Discovery 3: Use NSO APIs

Introduction

In this lab, you will enable the NSO RESTCONF API and test it by using the Postman tool and curl. You will also configure SNMP to enable alarm collection from NSO.

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

- Configure the NSO RESTCONF API.
- Test the RESTCONF API with Postman and curl.
- Integrate NSO with the external system over RESTCONF.
- Integrate NSO with a monitoring system over SNMP.

Job Aid

The following job aid is available to help you complete the lab activities:

- Lab Guide

The following table contains passwords that you might need.

Device	Username	Password
student-vm	student	1234QWer
nso-server	student	1234QWer
NSO application	student	1234QWer
NSO application	oper	oper123
NSO application	monitor	monitor123

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 List

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

Linux Shell:

Command	Comment
source /opt/ncs/ncs-6.1/ncsrc	Source NSO environmental variable in Docker container.

Command	Comment
ls|ll	Display contents of the current directory.
cd	Move directly to user home directory.
cd ..	Exit out of current directory.
cd test	Move into the "test" folder, which is a subfolder of the current directory.
cd /home/student	Move into the "student" folder by specifying the direct path to it starting from the root of the directory system.
ncs_cli -C	Log in to NSO CLI directly from 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).

Makefile commands for Docker environment:

Command	Comment
make build	Builds the main NSO Docker image.
make testenv-start	Starts the NSO Docker environment.
make testenv-stop	Stops the NSO Docker environment.
make testenv-build	Recompiles and reloads the NSO packages.
make testenv-cli	Enters the NSO CLI of the NSO Docker container.
make testenv-shell	Enters the Linux shell of the NSO Docker container.
make dev-shell	Enters the Linux shell of the NSO Docker development container.

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.
Example	Type **show running config**
Example	Use the **name** command.

Formatting	Description and Examples
`show running config`	Commands in CLI outputs and configurations use this formatting.
highlight	CLI output that is important is highlighted.

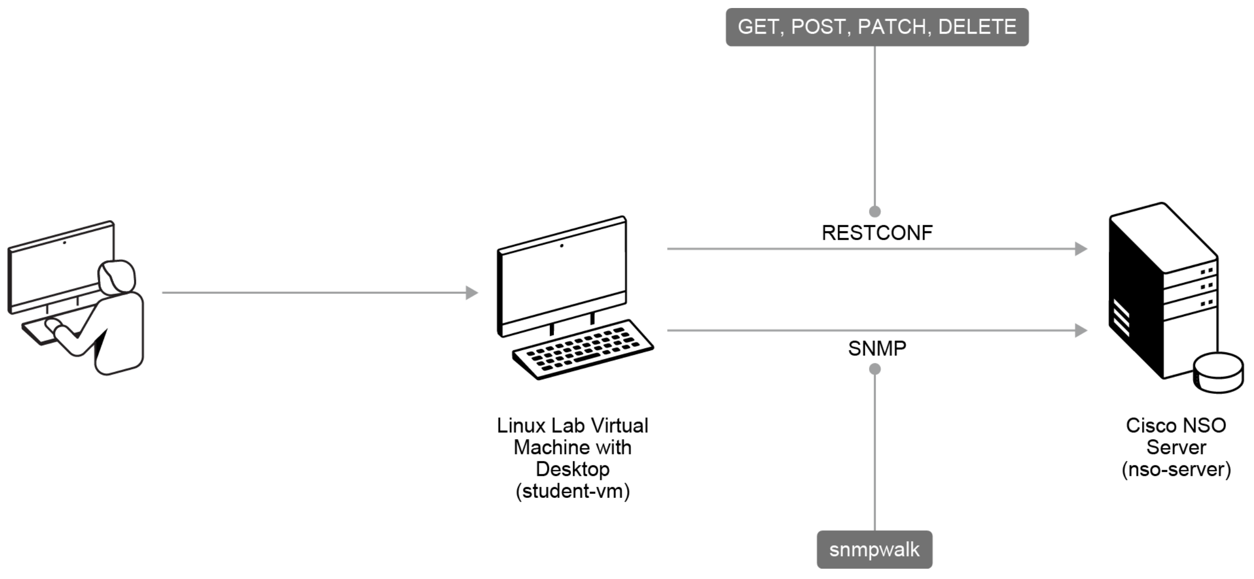
Example	`student@student-vm:~$ ncs --version`
	`6.1`

Example	Save your current configuration as the default **startup config**. `Router Name# copy running startup`
brackets ([ ])	Indicates optional element. You can choose one of the options.
Example:	`(config-if)# frame-relay lmi-type {ansi|cisco|q933a}`
italics font	Arguments for which you supply values.
Example	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.
Example	If the command syntax is **ping** <*ip_address*>, you enter ping *10.0.0.102*.
string	A non-quoted set of characters. Type the characters as-is.
Example	(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.
Example	If the command syntax is **show ip route|arp**, you enter either **show ip route** or **show ip arp**, but not both.

Lab Topology Information

Your lab session is your own personal sandbox. Whatever you do in your session will not be reflected in anyone else's session. Your lab environment is a Linux server (Student-VM) acting as a jumphost and a Linux server (NSO-server) acting as a NSO server. This is where you will be installing NSO.

Topology


GET, POST, PATCH, DELETE
RESTCONF
SNMP
snmpwalk
Linux Lab Virtual Machine with Desktop (student-vm)
Cisco NSO Server (nso-server)

Task 1: Configure RESTCONF API

In this task, you will edit **ncs.conf** to enable RESTCONF API and test functionality by using **curl**.

Activity

Complete the following steps:

Step 1

Connect to the Student-VM.

You can connect to the server either by choosing the **Student-VM** from the device list or by clicking on the **Student-VM** icon in the topology map.

Step 2

Open the terminal window.

Open the terminal window by clicking the **Terminal** icon in the bottom bar.

```
student@student-vm:~$
```

Step 3

Connect to the NSO server **nso-server.**

Connect to NSO server **nso-server** with the **student** user using the SSH client. The authentication is already pre-configured with public key authentication, therefore the password is not needed. The prompt will change, stating you are now connected to the nso-server.

```
student@student-vm:~$ ssh student@nso-server
Last login: Tue Oct  3 09:14:42 2023 from 10.0.0.102
student@nso-server:~$
```

Step 4

Enable RESTCONF API in the **ncs.conf**.

Open **ncs.conf** and find the **webui** section. Enable the **tcp** transport option.

```
student@nso-server:~$ sudo nano /etc/ncs/ncs.conf
  ...

  <webui>
    <enabled>true</enabled>
    <transport>
      <tcp>
        <enabled>true</enabled>
        <ip>0.0.0.0</ip>
        <port>8080</port>
      </tcp>
      <ssl>
        <enabled>false</enabled>
        <ip>0.0.0.0</ip>
        <port>8888</port>
        <key-file>${NCS_CONFIG_DIR}/ssl/cert/host.key</key-file>
        <cert-file>${NCS_CONFIG_DIR}/ssl/cert/host.cert</cert-file>
      </ssl>
    </transport>
    <cgi>
      <enabled>true</enabled>
      <php>
        <enabled>false</enabled>
      </php>
    </cgi>
  </webui>

  <restconf>
    <enabled>true</enabled>
  </restconf>

  ...
```

In production, TCP transport should be disabled, and only SSL (HTTPS) with valid certificates enabled. All WebUI settings are also applied to the RESTCONF interface.

Step 5

Reload Cisco NSO to apply configuration changes.

In the terminal, issue reload of the **ncs** process to apply configuration changes.

```
student@nso-server:~$ sudo /etc/init.d/ncs reload
Reloading ncs: .
student@nso-server:~$
```

Step 6

Test the availability of the RESTCONF API.

Test the RESTCONF API by using the **curl** tool. Basic HTTP authentication is used by default, so you need to specify the username and password. Use the **student** user.

```
student@nso-server:~$ curl -u student:1234QWer nso-server:8080/restconf
<restconf xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
  <data/>
  <operations/>
  <yang-library-version>2019-01-04</yang-library-version>
</restconf>
student@nso-server:~$
```

Step 7

By using the RESTCONF API, try to access the PE11 configuration using the monitor user.

Access the RESTCONF API by using the **curl** tool. Since the **monitor** user has read-only access to the **l3vpn** service instances configuration only, you should receive the access-denied message when requesting other parts of the configuration.

```
student@nso-server:~$ curl -u monitor:monitor123 nso-server:8080/
restconf/data/tailf-ncs:devices/device=PE11/config
<errors xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
  <error>
    <error-message>access denied</error-message>
    <error-tag>access-denied</error-tag>
    <error-type>application</error-type>
  </error>
</errors>
student@nso-server:~$
```

Step 8

Open the Postman tool by using the shortcut on your desktop.

After opening the Postman tool by using the shortcut, ignore and close the login prompt. Open a new tab by clicking the plus (+) sign next to the **Launchpad** tab.

Step 9

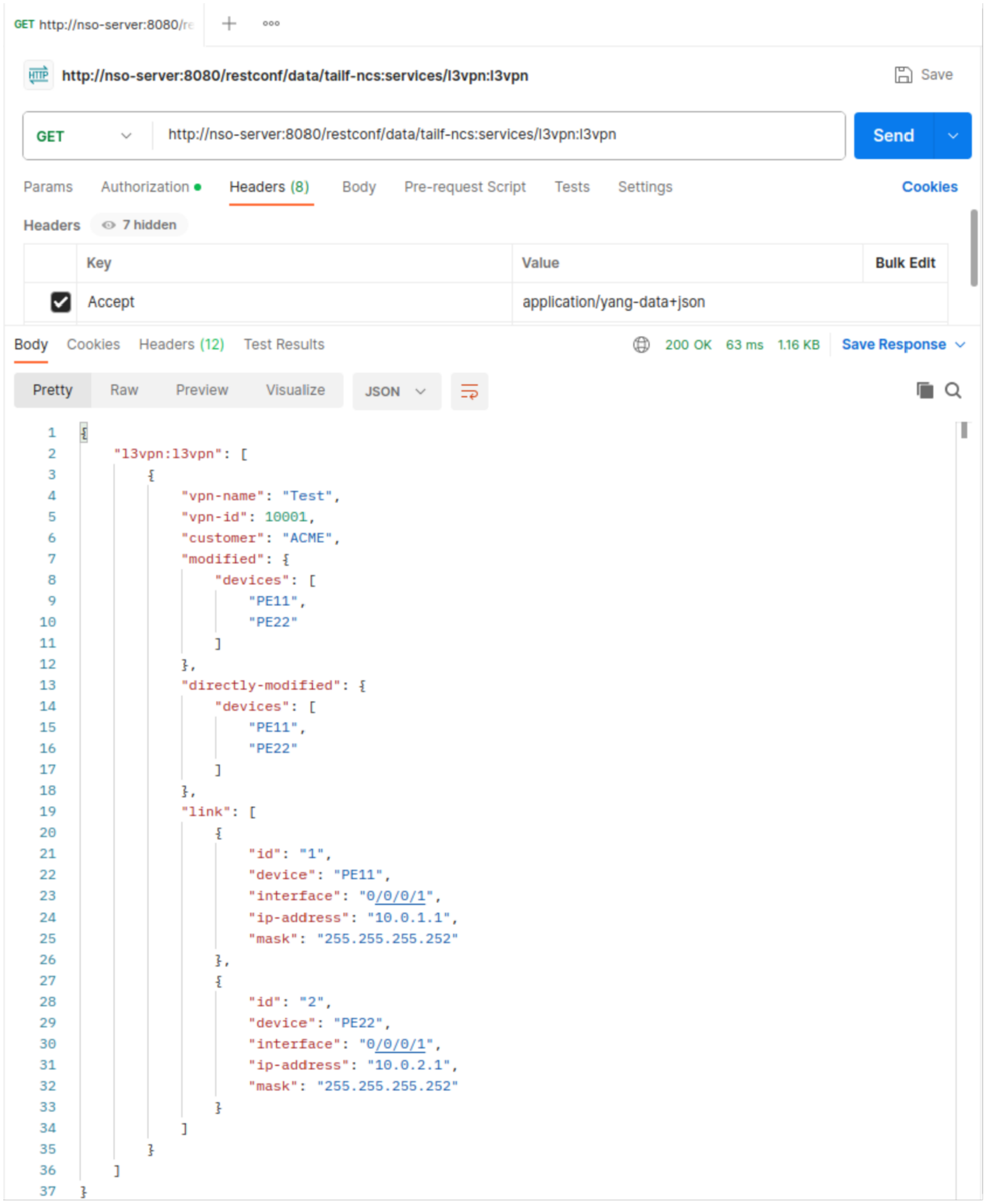
Prepare a HTTP request that will retrieve **l3vpn** service instances.

Input the base URL of the NSO RESTCONF API (**http:<NSO IP>:<port>/restconf/**) and set the request type to **GET**.

Click the **Authorization** tab and choose **Basic Auth**. Use the **oper** user with **oper123** password that has full access to the l3vpn service that you will provision with the RESTCONF API.

Set **Accept** with a value *application/yang-data+json* under the **Headers** tab. The request URL should look like this:

```
http://nso-server:8080/restconf/data/tailf-ncs:services/l3vpn:l3vpn
```



```
http://nso-server:8080/restconf/data/tailf-ncs:services/l3vpn:l3vpn
```

	Key	Value
✓	Accept	application/yang-data+json

```
{
    "l3vpn:l3vpn": [
        {
            "vpn-name": "Test",
            "vpn-id": 10001,
            "customer": "ACME",
            "modified": {
                "devices": [
                    "PE11",
                    "PE22"
                ]
            },
            "directly-modified": {
                "devices": [
                    "PE11",
                    "PE22"
                ]
            },
            "link": [
                {
                    "id": "1",
                    "device": "PE11",
                    "interface": "0/0/0/1",
                    "ip-address": "10.0.1.1",
                    "mask": "255.255.255.252"
                },
                {
                    "id": "2",
                    "device": "PE22",
                    "interface": "0/0/0/1",
                    "ip-address": "10.0.2.1",
                    "mask": "255.255.255.252"
                }
            ]
        }
    ]
}
```

Step 10

Send a HTTP request that will retrieve **l3vpn** service instances.

Send a GET request to the **l3vpn** services configuration by clicking the **Send** button to obtain the structure needed for the body of a POST request. The following is the response that you should receive.

```
{
  "l3vpn:l3vpn": [
    {
      "vpn-name": "Test",
      "vpn-id": 10001,
      "customer": "ACME",
      "modified": {
        "devices": ["PE11", "PE22"]
      },
      "directly-modified": {
        "devices": ["PE11", "PE22"]
      },
      "link": [
        {
          "id": "1",
          "device": "PE11",
          "interface": "0/0/0/1",
          "ip-address": "10.0.1.1",
          "mask": "255.255.255.252"
        },
        {
          "id": "2",
          "device": "PE22",
          "interface": "0/0/0/1",
          "ip-address": "10.0.2.1",
          "mask": "255.255.255.252"
        }
      ]
    }
  ]
}
```

Step 11

Prepare a request that will create a new **l3vpn** service instance.

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

Click the **Body** tab and choose **raw** as the input type.

Copy the output you got with the GET request to create a POST request body.

The top node should be l3vpn:l3vpn list.

Remove the **modified**, **directly-modified**, and **device-list** objects because they are not writable.

Modify the name to **ACME**.

Modify the vpn-id to **10002**.

Modify the link with id 1: change the interface on the PE11 router to **0/0/0/2** and set the ip-address to **10.100.100.1**.

Modify the link with id 2: Change the interface on the PE22 router to **0/0/0/2** and set the ip-address to **10.100.200.1**.

```
{
  "l3vpn:l3vpn": [
    {
      "vpn-name": "ACME",
      "vpn-id": 10002,
      "customer": "ACME",
      "link": [
        {
          "id": "1",
          "device": "PE11",
          "interface": "0/0/0/2",
          "ip-address": "10.100.100.1",
          "mask": "255.255.255.252"
        },
        {
          "id": "2",
          "device": "PE22",
          "interface": "0/0/0/2",
          "ip-address": "10.100.200.1",
          "mask": "255.255.255.252"
        }
      ]
    }
  ]
}
```

Step 12

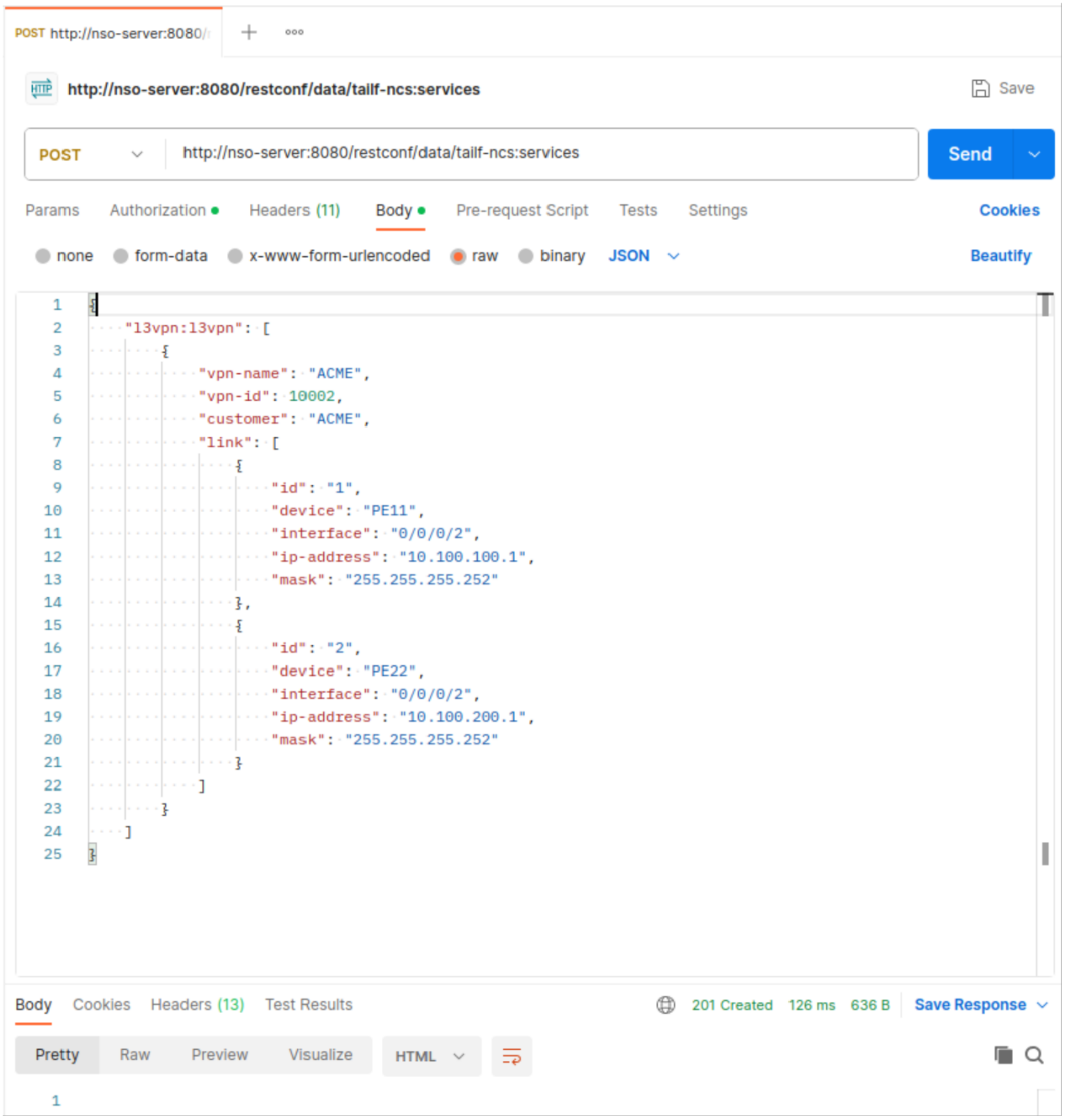
Change the parameters and request a new service instance creation with parameters in the body as shown in the following step.

The POST request needs to be done directly on the parent, which is why the **l3vpn:l3vpn** needs to be removed from the URI. URI should be set to **http://nso-server:8080/restconf/data/tailf-ncs:services/**. Also, set the Content-Type under the Headers tab to **application/yang-data+json**.

Step 13

Send a request and verify its success.

You should receive the status 201 Created as a response.


POST http://nso-server:8080/
http://nso-server:8080/restconf/data/tailf-ncs:services
Save
POST
Send
Params Authorization Headers (11) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary JSON
Beautify
Body Cookies Headers (13) Test Results
Save Response
Pretty Raw Preview Visualize HTML

POST http://nso-server:8080/restconf/data/tailf-ncs:services

```
{
    "l3vpn:l3vpn": [
        {
            "vpn-name": "ACME",
            "vpn-id": 10002,
            "customer": "ACME",
            "link": [
                {
                    "id": "1",
                    "device": "PE11",
                    "interface": "0/0/0/2",
                    "ip-address": "10.100.100.1",
                    "mask": "255.255.255.252"
                },
                {
                    "id": "2",
                    "device": "PE22",
                    "interface": "0/0/0/2",
                    "ip-address": "10.100.200.1",
                    "mask": "255.255.255.252"
                }
            ]
        }
    ]
}
```

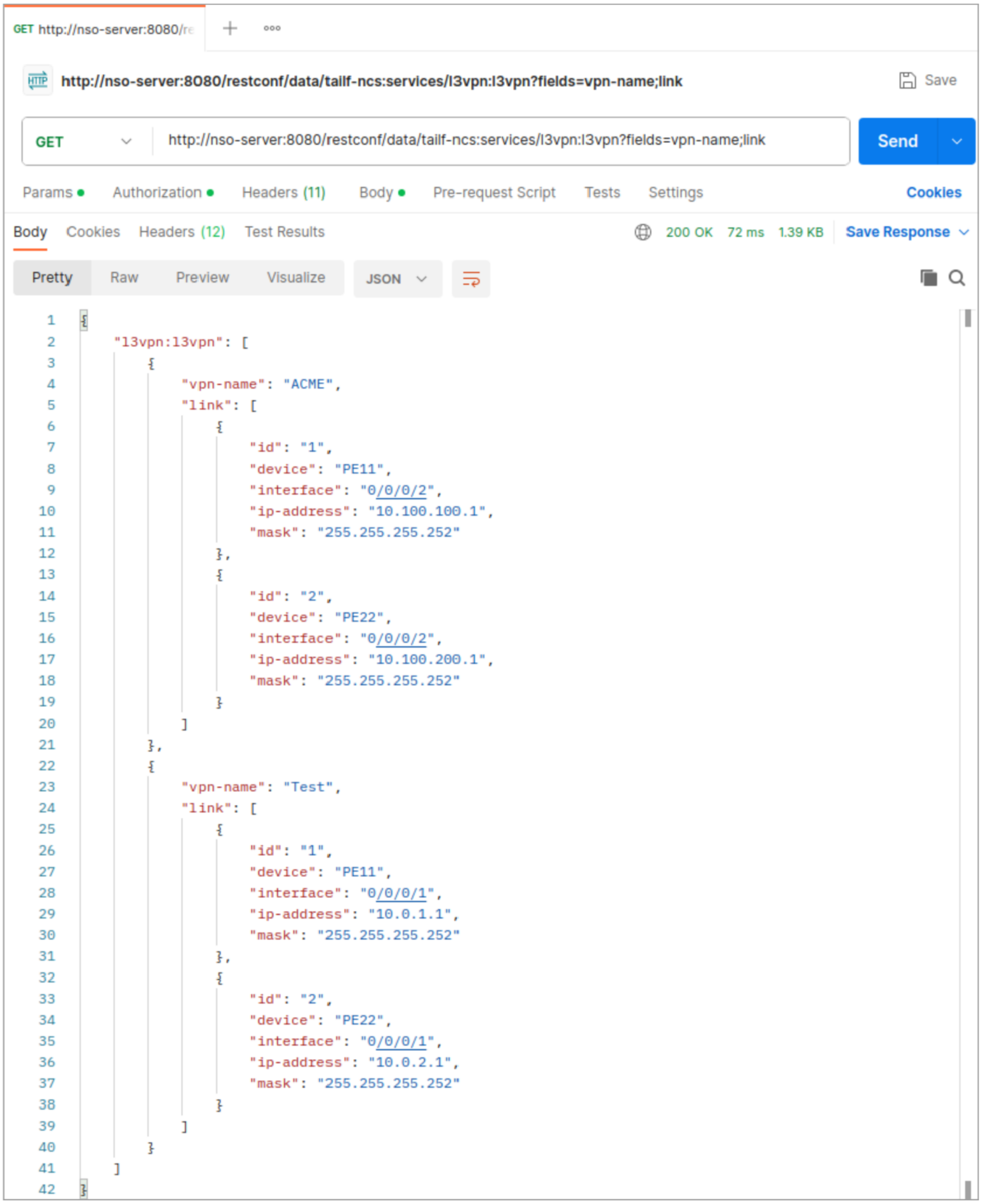
201 Created 126 ms 636 B

Step 14

With the **monitor** user, review if the new service instance was provisioned.

Click the **Authorization** tab and choose **Basic Auth**. Use the **monitor** user with the **monitor123** password. Review through by going to the following link:

```
http://nso-server:8080/restconf/data/tailf-ncs:services/l3vpn:l3vpn?
fields=vpn-name;link
```



http://nso-server:8080/restconf/data/tailf-ncs:services/l3vpn:l3vpn?fields=vpn-name;link

GET http://nso-server:8080/restconf/data/tailf-ncs:services/l3vpn:l3vpn?fields=vpn-name;link

```
{
    "l3vpn:l3vpn": [
        {
            "vpn-name": "ACME",
            "link": [
                {
                    "id": "1",
                    "device": "PE11",
                    "interface": "0/0/0/2",
                    "ip-address": "10.100.100.1",
                    "mask": "255.255.255.252"
                },
                {
                    "id": "2",
                    "device": "PE22",
                    "interface": "0/0/0/2",
                    "ip-address": "10.100.200.1",
                    "mask": "255.255.255.252"
                }
            ]
        },
        {
            "vpn-name": "Test",
            "link": [
                {
                    "id": "1",
                    "device": "PE11",
                    "interface": "0/0/0/1",
                    "ip-address": "10.0.1.1",
                    "mask": "255.255.255.252"
                },
                {
                    "id": "2",
                    "device": "PE22",
                    "interface": "0/0/0/1",
                    "ip-address": "10.0.2.1",
                    "mask": "255.255.255.252"
                }
            ]
        }
    ]
}
```

Activity Verification

You have completed this task when you attain these results:

- You added a new l3vpn service instance through the RESTCONF POST method.
- You checked the instance configuration through the RESTCONF GET method.

Task 2: Configure and Test SNMP

In this task, you will configure SNMP to enable integration with monitoring systems.

Activity

Complete these steps:

Step 1

Open the terminal window.

Open the terminal window by clicking the **Terminal** icon in the bottom bar.

```
student@student-vm:~$
```

Step 2

Connect to the NSO server **nso-server**.

Connect to NSO server **nso-server** with the **student** user using the SSH client. The authentication is already pre-configured with public key authentication, therefore the password is not needed. The prompt will change, stating you are now connected to the nso-server.

```
student@student-vm:~$ ssh student@nso-server
Last login: Fri Oct  6 10:27:21 2023 from 10.0.0.102
student@nso-server:~$
```

Step 3

Configure SNMP agent through the NSO CLI.

Use the following commands:

```
student@nso-server:~$ ncs_cli -C

User student last logged in 2023-10-06T11:18:54.741388+00:00, to nso-
server, from 10.0.0.102 using cli-ssh
student connected from 10.0.0.102 using ssh on nso-server
student@ncs# config
Entering configuration mode terminal
student@ncs(config)# snmp agent enabled
student@ncs(config)# snmp agent udp-port 4000
student@ncs(config)# snmp community public sec-name public
student@ncs(config-community-public)# top
student@ncs(config)# commit
Commit complete.
student@ncs(config)# exit
student@ncs# exit
student@nso-server:~$
```

Step 4

Verify the current state of alarm table in NSO by using the **snmpwalk** tool.

Use the **snmpwalk** tool with the NSO Alarm Management Information Base (MIB) and the Alarms object identifier (OID) to access the current state of the alarm table.

```
student@nso-server:~$ snmpwalk -v 2c -c public -M /opt/ncs/current/src/
ncs/snmp/mibs/ -m TAILF-ALARM-MIB localhost:4000 TAILF-TOP-
MIB::tfModules
TAILF-ALARM-MIB::tfAlarmNumber.0 = Gauge32: 0
student@nso-server:~$
```

Unless you have any issues in the lab, you should see that no alarms are active in the alarm summary section of the previous printout. The very first **snmpwalk** run might not show the output and the printout might be different from the result that you get.

Step 5

Do a configuration change on one of the devices and commit the changes with no-networking option, followed by a device check-sync.

Log in to the NSO CLI and do a configuration change on one of the devices but use the **commit** with **no-networking** option. Then execute **check-sync** towards the device. This action will trigger an alarm that the device is out-of-sync.

```
student@nso-server:~$ ncs_cli -C

User student last logged in 2023-10-06T11:19:15.485202+00:00, to nso-
server, from 10.0.0.102 using cli-ssh
student connected from 10.0.0.102 using ssh on nso-server
student@ncs# config
Entering configuration mode terminal
student@ncs(config)# devices device PE11 config interface Loopback 10
ipv4 address 1.3.3.7 /32
student@ncs(config-if)# top
student@ncs(config)# commit no-networking
Commit complete.
student@ncs(config)# devices device PE11 check-sync
result out-of-sync
info Out of sync due to no-networking or failed commit-queue commits
student@ncs(config)# *** ALARM out-of-sync: Out of sync due to no-
networking or failed commit-queue commits
student@ncs(config)# exit
student@ncs# exit
student@nso-server:~$
```

Step 6

Use the **snmpwalk** command again and verify that you see new alarms.

Exit the NSO CLI and use the **snmpwalk** command again. You should see an alarm that has been triggered due to the device being out-of-sync. Pay attention to the AlarmCleared value.

```
student@nso-server:~$ snmpwalk -v 2c -c public -M /opt/ncs/current/src/
ncs/snmp/mibs/ -m TAILF-ALARM-MIB localhost:4000 TAILF-TOP-
MIB::tfModules
TAILF-ALARM-MIB::tfAlarmNumber.0 = Gauge32: 1
TAILF-ALARM-MIB::tfAlarmLastChanged.0 = STRING: 2023-10-6,11:24:34.7,
+0:0
TAILF-ALARM-MIB::tfAlarmType.1 = STRING: out-of-sync
TAILF-ALARM-MIB::tfAlarmDevice.1 = STRING: PE11
TAILF-ALARM-MIB::tfAlarmObject.1 = STRING: /ncs:devices/
ncs:device[ncs:name='PE11']
TAILF-ALARM-MIB::tfAlarmObjectOID.1 = OID: SNMPv2-SMI::zeroDotZero
TAILF-ALARM-MIB::tfAlarmObjectStr.1 = STRING:
TAILF-ALARM-MIB::tfAlarmSpecificProblem.1 = STRING:
TAILF-ALARM-MIB::tfAlarmEventType.1 = INTEGER: other(1)
TAILF-ALARM-MIB::tfAlarmProbableCause.1 = Gauge32: 0
TAILF-ALARM-MIB::tfAlarmOrigTime.1 = STRING: 2023-10-6,11:24:34.7,+0:0
TAILF-ALARM-MIB::tfAlarmTime.1 = STRING: 2023-10-6,11:24:34.7,+0:0
TAILF-ALARM-MIB::tfAlarmSeverity.1 = INTEGER: major(4)
TAILF-ALARM-MIB::tfAlarmCleared.1 = INTEGER: false(2)
TAILF-ALARM-MIB::tfAlarmText.1 = STRING: Out of sync due to no-
networking or failed commit-queue commits
student@nso-server:~$
```

Step 7

Sync the configuration from CDB to the **PE11** device.

Use the NSO CLI again. Sync the configuration from CDB to the **PE11** device.

```
student@nso-server:~$ ncs_cli -C

User student last logged in 2023-10-06T11:24:01.363129+00:00, to nso-
server, from 10.0.0.102 using cli-ssh
student connected from 10.0.0.102 using ssh on nso-server
student@ncs# devices device PE11 sync-to
result true
student@ncs# exit
student@nso-server:~$
```

Step 8

Execute **snmpwalk** again and verify if the alarm was cleared.

Observe the AlarmCleared value. It should be set to true.

```
student@nso-server:~$ snmpwalk -v 2c -c public -M /opt/ncs/current/src/
ncs/snmp/mibs/ -m TAILF-ALARM-MIB localhost:4000 TAILF-TOP-
MIB::tfModules
TAILF-ALARM-MIB::tfAlarmNumber.0 = Gauge32: 1
TAILF-ALARM-MIB::tfAlarmLastChanged.0 = STRING: 2023-10-6,11:26:40.7,
+0:0
TAILF-ALARM-MIB::tfAlarmType.1 = STRING: out-of-sync
TAILF-ALARM-MIB::tfAlarmDevice.1 = STRING: PE11
TAILF-ALARM-MIB::tfAlarmObject.1 = STRING: /ncs:devices/
ncs:device[ncs:name='PE11']
```

```
TAILF-ALARM-MIB::tfAlarmObjectOID.1 = OID: SNMPv2-SMI::zeroDotZero
TAILF-ALARM-MIB::tfAlarmObjectStr.1 = STRING:
TAILF-ALARM-MIB::tfAlarmSpecificProblem.1 = STRING:
TAILF-ALARM-MIB::tfAlarmEventType.1 = INTEGER: other(1)
TAILF-ALARM-MIB::tfAlarmProbableCause.1 = Gauge32: 0
TAILF-ALARM-MIB::tfAlarmOrigTime.1 = STRING: 2023-10-6,11:24:34.7,+0:0
TAILF-ALARM-MIB::tfAlarmTime.1 = STRING: 2023-10-6,11:26:40.7,+0:0
TAILF-ALARM-MIB::tfAlarmSeverity.1 = INTEGER: major(4)
TAILF-ALARM-MIB::tfAlarmCleared.1 = INTEGER: true(1)
TAILF-ALARM-MIB::tfAlarmText.1 = STRING: Out of sync due to no-
networking or failed commit-queue commits
student@nso-server:~$
```

Step 9

The alarm is now cleared, but still visible in the alarm table. Purge the alarm from the alarm table.

To remove the out-of-sync alarm completely, use the **alarm purge** command from the NSO CLI.

```
student@nso-server:~$ ncs_cli -C

User student last logged in 2023-10-06T11:26:33.981292+00:00, to nso-
server, from 10.0.0.102 using cli-ssh
student connected from 10.0.0.102 using ssh on nso-server
student@ncs# alarms alarm-list alarm PE11 out-of-sync /devices/
device[name='PE11'] "" purge
student@ncs#exit
```

Step 10

Execute **snmpwalk** one more time and verify that the. The alarm count is back to zero.

The output should be similar to the following output:

```
student@nso-server:~$ snmpwalk -v 2c -c public -M /opt/ncs/current/src/
ncs/snmp/mibs/ -m TAILF-ALARM-MIB localhost:4000 TAILF-TOP-
MIB::tfModules
TAILF-ALARM-MIB::tfAlarmNumber.0 = Gauge32: 0
TAILF-ALARM-MIB::tfAlarmLastChanged.0 = STRING: 2023-10-6,11:29:9.7,
+0:0
student@nso-server:~$
```

Step 11

Verify your SNMP agent configuration.

Verify by using the following commands:

```
student@nso-server:~$ ncs_cli -C
```

```
User student last logged in 2023-10-06T11:28:19.245322+00:00, to nso-
server, from 10.0.0.102 using cli-ssh
student connected from 10.0.0.102 using ssh on nso-server
student@ncs# show running-config snmp agent
snmp agent enabled
snmp agent ip    0.0.0.0
snmp agent udp-port 4000
snmp agent version v1
snmp agent version v2c
snmp agent version v3
snmp agent engine-id enterprise-number 32473
snmp agent engine-id from-text testing
snmp agent max-message-size 50000
student@ncs#
```

Activity Verification

You have completed this task when you attain these results:

- You configured the SNMP agent on NSO.
- You verified the SNMP configuration on NSO.

What response code is returned when you create a new service instance using the RESTCONF?

- ○ 200 OK
- ○ 201 Created
- ○ 202 Accepted
- ○ 204 No content