Discovery 11: Build a NETCONF NED

Introduction

In this activity, you will learn how to use NSO to create network element drivers for NETCONF devices. You will build a NETCONF NED by first fetching a list of available YANG modules from the device. Based on this list and your selection, you will then build a network element driver and use it to manage a NETCONF device.

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

- Add a NETCONF device to NSO.
- Create a NETCONF NED using the netconf-ned-builder tool.
- Use the NETCONF NED to manage a NETCONF device.

Job Aid

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

- This Lab Guide

The following table contains passwords that you might need.

Device	Username	Password
student-vm	student	1234QWer
nso-server	student	1234QWer
IOS0 device	admin	admin

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.
ls|ll	Display contents of the current directory.
cd	Move directly to user home directory.

Command	Comment
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 "nso300" 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**:

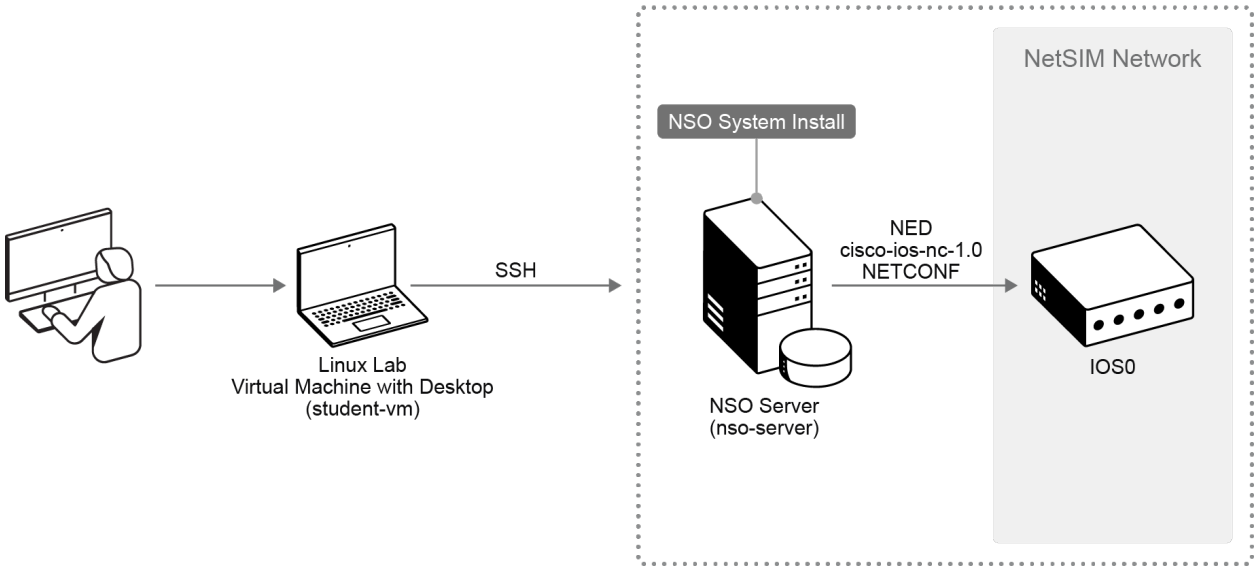
Formatting	Description and Examples
show running config	Commands in steps use this formatting.
Example	Type **show running config**
Example	Use the **name** command.
`show running`	Commands in CLI outputs and configurations use this formatting.

Formatting	Description and Examples
`config`	
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 an NSO server. NSO server includes a NetSIM router. This will be the network that you will orchestrate with your NSO.

Topology



Task 1: Add a NETCONF Device to NSO

In this task, you will add a NETCONF-supported device to NSO. For this purpose, a NetSim-based Cisco IOS device will be used. However, this procedure works for any NETCONF device that implements the YANG models based on the RFC 6022 - YANG Module for NETCONF Monitoring.

Activity

Complete these 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 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 the **nso-server** NSO server with the **student** user using the SSH client. The authentication is already preconfigured with the 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

Navigate to the **~/lab** folder and list the netsim devices with the **ncs-netsim list** command.

Take note of the CLI and NETCONF ports. You will need to use these specific ports later in the lab.

```
student@nso-server:~$ cd lab
student@nso-server:~/lab$ ncs-netsim list
ncs-netsim list for  /home/student/lab/netsim

name=IOS0 netconf=12022 snmp=11022 ipc=5010 cli=10022
dir=/home/student/lab/netsim/IOS/IOS0
student@nso-server:~/lab$
```

Step 5

Connect to the IOS0 device via SSH, using the CLI port you observed in the previous step.

Use the **admin** username and the **admin** password.

```
student@nso-server:~/lab$ ssh admin@127.0.0.1 -p 10022
The authenticity of host '[127.0.0.1]:10022 ([127.0.0.1]:10022)' can't
be established.
ED25519 key fingerprint is
SHA256:tqoTaGI6q730yrWTY28kySq3vmXIjIR1TvY2WC24vbg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
yes
Warning: Permanently added '[127.0.0.1]:10022' (ED25519) to the list of
known hosts.
admin@127.0.0.1's password:

admin connected from 127.0.0.1 using ssh on nso-server
IOS0>
```

Step 6

Enter the privileged mode and display the running configuration of the IOS device, which currently contains nothing. Close the SSH session after that.

Use the commands in the following output.

```
IOS0> enable
IOS0# show running-config
tailfned police cirmode
no service password-encryption
aaa accounting delay-start
```

```
no cable admission-control preempt priority-voice
...
...
interface GigabitEthernet0/1
 no shutdown
 no switchport
 no ip address
exit
IOS0# exit
Connection to 127.0.0.1 closed.
student@nso-server:~/lab$
```

Step 7

Enter the NSO CLI configuration mode.

Use the **ncs_cli -C** command.

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

User student last logged in 2024-02-05T15:17:20.10396+00:00, to nso-
server, from 100.64.0.11 using cli-ssh
student connected from 100.64.0.11 using ssh on nso-server
student@ncs# config
Entering configuration mode terminal
student@ncs(config)#
```

Step 8

Add the IOS device.

Use the following parameters:

- **Authgroup: netconf**
- **Address: 127.0.0.1**
- **Port:** NETCONF port you observed with the **ncs-netsim list** command
- **Device type: netconf**
- **NED ID: netconf**
- **State: unlocked**

The NED-ID: **netconf** parameter makes the device use a generic NETCONF NED, since you do not yet have an IOS NETCONF NED.

```
student@ncs(config)# devices device IOS0 authgroup default address
127.0.0.1 port 12022 device-type netconf ned-id netconf
student@ncs(config-device-IOS0)# state admin-state unlocked
student@ncs(config-device-IOS0)# commit
Commit complete.
student@ncs(config-device-IOS0)# top
student@ncs(config)# exit
```

```
student@ncs#
```

Step 9

Fetch the SSH keys and sync the devices.

Use the **devices fetch-ssh-host-keys** and **devices sync-from** commands.

```
student@ncs# devices fetch-ssh-host-keys
fetch-result {
    device IOS0
    result updated
    fingerprint {
        algorithm ssh-ed25519
        value a2:d4:2a:99:3e:a9:3b:f9:96:76:ea:fa:76:9d:33:a9
    }
}
student@ncs# devices sync-from
sync-result {
    device IOS0
    result true
}
student@ncs#
```

Activity Verification

You have completed this task when you attain this result:

- The device IOS0 has been added as a NETCONF device.

Task 2: Create a NETCONF NED

In this task, you will use the netconf-ned-builder tool to fetch a list of YANG modules, that the NETCONF device supports from the device. Based on your selection of these modules, you will create a NETCONF NED.

Activity

Complete these steps:

Step 1

Enable the developer tools with the **devtools true** command.

Developer tools are needed to use the NETCONF NED building tools.

```
student@ncs# devtools true
student@ncs#
```

Step 2

Enter the configuration mode and add a new NETCONF NED builder project named **cisco-ios**. Set **1.0** as the project version, **Cisco** as the vendor, and **admin** as the

local user. Commit the changes.

The **admin** user must be a local Linux user who uses the same credentials as the NETCONF device.

```
student@ncs# config
student@ncs(config)# netconf-ned-builder project cisco-ios 1.0 device
IOS0 local-user admin vendor Cisco
student@ncs(config-project-cisco-ios/1.0)# commit
Commit complete.
student@ncs(config-project-cisco-ios/1.0)#
```

Step 3

Fetch a list of all the YANG modules by using the **fetch-module-list** command.

This action will retrieve all available modules on the NETCONF device.

```
student@ncs(config-project-cisco-ios/1.0)# fetch-module-list
student@ncs(config-project-cisco-ios/1.0)#
```

Step 4

Display a list of available modules with the **module ?** command.

Any of these modules can be used to create a NETCONF NED.

```
student@ncs(config-project-cisco-ios/1.0)# module ?
Possible completions:
  iana-crypt-hash                          ietf-datastores
  ietf-inet-types                          ietf-interfaces
  ietf-ip                                  ietf-netconf
  ietf-netconf-monitoring                  ietf-netconf-nmda
  ietf-netconf-notifications               ietf-netconf-partial-lock
  ietf-netconf-with-defaults               ietf-network-instance
  ietf-origin                              ietf-restconf
  ietf-restconf-monitoring                 ietf-subscribed-
notifications
  ietf-subscribed-notifications-deviation  ietf-x509-cert-to-name
  ietf-yang-library                        ietf-yang-metadata
  ietf-yang-patch                          ietf-yang-push
  ietf-yang-push-deviation                 ietf-yang-schema-mount
  ietf-yang-types                          tailf-acm
  tailf-common                             tailf-common-monitoring
  tailf-common-monitoring2                 tailf-common-query
  tailf-confd-monitoring                   tailf-confd-monitoring2
  tailf-confd-progress                     tailf-kicker
  tailf-last-login                         tailf-ned-cisco-ios
  tailf-netconf-extensions                 tailf-netconf-forward
  tailf-netconf-inactive                   tailf-netconf-monitoring
  tailf-netconf-query                      tailf-netconf-rollback
  tailf-netconf-transactions               tailf-netconf-with-rollback-
id
  tailf-netconf-with-transaction-id        tailf-progress
```

```
  tailf-rollback                          tailf-tls
  tailf-webui                             tailf-xsd-types
  tailf-yang-patch
student@ncs(config-project-cisco-ios/1.0)#
```

Step 5

Add all the discovered modules to the project selection by using the **module <module> <revision> select** command, and then remove the unnecessary one by using the module <module> <revision> deselect command.

The two parameters support wildcard characters. Add all the modules and all the revisions, and then deselect modules that use **the tailf-acm** and **ietf-netconf-acm** module. This causes the NED compilation to fail.

```
student@ncs(config-project-cisco-ios/1.0)# module * * select
student@ncs(config-project-cisco-ios/1.0)# module tailf-acm * deselect
student@ncs(config-project-cisco-ios/1.0)# module tailf-tls * deselect
student@ncs(config-project-cisco-ios/1.0)# module tailf-webui *
deselect
student@ncs(config-project-cisco-ios/1.0)# module ietf-yang-push *
deselect
student@ncs(config-project-cisco-ios/1.0)# module ietf-yang-push-
deviation * deselect
student@ncs(config-project-cisco-ios/1.0)# module ietf-subscribed-
notifications * deselect
student@ncs(config-project-cisco-ios/1.0)# module ietf-subscribed-
notifications-deviation * deselect
student@ncs(config-project-cisco-ios/1.0)#
```

Step 6

Build the NETCONF NED with the **build-ned** command.

This operation might take a minute or two.

```
student@ncs(config-project-cisco-ios/1.0)# build-ned
student@ncs(config-project-cisco-ios/1.0)#
```


When building NETCONF NEDs, errors can occur due to incompatible YANG modules. Two options are available in such cases. You can either remove the module, if it is not critical for device management, or build a development NED, using the **make-development-ned** command, and fixing the YANG module.

Errors and warnings during the NED build can be inspected by unhiding the debug tools using the **unhide debug** command and then running the **show netconf-ned-builder project compiler-output** command in the operational mode.

Step 7

Export the NED by using the **export-ned to-directory** command.

Make sure that the export directory is writable by the local user used to connect to the NetSim device.

```
student@ncs(config-project-cisco-ios/1.0)# export-ned to-directory /
home/student
tar-file /home/student/ncs-6.1-cisco-ios-nc-1.0.tar.gz
student@ncs(config-project-cisco-ios/1.0)#
```

Step 8

Exit the NSO CLI.

Use the **top** and **exit** commands.

```
student@ncs(config-project-cisco-ios/1.0)# top
student@ncs(config)# exit
student@ncs# exit
student@nso-server:~$
```

Step 9

Copy the exported package to the **packages** folder of the NSO running directory.

The exported package is pre-compiled and does not need to be uncompressed or built.

```
student@nso-server:~/lab$ cd
student@nso-server:~$ cp -r ncs-6.1-cisco-ios-nc-1.0.tar.gz /var/opt/
ncs/packages/
student@nso-server:~$
```

Step 10

Enter the NSO CLI and reload the packages.

To reload the packages, use the **packages reload** command. Make sure that the **cisco-ios-nc-1.0** package is successfully reloaded by inspecting the reload result.

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

student@ncs# packages reload

>>> System upgrade is starting.
>>> Sessions in configure mode must exit to operational mode.
>>> No configuration changes can be performed until upgrade has
completed.
>>> System upgrade has completed successfully.
reload-result {
```

```
    package cisco-ios-nc-1.0
    result true
}
```

Activity Verification

You have completed this task when you attain these results:

- The **cisco-ios-nc-1.0** NETCONF NED has been successfully built and reloaded.

Task 3: Use a NETCONF NED to Manage a Device

In this task, you will use the **cisco-ios-nc-1.0** NETCONF NED to manage an NSO device.

Activity

Complete these steps:

Step 1

Enter the config mode and set the ned-id for the IOS device to the **cisco-ios-nc-1.0** NED.

Commit the transaction. The device can now be managed by NSO.

```
student@ncs# config
Entering configuration mode terminal
student@ncs(config)# devices device IOS0
student@ncs(config-device-IOS0)# device-type netconf ned-id cisco-ios-
nc-1.0
student@ncs(config-device-IOS0)# commit
Commit complete.
student@ncs(config-device-IOS0)# top
student@ncs(config)#
```

Step 2

Check your configuration options with the **devices device IOS0 config ?** command.

The capabilities of the device depend on the loaded modules. In your case, the NETCONF NED is practically identical to the CLI NEDs you have been using in other labs.

```
student@ncs(config)# devices device IOS0 config ?
Possible completions: (first 100)
  aaa                          Authentication, Authorization and
Accounting.
  access-list                  Add an access list entry
  access-session               Access Sesion Global Configuration
Commands
  acr                          Configure ACR type
  alarm-contact                Configure the system alarm contact
settings
```

```
  alarm-profile               Configure Alarm Profile
  alias                       Create command alias
  ap                          Configures Cisco APs
  app-hosting                 Application hosting configuration mode
  aqm-register-fnf            Export audio/voice stats to flow record
  archive                     Archive the configuration
  arp                         Set a static ARP entry
  audit                       Router Audit
  authentication              Auth Manager Global Configuration
Commands
  auto                        Configure Automation
  autonomic                   Autonomic Networking
  avc                         Application visibility and control
  banner                      Define a login banner
  bba-group                   Configure BBA Group
  ...
```

Step 3

Add a static route to the IOS0 device, commit the changes, and exit the NSO CLI.

Use the following commands:

```
student@ncs(config)# devices device IOS0 config ip route 10.100.0.0
255.255.0.0 1.0.0.1
student@ncs(config-config)# commit
Commit complete.
student@ncs(config-config)# top
student@ncs(config)# exit
student@ncs# exit
student@nso-server:~/lab$
```

Step 4

Connect to the IOS0 device via SSH again.

Password for the IOS0 device is **admin**.

```
student@nso-server:~/lab$ ssh admin@127.0.0.1 -p 10022
admin@127.0.0.1's password:

admin connected from 127.0.0.1 using ssh on student-vm
IOS0>
```

Step 5

Verify that the route has been added to the device.

Enter the privileged mode and display the running configuration. By inspecting the running configuration, make sure that the route has been added to the device.

```
IOS0> enable
IOS0# show running-config
```

```
tailfned police cirmode
no service password-encryption
aaa accounting delay-start
no cable admission-control preempt priority-voice
no cable qos permission create
no cable qos permission update
no cable qos permission modems
ip source-route
no ip gratuitous-arps
no ip cef
ip finger
no ip http server
no ip http secure-server
no ip forward-protocol nd
ip route 10.100.0.0 255.255.0.0 1.0.0.1
no ipv6 cef
no dot11 syslog
interface Loopback0
 no shutdown
 ip address 127.0.0.1 255.0.0.0
...
```

Activity Verification

You have completed this task when you attain this result:

- You have added a route to the IOS0 device using the NETCONF NED.

Which command do you need to invoke first before you can build a NED using a NETCONF Builder?

- **devices check-sync**
- **devtools true**
- **fetch-module-list**
- **netconf-ned-builder**