## Lab 20: Access a Cisco IOS XE device using NETCONF

### Case Study

GlobalServe Solutions, a regional cloud service provider operating its own data center, faced increasing operational inefficiencies due to the manual configuration of networking devices. Network engineers were required to perform repetitive CLI-based changes across hundreds of switches, routers, and firewalls. This manual approach often resulted in configuration drift, delays in service provisioning, and an elevated risk of human error. These limitations significantly hindered the company’s ability to scale its services and respond promptly to evolving customer demands.

To overcome these challenges, the company adopted an automation framework based on NETCONF and YANG, enabling centralized, model-driven configuration management across a multi-vendor network environment. Following the implementation, GlobalServe Solutions achieved substantial improvements in operational efficiency, including a 70% reduction in configuration time and a notable decrease in deployment-related errors. The inclusion of automated rollback and validation capabilities further enhanced service reliability.

Moreover, the ability to rapidly deploy and adjust network services empowered the company to deliver more agile, scalable, and secure solutions to its customers. As a result, customer satisfaction increased, driven by faster onboarding, reduced downtime, and consistent service quality.

### Business Challenge

As GlobalServe Solutions networks continue to grow in size and complexity, organizations face increasing challenges in managing configurations across a diverse and dynamic infrastructure. Manual configuration through traditional command-line interfaces is time-consuming, error-prone, and lacks scalability, making it difficult to maintain consistency, enforce compliance, and respond swiftly to operational changes. These limitations hinder network agility, increase the risk of downtime, and result in inefficient resource utilization.

To address these challenges, organizations require a standardized and automated solution that enables secure, programmatic configuration and state management of network devices through structured YANG models. To this end, the company engaged your expertise as a certified DevNet Associate to design and implement an effective, automation-driven solution.

### Solution

To address the operational inefficiencies and risks associated with manual network configuration, you propose the implementation of a NETCONF-based automation framework integrated with YANG data models. This solution enables centralized, secure, and programmable management of network devices, ensuring consistent and error-free configurations across multi-vendor environments.

By leveraging NETCONF, the organization can automate routine tasks, validate configurations before deployment, and implement version-controlled changes with rollback capabilities. This transformation will significantly enhance network agility, reduce downtime, and improve overall performance, aligning the infrastructure with modern business requirements and ensuring faster, more reliable service delivery to end customers.

The Network Configuration Protocol (NETCONF), as defined in RFCs 4741 and 6241, facilitates communication with network devices using YANG data models. A data modeling language called Yet Another Next Generation (YANG) defines the format of data sent by network management protocols like NETCONF. When NETCONF is used to communicate with a Cisco IOS XE device, the data is formatted in XML. In this lab exercise, we will utilize ncclient, a Python library used for creating NETCONF clients. Ncclient will be used to obtain the device configuration, verify the NETCONF connection, and edit it programmatically.

1. Launch the DEVNET IOS XE VM and verify connectivity.
2. Use a NETCONF session to gather information.
3. Use ncclient to connect to NETCONF.
4. Use ncclient to retrieve the configuration.
5. Use ncclient to configure a device.

|  |
| --- |
| **// Launch the DENVET Cisco IOS XE VM and Verify Connectivity**  1. Go to the following website link:  **https://developer.cisco.com/sandbox.html?ReturnUrl=https://devnetsandbox.cisco.com**.Thenloginto DevNet Sandbox.    2. Click on the **Labels** dropdown. Select the **Always-On** check box.    3. In the **Catalyst 8000 Always-On Sandbox**, click on the **Launch** button.    4. Click on the **Review Summary** button. Here you can manage the time duration of the environment. But for this lab, we are using the default 2 days.    5. Click on the **Launch Environment** button.    6. It will take a few seconds to complete the launch of an environment.    7. Click on the **I/O** tab. Here you will get your credentials to access the Cisco IOS XE device via SSH.    8. Turn on the Ubuntu VM and open the **Terminal**. Execute the following command: **ssh username@ip\_address** to SSH the Cisco IOS XE router.  The first time you SSH to the Cisco IOS XE router, the Ubuntu VM warns you about the authenticity of the Cisco IOS XE router. Because you trust the Cisco IOS XE router, answer **yes** to the prompt.  9. Enter your **password** and press **ENTER**. You should now be at the privileged EXEC command prompt for the Cisco IOS XE router. |

|  |  |  |  |
| --- | --- | --- | --- |
| **// Use a NETCONF Session to Gather Information**  1. Execute the following command: **show platform software yang-management process** to verify that the NETCONF SSH daemon (ncsshd) is running.    2. If NETCONF is not running, as shown in the output above, execute the global configuration command **netconf-yang**.  3. Open a new **Terminal** window. In this step, we will re-establish an SSH session with the Cisco IOS XE router. However, this time, provide the NETCONF port 830 and run netconf as a subsystem command. Execute the following command: **ssh username@<IP\_Address> -p 830 -s netconf**. After executing this command, enter your **password**.    4. The Cisco IOS XE router will respond with a **hello** message including almost 400 lines of output describing all of its NETCONF capabilities. The end of NETCONF messages is identified with **]]>]]>**.    5. To initiate a NETCONF session, the client must send its own welcome message. The hello message should indicate the NETCONF basic capabilities version that the client intends to utilize. Copy and paste the below-provided XML code into the SSH session.   |  | | --- | | <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  <capabilities>  <capability>urn:ietf:params:netconf:base:1.0</capability>  </capabilities>  </hello>  ]]>]]> |     6. Navigate to the Cisco IOS XE Router VM SSH terminal window and execute the **show netconf-yang sessions** command to verify that a NETCONF session has been started.    7. Switch to the NETCONF SSH session terminal window of Cisco IOS XE. Copy and paste the below-provided RPC get message XML code into the terminal SSH session to retrieve information about the interfaces on the Cisco IOS XE router.   |  | | --- | | <rpc message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  <get>  <filter>  <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"/>  </filter>  </get>  </rpc>  ]]>]]> |     8. Remember that XML does not require indentation or whitespace. As a result, the Cisco IOS XE router will return a large string of XML data.    9. Copy the returned XML, but exclude the last **]]>]]>** characters. These characters do not appear in the XML given by the router. Search the internet for **prettify XML**. Find an appropriate site and utilize their tool to convert your XML into a more readable format.  10. Copy the below-provided RPC message and paste it into the NETCONF SSH session terminal window of Cisco IOS XE to close the NETCONF session.   |  | | --- | | <rpc message-id="9999999" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  <close-session />  </rpc> |     11. After a few seconds, you will be taken back to the terminal prompt. Switch to the Cisco IOS XE router SSH session and execute the **show open netconf sessions** command. You will see that the session has been closed. |

|  |  |  |
| --- | --- | --- |
| **// Use ncclient to Connect to NETCONF**  1. In the terminal window, execute the following command: **pip3 list --format=columns | grep ncclient** to see ncclient Python module is installed or not. If not, run the **pip3 install ncclient** command to install it.    2. Open **Visual Studio Code**. Click on the **File.** Then click on the **Open Folder…**    3.Click on the **devnet-src** directory. Then click on the **OK** button to open it.    4. Open the **Terminal** in Visual Studio Code by dragging it up and execute the following command: **mkdir netconf** to create a directory.    5. In the **EXPLORER** pane under **DEVNET-SRC**, right-click the **netconf** directory. Then click on the **New File**.    6. Name the file **ncclient-netconf.py**.    7. Copy and paste the below-provided script in **ncclient-netconf.py.** Press **Ctrl+S** to save it. Enter your **IP address**, **username**, and **password**.   |  | | --- | | from ncclient import manager  m = manager.connect(  host="Your\_IP\_Address",  port=830,  username="Your\_Username",  password="Your\_Password",  hostkey\_verify=False  ) |     8. Execute the **cd netconf** command to go inside a directory. Then execute the following command: **python3 ncclient-netconf.py** to run the Python script and also verify that there are no errors. You will not see any output yet.    9. You can verify that the Cisco IOS XE router accepted the request for a NETCONF session. There should be a **%DMI-5-AUTH\_PASSED** syslog message in the Cisco IOS XE router VM.  10. Copy and paste the below-provided script in **ncclient-netconf.py.** Press **Ctrl+S** to save it. Use a **for** loop and a **print** function to display the device capabilities.   |  | | --- | | print("#Supported Capabilities (YANG models):")  for capability in m.server\_capabilities:  print(capability) |     11. Execute following command: **python3 ncclient-netconf.py** to run Python script. The result is the same as delivering the complicated **hello** message earlier, but without the opening and closing <capability> XML tags on each line. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **// Use ncclient to Retrieve the Configuration**  1. To avoid showing capabilities output 400+ lines, comment out the block of lines that print the capabilities, as demonstrated in the screenshot below.    2. The Cisco IOS XE router settings may be retrieved using the **get\_config()** function of the **m** NETCONF session object. The **get\_config()** function requires a source string parameter that identifies the NETCONF datastore from which the configuration is retrieved. To see the results, use the print function. The **running** datastore is the only NETCONF datastore currently present on the Cisco IOS XE router. You may confirm this by using the **show netconf-yang datastores** command. Copy and paste the below-provided script in **ncclient-netconf.py.** Press **Ctrl+S** to save it.   |  | | --- | | netconf\_reply = m.get\_config(source="running")  print(netconf\_reply) |       3. Execute the following command: **python3 ncclient-netconf.py** to run the script. The output will be much longer than 100 lines, thus IDLE may compress it. Double-click the **Squeezed text** message in the IDLE shell window to enlarge the results.    4. Note that the resulting XML is not structured. To prettify the XML, copy and paste it into the same webpage you discovered previously.  5. Python has capabilities for working with XML files. To prettify the output, use the **toprettyxml()** method from the **xml.dom.minidom** module. Add a line at the start of your script to import the **xml.dom.minidom** module. Copy and paste the below-provided script in **ncclient-netconf.py.**   |  | | --- | | import xml.dom.minidom |   6. Replace the basic print function **print(netconf\_reply)** with one that outputs prettified XML output. Copy and paste the below-provided script in **ncclient-netconf.py.** Press **Ctrl+S** to save it.   |  | | --- | | print(xml.dom.minidom.parseString(netconf\_reply.xml).toprettyxml()) |     7. Execute the following command: **python3 ncclient-netconf.py** to run the script. XML is displayed in a more readable format.    8. A network administrator may simply want to obtain a subset of a device's current configuration. NETCONF can provide only data specified in a filter argument of the **get\_conf()** method. Create a variable named **netconf\_filter** to obtain only data described by the Cisco IOS XE Native YANG model. Copy and paste the below-provided script in **ncclient-netconf.py.** Press **Ctrl+S** to save it.   |  | | --- | | netconf\_filter = """  <filter xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">  </native>  </filter>  """  netconf\_reply = m.get\_config(source="running", filter=netconf\_filter)  print(xml.dom.minidom.parseString(netconf\_reply.xml).toprettyxml()) |     9. Execute the following command: **python3 ncclient-netconf.py** to run the script. The beginning of the output is the same, as illustrated below. However, just the <native> XML element is shown this time. Previously, all YANG models accessible for the Cisco IOS XE router were presented. Filtering the obtained data to show only the native YANG module greatly minimizes the output. This is because the native YANG module only contains a portion of the Cisco IOX XE YANG models. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **//** **Use ncclient to Configure a Device**  1. To change an existing setting in the Cisco IOS XE router configuration, extract the setting location from the previously received configuration. In this step, you will set a variable to modify the **<hostname>** value.  2. Previously, you defined the variable. To change a device configuration, create a variable. Copy and paste the below-provided script in **ncclient-netconf.py.** Press **Ctrl+S** to save it. You can use NEWHOSTNAME or any other hostname you choose.   |  | | --- | | netconf\_hostname = """  <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">  <hostname>NEWHOSTNAME</hostname>  </native>  </config>  """ |   3. Use the **m** NETCONF session object's **edit\_config()** method to submit the configuration and save the results in the **netconf\_reply** variable for printing. The arguments of the **edit\_config()** function are as follows:   * **target**: The targeted NETCONF datastore to be updated * **config:** The configuration modification that is to be sent   Copy and paste the below-provided script in **ncclient-netconf.py.**   |  | | --- | | netconf\_reply = m.edit\_config(target="running", config=netconf\_hostname) |   4. The **edit\_config()** method produces an XML RPC reply message containing **<ok/>**, indicating that the modification was successfully implemented. To display the results, simply repeat the preceding print command. Copy and paste the below-provided script in **ncclient-netconf.py.** Press **Ctrl+S** to save it.   |  | | --- | | print(xml.dom.minidom.parseString(netconf\_reply.xml).toprettyxml()) |     5. Execute the following command: **python3 ncclient-netconf.py** to run the script. You should obtain results similar to the one shown below. You may also check if the hostname has changed by switching to the Cisco IOS XE router VM.    6. Make a new <config> variable to store the configuration for a loopback interface. Copy and paste the below-provided script in **ncclient-netconf.py**. You can use whatever **description** you want. However, only use alphanumeric characters, or you will need to escape them with the backslash ( \ ). Copy and paste the below-provided script in **ncclient-netconf.py**.   |  | | --- | | netconf\_loopback = """  <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">  <interface>  <Loopback>  <name>1</name>  <description>My first NETCONF loopback</description>  <ip>  <address>  <primary>  <address>10.1.1.1</address>  <mask>255.255.255.0</mask>  </primary>  </address>  </ip>  </Loopback>  </interface>  </native>  </config>  """ |       7. Add the **edit\_config()** method to transmit the updated loopback settings to the Cisco IOS XE router and report the results. Copy and paste the below-provided script in **ncclient-netconf.py**. Press **Ctrl+S** to save it.   |  | | --- | | netconf\_reply = m.edit\_config(target="running", config=netconf\_loopback)  print(xml.dom.minidom.parseString(netconf\_reply.xml).toprettyxml()) |     8. Execute the following command: **python3 ncclient-netconf.py** to run the script. You should get output similar to the screenshot below.    9. Switch to Cisco IOS XE router and execute the following command: **show ip interface brief** to verify that the new loopback interface was created.    10. Create a new variable named **netconf\_newloop**. It will contain a configuration that generates a new loopback 2 interface but uses the same IPv4 address as loopback 1: 10.1.1.1/24. This would result in an error in the router CLI since it attempted to allocate a duplicate IP address to an interface. Copy and paste the below-provided script in **ncclient-netconf.py.**   |  | | --- | | netconf\_newloop = """  <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">  <interface>  <Loopback>  <name>2</name>  <description>My second NETCONF loopback</description>  <ip>  <address>  <primary>  <address>10.1.1.1</address>  <mask>255.255.255.0</mask>  </primary>  </address>  </ip>  </Loopback>  </interface>  </native>  </config>  """ |     11. Add the **edit\_config()** method to transmit the updated loopback settings to the Cisco IOS XE router and report the results. Copy and paste the below-provided script in **ncclient-netconf.py**. Press **Ctrl+S** to save it   |  | | --- | | netconf\_reply = m.edit\_config(target="running", config=netconf\_newloop) |     12. Execute the following command: **python3 ncclient-netconf.py** to run the script. You should get error output similar to the following with the RPCError message **Device refused one or more commands**.    13. NETCONF will not apply any of the configuration that is sent if one or more commands are rejected. To verify this, switch to the Cisco IOS XE router VM and execute the following command: **show ip interface brief**. Notice that your new interface was not created. |

|  |  |
| --- | --- |
| **// Complete Python Script**  The following is the whole Python script created in this lab, with no code commented out, so that you may execute the script without errors. Your script might look different. Modify the program to send alternative verification and configuration instructions to improve your Python abilities.   |  | | --- | | from ncclient import manager  import xml.dom.minidom  m = manager.connect(  host="Your\_IP\_Address",  port=830,  username="Your\_Username",  password="Your\_Password",  hostkey\_verify=False  )  print("#Supported Capabilities (YANG models):")  for capability in m.server\_capabilities:  print(capability)  netconf\_reply = m.get\_config(source="running")  print(xml.dom.minidom.parseString(netconf\_reply.xml).toprettyxml())  netconf\_filter = """  <filter xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native" />  </filter>  """  netconf\_reply = m.get\_config(source="running", filter=netconf\_filter)  print(xml.dom.minidom.parseString(netconf\_reply.xml).toprettyxml())  netconf\_hostname = """  <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">  <hostname>IOS XE1kv</hostname>  </native>  </config>  """  netconf\_reply = m.edit\_config(target="running", config=netconf\_hostname)  print(xml.dom.minidom.parseString(netconf\_reply.xml).toprettyxml())  netconf\_loopback = """  <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">  <interface>  <Loopback>  <name>1</name>  <description>My NETCONF loopback</description>  <ip>  <address>  <primary>  <address>10.1.1.1</address>  <mask>255.255.255.0</mask>  </primary>  </address>  </ip>  </Loopback>  </interface>  </native>  </config>  """  netconf\_reply = m.edit\_config(target="running", config=netconf\_loopback)  print(xml.dom.minidom.parseString(netconf\_reply.xml).toprettyxml())  netconf\_newloop = """  <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">  <interface>  <Loopback>  <name>2</name>  <description>My second NETCONF loopback</description>  <ip>  <address>  <primary>  <address>10.1.1.1</address>  <mask>255.255.255.0</mask>  </primary>  </address>  </ip>  </Loopback>  </interface>  </native>  </config>  """  netconf\_reply = m.edit\_config(target="running", config=netconf\_newloop) | |