇄                         ⌷   Q

  1
  2   <devices xmlns="http://tail-f.com/ns/ncs"  xmlns:ncs="http://tail-f.com/ns/ncs">
  3       <global-settings>
  4           <trace-dir>./logs</trace-dir>
  5       </global-settings>
  6       <authgroups>
  7           <group/>
  8           <group/>
  9           <snmp-group/>
 10       </authgroups>
 11       <template>
 12           <name>u_DNS_SNMP</name>
 13       </template>
 14       <device-group>
 15           <name>All-Routers</name>
 16           <device-group>Branches</device-group>
 17           <device-group>Headquarters</device-group>
 18           <member>R1</member>
 19           <member>R2</member>
 20           <member>R3</member>
 21           <ned-id/>
 22           <alarm-summary xmlns="http://tail-f.com/ns/ncs-alarms"/>
 23       </device-group>
```

Step 3 Use **Postman** to obtain information about the **R1** device.

Choose **GET** as the request type. In the request URL, enter the API address **127.0.0.1:8080/restoncf/data/tailf-ncs:devices/device=R1** and click **Send** or press **Enter**.

```
GET        ⌄    http://127.0.0.1:8080/restconf/data/tailf-ncs:devices/device=R1          Send  ⌄
```

Step 4   Verify that the GET request is successful with a **200 OK** response. The output contains detailed information about the **R1** device. Identify device name, IP address and protocol for connectivity, SSH algorithm, device type, ned-id, and other information. Scroll down to see a complete device configuration.

In the **HTTP Response status**, observe the **200 OK** code. In the **Response Body** section, you can see the device name **R1**, the IP address **172.21.1.21**, the port **22**, **RSA** for the SSH algorithm, **CLI** for the device type, the specific **ned-id**, and other settings.

Activity Verification

You have completed this task when you attain these results:

- You used Postman to make a GET request to retrieve a list of devices from Cisco NSO and received a 200 OK message that contained the list.

- You used Postman to make a GET request to retrieve information for R1 from Cisco NSO and received a 200 OK message that contained detailed information about it.
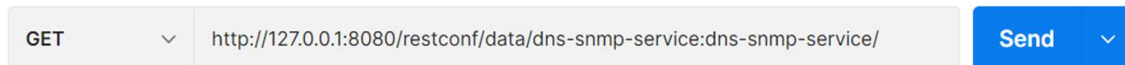
Task 3: Obtain Information for Service Instances

Activity

This task requires you to obtain information about existing DNS and SNMP service instances by using the GET method of the Cisco NSO RESTCONF API. The DNS and SNMP service and one instance of it were created in advance to save lab time.

Step 1   Use **Postman** to list the created instances of the **dns-snmp-service** service.

Choose **GET** as the request type. In the request URL, enter the API address **http://127.0.0.1:8080/restconf/data/dns-snmp-service:dns-snmp-service**/ and click **Send** or press **Enter**.

GET ˅	http://127.0.0.1:8080/restconf/data/dns-snmp-service:dns-snmp-service/	**Send** ˅

Observe the part of the HTTP request, resource **dns-snmp-service:dns-snmp-service.** The first part, **dns-snmp-service**, is DNS and SNMP Service YANG **module name,** and the second part is the **service name**. This can be also seen in the YANG service model that is being used in the DNS and SNMP service. The file contents are displayed in the following figure:

```
1    module dns-snmp-service {
2      namespace "http://cisco.com/examples/dnssnmpservice";
3      prefix dns-snmp-service;
4
5      import ietf-inet-types {
6        prefix inet;
7      }
8      import tailf-ncs {
9        prefix ncs;
10     }
11
12     // add support for NSO syntax help
13     import tailf-common {
14       prefix tailf;
15     }
16
17     organization "CISCO SYSTEMS";
18     contact "CISCO SYSTEMS; support@cisco.com";
19     description "The module to showcase NSO capabilities.";
20     revision 2022-06-22 {
21       description "version 1.0.0.0, see README file.";
22     }
23
24     list dns-snmp-service {
25       tailf:info "DNS and SNMP configuration service";
26       key name;
27
28       uses ncs:service-data;
29       ncs:servicepoint "dns-snmp-service";
30
31       leaf name {
32         tailf:info "Unique service instance name";
33         type string;
34       }
```

Step 2 Verify that the GET request is successful with a **200 OK** response. The output contains all the information about the created instances (**InternalDevices**).

In the **HTTP Response status**, observe the **200 OK** code. In the **Response Body** section, you can see the instance name **InternalDevices** that is applied to devices **R1** and **R2.** The DNS Server IP is **172.21.1.10** and the SNMP Server IP is **172.21.20**.

```
Body ∨                                      ⊕  200 OK  42 ms  1.08 KB    Save Response ∨

  Pretty    Raw    Preview    Visualize    XML ∨   ⇄                          ⎘   🔍

  1
  2    <dns-snmp-service xmlns="http://cisco.com/examples/dnssnmpservice"
         xmlns:dns-snmp-service="http://cisco.com/examples/dnssnmpservice">
  3        <name>InternalDevices</name>
  4        <modified>
  5            <devices>R1</devices>
  6            <devices>R2</devices>
  7        </modified>
  8        <directly-modified>
  9            <devices>R1</devices>
  10           <devices>R2</devices>
  11       </directly-modified>
  12       <devices>R1</devices>
  13       <devices>R2</devices>
  14       <dns-server>172.21.1.10</dns-server>
  15       <snmp-server>172.21.1.20</snmp-server>
  16   </dns-snmp-service>
```

Activity Verification

You have completed this task when you attain these results:

• You used Postman to make a GET request to retrieve details for instances of the DNS and SNMP service from Cisco NSO and received a 200 OK message with the details.

Task 4: Use POST to Create a Service Instance

Activity

This task requires you to create a DNS and SNMP configuration for the device R3 with the POST method in Postman. You must provide the desired service configuration in the body of a POST request. You can speed up the instance creation if similar instances are already created. If so, you can use the GET method to obtain the existing instance configuration and copy and paste it to a text editor. From there, you edit and then copy and paste it into the body of your POST request.

Step 1   Perform a GET request for the **InternalDevices** instance of the **dns-snmp-service** service.

Choose **GET** as the request type. In the request URL, enter the API address **http://127.0.0.1:8080/restconf/data/dns-snmp-service:dns-snmp-service=InternalDevices** and click **Send** or press **Enter**.
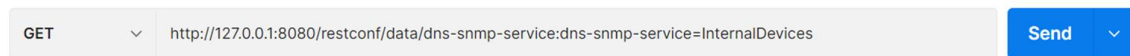
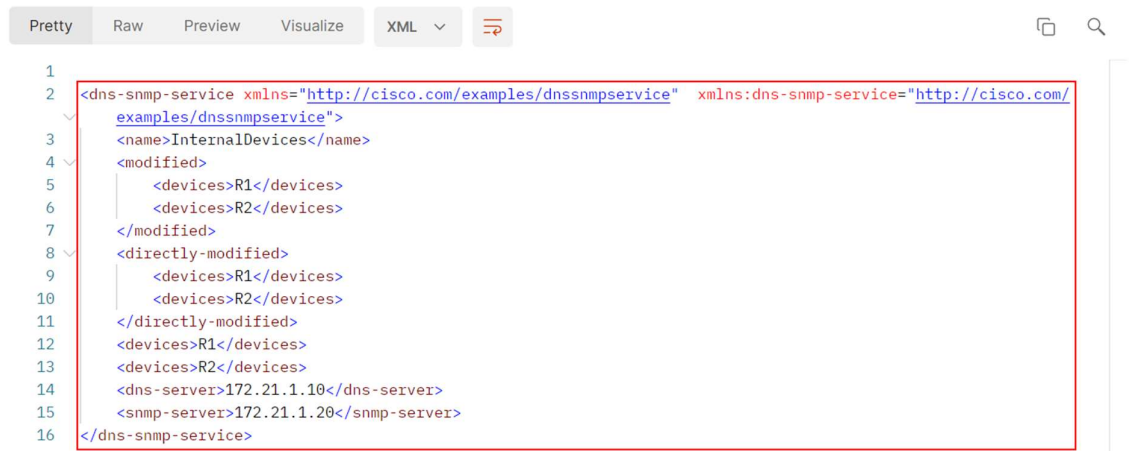| GET | ⌄ | http://127.0.0.1:8080/restconf/data/dns-snmp-service:dns-snmp-service=InternalDevices | **Send** ⌄ |

Step 2   Verify that the GET request is successful with a **200 OK** response. The output contains all the information about the **InternalServices** service instance.

In the **HTTP Response status**, observe the **200 OK** code. In the **Response Body** section, you can see the instance name **InternalDevices** that is applied to devices **R1** and **R2.** The DNS Server IP is **172.21.1.10** and the SNMP Server IP is **172.21.20**.

```
Body   Cookies   Headers (13)   Test Results              🌐 Status: 200 OK  Time: 44 ms  Size: 1.08 KB   Save Response ⌄

  Pretty    Raw    Preview    Visualize       XML ⌄   ⇥                                                        ⧉  🔍

    1
    2   <dns-snmp-service xmlns="http://cisco.com/examples/dnssnmpservice"  xmlns:dns-snmp-service="http://cisco.com/
        examples/dnssnmpservice">
    3       <name>InternalDevices</name>
    4       <modified>
    5           <devices>R1</devices>
    6           <devices>R2</devices>
    7       </modified>
    8       <directly-modified>
    9           <devices>R1</devices>
   10           <devices>R2</devices>
   11       </directly-modified>
   12       <devices>R1</devices>
   13       <devices>R2</devices>
   14       <dns-server>172.21.1.10</dns-server>
   15       <snmp-server>172.21.1.20</snmp-server>
   16   </dns-snmp-service>
```

Step 3   To obtain a structure that allows you to create a new **dns-snmp-service** service instance in **R3**, copy the response body contents from **Postman**.
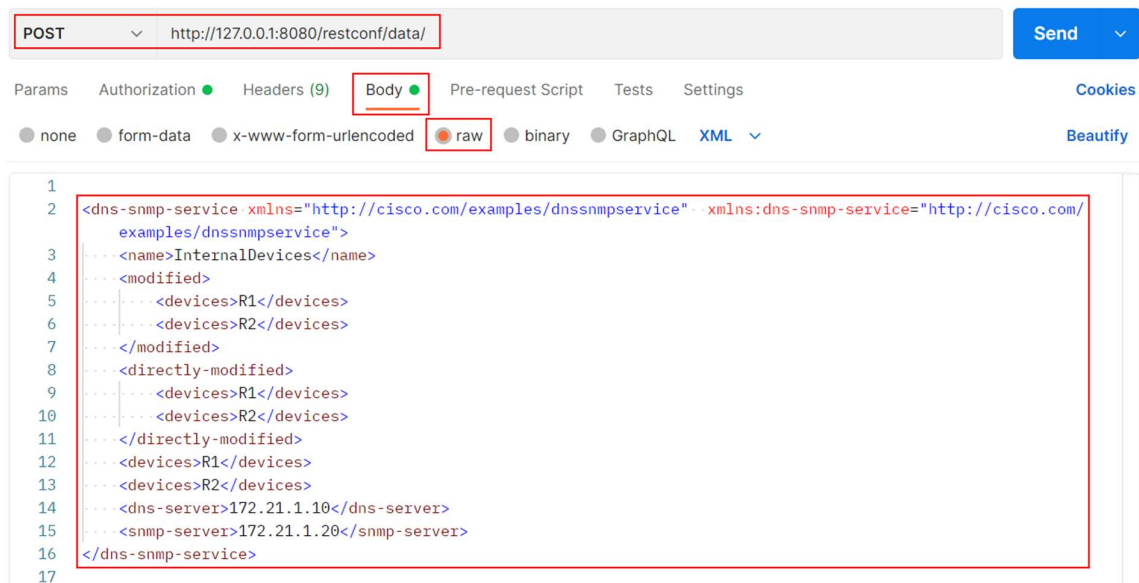
Copy the complete structure.

```
Pretty   Raw   Preview   Visualize   XML  ∨   ⇥

1
2  <dns-snmp-service xmlns="http://cisco.com/examples/dnssnmpservice"  xmlns:dns-snmp-service="http://cisco.com/
       examples/dnssnmpservice">
3      <name>InternalDevices</name>
4      <modified>
5          <devices>R1</devices>
6          <devices>R2</devices>
7      </modified>
8      <directly-modified>
9          <devices>R1</devices>
10         <devices>R2</devices>
11     </directly-modified>
12     <devices>R1</devices>
13     <devices>R2</devices>
14     <dns-server>172.21.1.10</dns-server>
15     <snmp-server>172.21.1.20</snmp-server>
16 </dns-snmp-service>
```

Step 4   Modify your **Postman** request.

Choose the **POST** request type. In the request **URL**, change the API address to **http://127.0.0.1:8080/restconf/data/**, then choose the tab named **Body**, choose the **raw** request body content type, and paste the structure, copied in the previous step, to the request body.

```
POST    ∨    http://127.0.0.1:8080/restconf/data/                          Send  ∨

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

○ none  ○ form-data  ○ x-www-form-urlencoded  ● raw  ○ binary  ○ GraphQL   XML ∨      Beautify

1
2  <dns-snmp-service xmlns="http://cisco.com/examples/dnssnmpservice"  xmlns:dns-snmp-service="http://cisco.com/
       examples/dnssnmpservice">
3      <name>InternalDevices</name>
4      <modified>
5          <devices>R1</devices>
6          <devices>R2</devices>
7      </modified>
8      <directly-modified>
9          <devices>R1</devices>
10         <devices>R2</devices>
11     </directly-modified>
12     <devices>R1</devices>
13     <devices>R2</devices>
14     <dns-server>172.21.1.10</dns-server>
15     <snmp-server>172.21.1.20</snmp-server>
16 </dns-snmp-service>
17
```

Step 5   Modify the body contents by modifying existing values and removing unnecessary XML tags.

Use copied content as a template and modify the service instance parameters. Modify the body contents by modifying existing values and removing unnecessary XML tags. Name your service instance **DMZ**, add only device **R3**, change the IP address of the **DNS Server to 10.1.1.10**, and change the IP address of the **SNMP Server** to **10.1.1.20**. XML tags "modified" and "directly-modified" represent operational data and must be deleted.

After modification, the request content must look exactly as the following:

```
<dns-snmp-service xmlns="http://cisco.com/examples/dnssnmpservice"
xmlns:dns-snmp-service="http://cisco.com/examples/dnssnmpservice">
    <name>DMZ</name>
    <devices>R3</devices>
    <dns-server>10.1.1.10</dns-server>
    <snmp-server>10.1.1.20</snmp-server>
</dns-snmp-service>
```

Verify that the Postman request resembles the following figure:
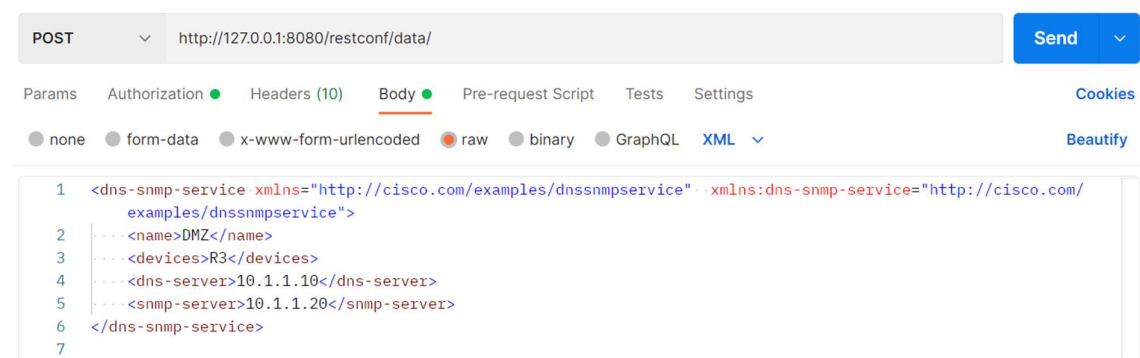
Step 6   Use **Postman** to create a new **dns-server-service** service instance **DMZ**.

Click **Send**.

Step 7   Verify that the result is a **415 Unsupported Media Type** error message.

The result should be a **415 Unsupported Media Type** error message with the following error details: **An Unsupported media type: application/xml ; should be one of: application/yang-data+xml, , application/ yang.data+json.**

Note	Details in the message indicate that data is transmitting to Cisco NSO in plaintext, and Cisco NSO expects it to be in XML or JSON format.

Step 8   Set a new key header field that is called **Content-Type** to the value **application/yang-data+xml** to adapt the data to a format that Cisco NSO understands (XML format in this case). If a Content-Type field exists, simply correct its value and send the POST request again.

Click the **Headers** tab in the **Request parameters or headers**. Create a new key header field that is called **Content-Type**. Use the value **application/yang.data+xml**. Click **Send**.

Step 9   Verify that the result is **201 Created**.

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

Step 10   Open the Terminal application and launch the NSO CLI as admin in the Cisco-style mode.

Open the Terminal application on the Student DevBox and launch the **ncs_cli** as the NSO local user **admin** indicating that you want to use the Cisco style immediately.

```
student@devbox:~$ ncs_cli -Cu admin


User admin last logged in 2022-07-125T15:00:00+00:00, to devbox, from
100.65.0.11 using cli-ssh admin connected from 100.65.0.11 using ssh
on devbox
admin@ncs#
```

Step 11   Confirm that a service instance DMZ for the **dns-snmp-service** service was created.

In NSO CLI, issue the **show running-config dns-snmp-service** command. You should get the following output:

```
admin@ncs# show running-config dns-snmp-service
dns-snmp-service DMZ
 devices     [ R3 ]
 dns-server  10.1.1.10
 snmp-server 10.1.1.20
!
dns-snmp-service InternalDevices
 devices     [ R1 R2 ]
 dns-server  172.21.1.10
 snmp-server 172.21.1.20
!
admin@ncs#
```

Activity Verification

You have completed this task when you attain these results:

- You used Postman with the POST method to create a new DMZ instance for a dns-snmp-service service and received a 201 OK response from Cisco NSO for the created instance.
- You reviewed the new DMZ service instance in NSO CLI.

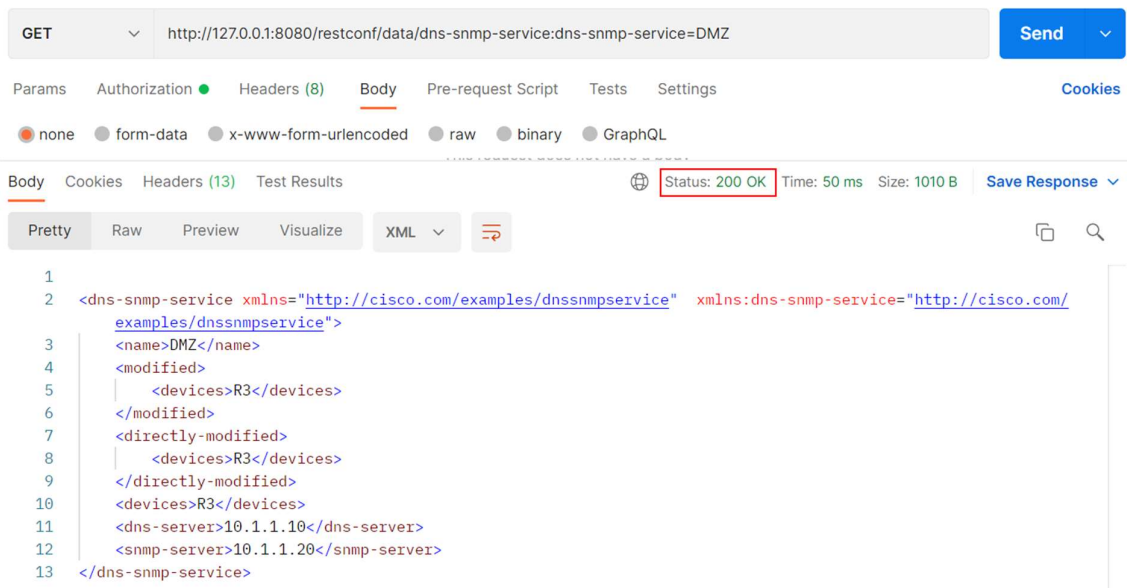# Task 5: Use DELETE to Remove a Service Instance

Activity

This task requires you to delete the recently created DMZ service instance by using the RESTCONF API.

Step 1   Return to the **Postman** application and obtain information for the service instance DMZ.

Modify the existing request. Choose **GET** as the request type. In the request URL, enter the API address **http://127.0.0.1:8080/restconf/data/dns-snmp-service:dns-snmp-service=DMZ** and click **Send** or press **Enter**.

| GET	∨	http://127.0.0.1:8080/restconf/data/dns-snmp-service:dns-snmp-service=DMZ	Send ∨
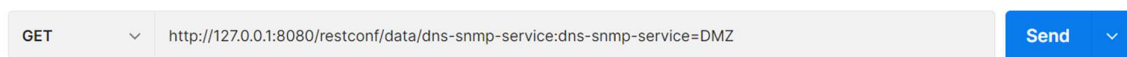
Step 2   Verify that the GET request is successful with a **200 OK** response. The output contains all the information about the **dns-snmp-service** service instance **DMZ**.

GET	∨	http://127.0.0.1:8080/restconf/data/dns-snmp-service:dns-snmp-service=DMZ	Send ∨

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

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

Body   Cookies   Headers (13)   Test Results   🌐 Status: 200 OK Time: 50 ms Size: 1010 B Save Response ∨

Pretty   Raw   Preview   Visualize   XML ∨

```
 1
 2  <dns-snmp-service xmlns="http://cisco.com/examples/dnssnmpservice"  xmlns:dns-snmp-service="http://cisco.com/
       examples/dnssnmpservice">
 3      <name>DMZ</name>
 4      <modified>
 5          <devices>R3</devices>
 6      </modified>
 7      <directly-modified>
 8          <devices>R3</devices>
 9      </directly-modified>
10      <devices>R3</devices>
11      <dns-server>10.1.1.10</dns-server>
12      <snmp-server>10.1.1.20</snmp-server>
13  </dns-snmp-service>
```

Step 3   Change the request type to **DELETE** and delete the service instance.

Choose **DELETE** as the request type. In the request URL, enter the API address **http://127.0.0.1:8080/restconf/data/dns-snmp-service:dns-snmp-service=DMZ** and click **Send** or press **Enter**.

DELETE	∨	http://127.0.0.1:8080/restconf/data/dns-snmp-service:dns-snmp-service=DMZ	Send ∨
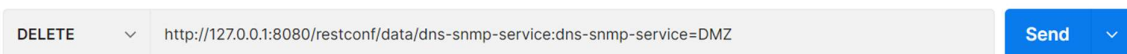
Step 4   Verify that the result is **204 No Content**. Because the deletion request is processed, it returns the message that the related content no longer exists. The body of the message should be empty.

The result should be a **204 No Content** answer with an empty body.
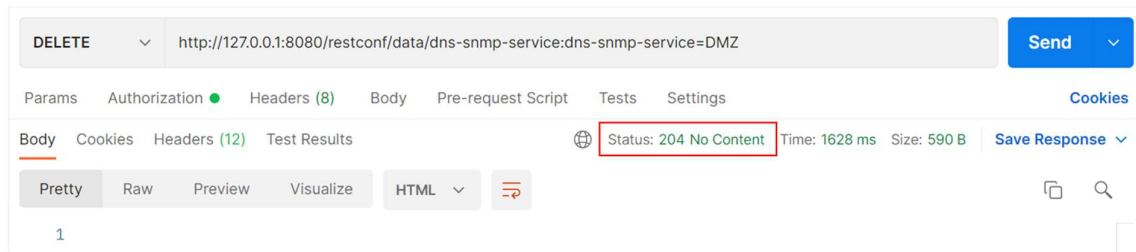
Step 5   On the Student DevBox, launch the NSO CLI as admin in the Cisco-style mode.

Open the Terminal application on the Student DevBox and launch the **ncs_cli** as the NSO local user **admin** indicating that you want to use the Cisco style immediately.

```
student@devbox:~$ ncs_cli -Cu admin


User admin last logged in 2022-07-125T15:00:00+00:00, to devbox, from
100.65.0.11 using cli-ssh admin connected from 100.65.0.11 using ssh
on devbox
admin@ncs#
```

Step 6   Confirm that a DMZ service instance for the **dns-snmp-service** service does not exist anymore.

In NSO CLI, issue the **show running-config dns-snmp-service** command. You should get the following output:

```
admin@ncs# show running-config dns-snmp-service
dns-snmp-service InternalDevices
 devices      [ R1 R2 ]
 dns-server   172.21.1.10
 snmp-server  172.21.1.20
!
admin@ncs#
```

Activity Verification

You have completed this task when you attain these results:

- You used Postman with the DELETE method to delete an existing DMZ instance for the dns-snmp-service service and received a 204 OK response from Cisco NSO for the deleted instance with an empty body.

- You verified that the service instance DMZ no longer exists in NSO CLI.

You have successfully completed all the tasks of this Discovery Lab.