2022

2.7 Lab – NETCONF w/Python: List Capabilities

Programación de Redes

UNIVERSIDAD TECNOLOGICA DEL NORTE DE GUANAJUATO

NOMBRE

JUAN PABLO PALMA APODERADO

CARRERA

TSU EN INFRAESTRUCTURA DE REDES DIGITALES

DOCENTE

BARRON RODRIGUEZ GABRIEL

NO. CONTROL

1221100259

FECHA

14 DE OCTUBRE DEL 2022

GRUPO

GIR0441

2.7 Lab – NETCONF w/Python: List Capabilities

Objectives

Part 1: Install the ncclient Python module

Part 2: Connect to IOS XE's NETCONF service using ncclient

Part 3: List the IOS XE's capabilities – supported YANG models

Background / Scenario

Working with NETCONF does not require working with raw NETCONF RPC messages and XML. In this lab you will learn how to use the ncclient Python module to easily interact with network devices using NETCONF. You will learn how to identify which YANG models are supported by the device. This information is helpful when building a production network automation system, that requires specific YANG models to be supported by the given network device.

Required Resources

- Access to a router with the IOS XE operating system version 16.6 or higher
- Python 3.x environment

Instructions

Part 1: Install the ncclient Python module

In this part, you will install ncclient module into your Python environment. ncclient is a python module that simplifies NETCONF operations with built in functions that deal with the XML messages and RPC calls.

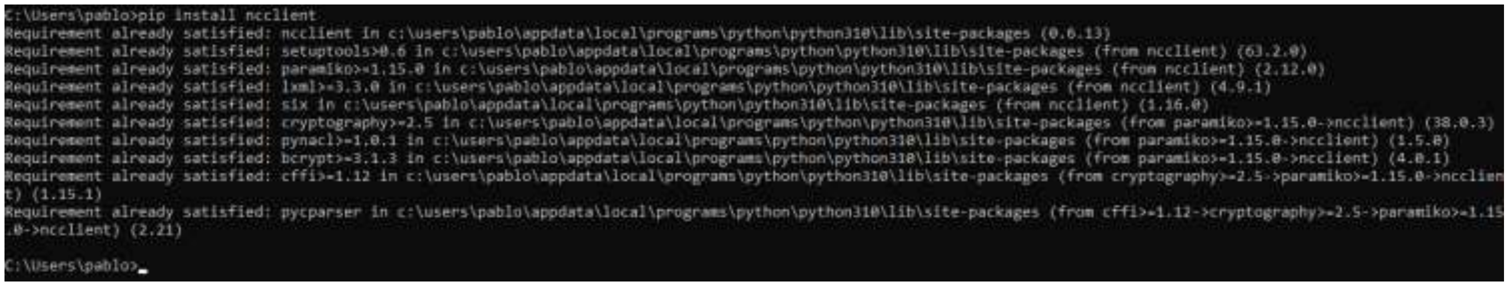
Explore the ncclient module on the project GitHub repository: https://github.com/ncclient/ncclient

Step 1: Use pip to install ncclient.

a. Start a new Windows command prompt (`cmd`).

b. Install ncclient using pip in the Windows command prompt:

```
pip install ncclient
```



```
C:\Users\pablo>pip install ncclient
Requirement already satisfied: ncclient in c:\users\pablo\appdata\local\programs\python\python310\lib\site-packages (0.6.13)
Requirement already satisfied: setuptools>0.6 in c:\users\pablo\appdata\local\programs\python\python310\lib\site-packages (from ncclient) (63.2.0)
Requirement already satisfied: paramiko>=1.15.0 in c:\users\pablo\appdata\local\programs\python\python310\lib\site-packages (from ncclient) (2.12.0)
Requirement already satisfied: lxml>=3.3.0 in c:\users\pablo\appdata\local\programs\python\python310\lib\site-packages (from ncclient) (4.9.1)
Requirement already satisfied: six in c:\users\pablo\appdata\local\programs\python\python310\lib\site-packages (from ncclient) (1.16.0)
Requirement already satisfied: cryptography>=2.5 in c:\users\pablo\appdata\local\programs\python\python310\lib\site-packages (from paramiko>=1.15.0->ncclient) (38.0.3)
Requirement already satisfied: pynacl>=1.0.1 in c:\users\pablo\appdata\local\programs\python\python310\lib\site-packages (from paramiko>=1.15.0->ncclient) (1.5.0)
Requirement already satisfied: bcrypt>=3.1.3 in c:\users\pablo\appdata\local\programs\python\python310\lib\site-packages (from paramiko>=1.15.0->ncclient) (4.0.1)
Requirement already satisfied: cffi>=1.12 in c:\users\pablo\appdata\local\programs\python\python310\lib\site-packages (from cryptography>=2.5->paramiko>=1.15.0->nccllen
t) (1.15.1)
Requirement already satisfied: pycparser in c:\users\pablo\appdata\local\programs\python\python310\lib\site-packages (from cffi>=1.12->cryptography>=2.5->paramiko>=1.15
.0->ncclient) (2.21)

C:\Users\pablo>_
```

c. Verify that ncclient has been successfully installed. Start Python IDLE and in the interactive shell try to import the ncclient module:

```
import ncclient
```

Part 2: Connect to IOS XE's NETCONF service using ncclient

Step 1: Connect to IOS XE's NETCONF service using ncclient.

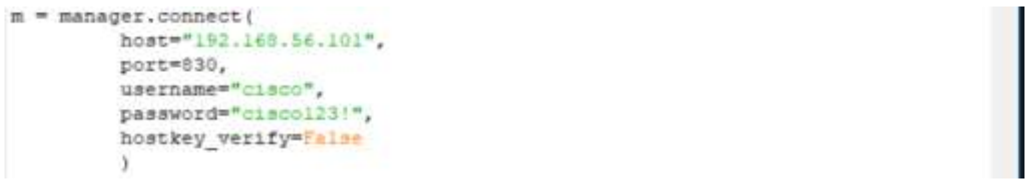
The ncclient module provides a "`manager`" class with "`connect()`" function to setup the remote NETCONF connection. After a successful connection, the returned object represents the NETCONF connection to the remote device.

a. In Python IDLE, create a new Python script file:

b. In the new Python script file editor, import the "`manager`" class from the ncclient module:

```
from ncclient import manager
```

c. Setup an `m` connection object using the `manager.connect()` function to the IOS XE device.



```
m = manager.connect(
        host="192.168.56.101",
        port=830,
        username="cisco",
        password="cisco123!",
        hostkey_verify=False
        )
```

The parameters of the `manager.connect()` function are:

- `host` – the address (host or IP) of the remote device (adjust the IP address to match the router's current address)
- `port` – the remote port of the NETCONF service
- `username` – remote ssh username (in this lab "cisco" for that was setup in the IOS XE VM)
- `password` – remote ssh password (in this lab "cisco123!" for that was setup in the IOS XE VM)
- `hostkey_verify` – whether to verify the ssh fingerprint (in lab it is safe to set to False, in production environments you should always verify the ssh fingerprints)

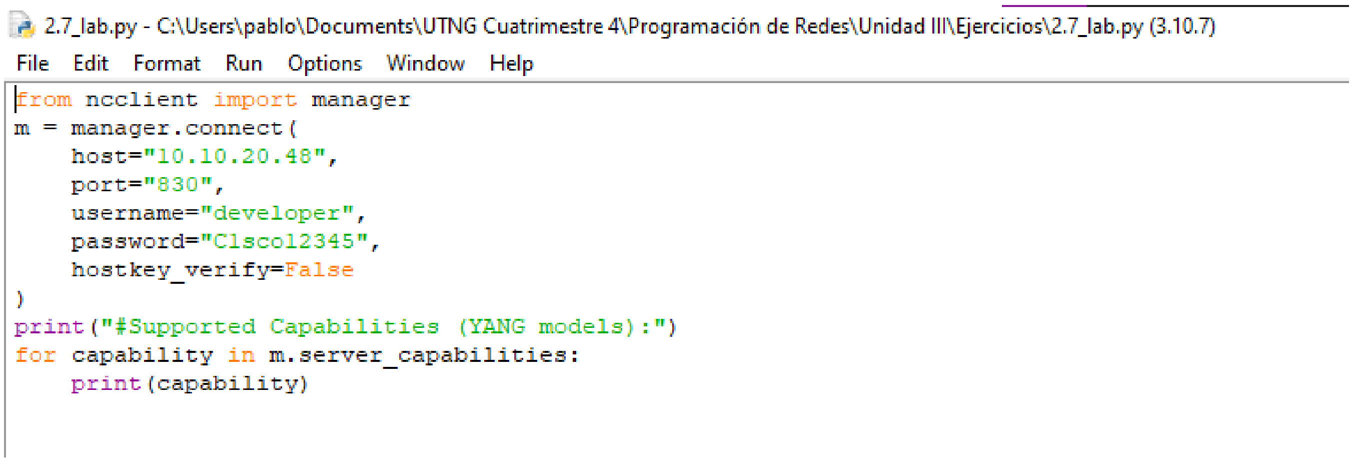
Part 3: List the IOS XE's capabilities – supported YANG models

Step 1: Send show commands and display the output

a. The `m` object, returned by the `manager.connect()` function that represents the NETCONF remote session. In every NETCONF session, the server first sends its list of capabilities – supported YANG models. With the ncclient module, the received list of capabilities is stored in the `m.server_capabilities` list.

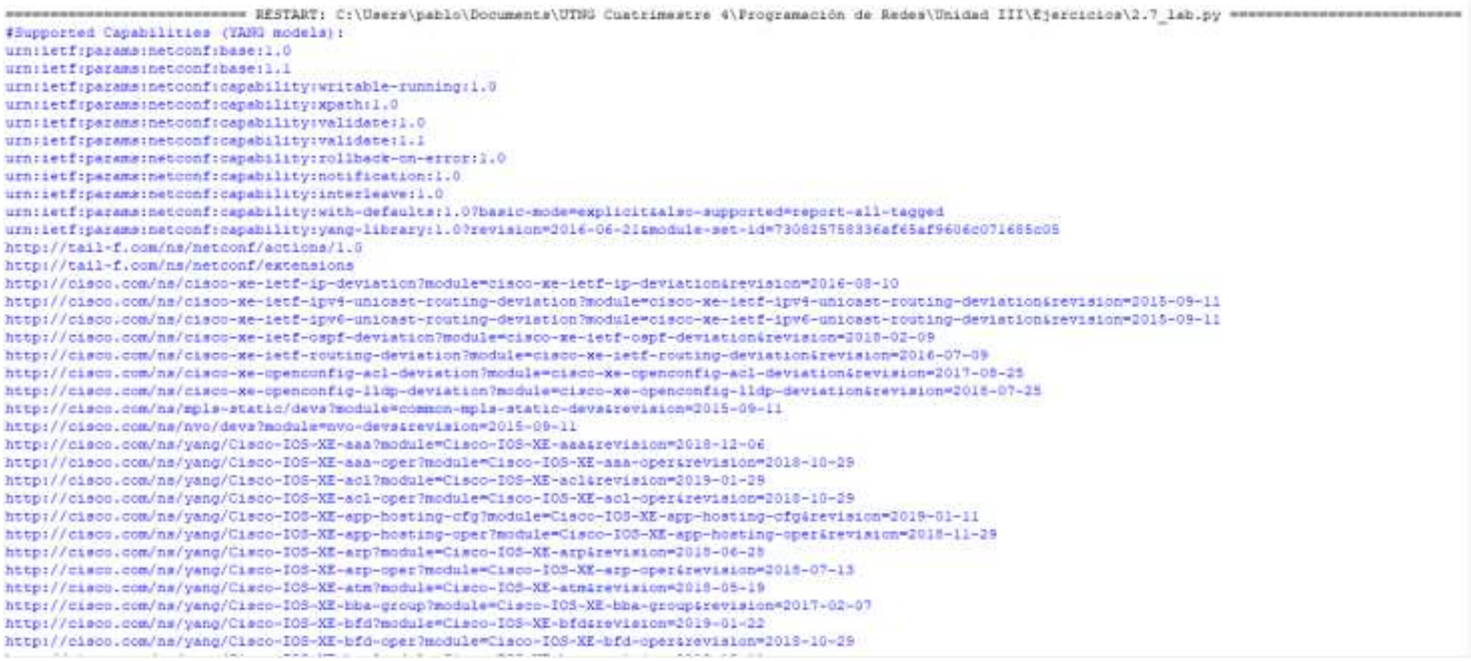
b. Use a for loop and a print function to print the device capabilities:

```python
print("#Supported Capabilities (YANG models):")
for capability in m.server_capabilities:
    print(capability)
```



2.7_lab.py - C:\Users\pablo\Documents\UTNG Cuatrimestre 4\Programación de Redes\Unidad III\Ejercicios\2.7_lab.py (3.10.7)

```python
from ncclient import manager
m = manager.connect(
    host="10.10.20.48",
    port="830",
    username="developer",
    password="C1sco12345",
    hostkey_verify=False
)
print("#Supported Capabilities (YANG models):")
for capability in m.server_capabilities:
    print(capability)
```



```
============================= RESTART: C:\Users\pablo\Documents\UTNG Cuatrimestre 4\Programación de Redes\Unidad III\Ejercicios\2.7_lab.py ============================
#Supported Capabilities (YANG models):
urn:ietf:params:netconf:base:1.0
urn:ietf:params:netconf:base:1.1
urn:ietf:params:netconf:capability:writable-running:1.0
urn:ietf:params:netconf:capability:xpath:1.0
urn:ietf:params:netconf:capability:validate:1.0
urn:ietf:params:netconf:capability:validate:1.1
urn:ietf:params:netconf:capability:rollback-on-error:1.0
urn:ietf:params:netconf:capability:notification:1.0
urn:ietf:params:netconf:capability:interleave:1.0
urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=explicit&also-supported=report-all-tagged
urn:ietf:params:netconf:capability:yang-library:1.0?revision=2016-06-21&module-set-id=730825758336af65af9606c071685c05
http://tail-f.com/ns/netconf/actions/1.0
http://tail-f.com/ns/netconf/extensions
http://cisco.com/ns/cisco-xe-ietf-ip-deviation?module=cisco-xe-ietf-ip-deviation&revision=2016-08-10
http://cisco.com/ns/cisco-xe-ietf-ipv4-unicast-routing-deviation?module=cisco-xe-ietf-ipv4-unicast-routing-deviation&revision=2015-09-11
http://cisco.com/ns/cisco-xe-ietf-ipv6-unicast-routing-deviation?module=cisco-xe-ietf-ipv6-unicast-routing-deviation&revision=2015-09-11
http://cisco.com/ns/cisco-xe-ietf-ospf-deviation?module=cisco-xe-ietf-ospf-deviation&revision=2018-02-09
http://cisco.com/ns/cisco-xe-ietf-routing-deviation?module=cisco-xe-ietf-routing-deviation&revision=2016-07-09
http://cisco.com/ns/cisco-xe-openconfig-acl-deviation?module=cisco-xe-openconfig-acl-deviation&revision=2017-08-25
http://cisco.com/ns/cisco-xe-openconfig-lldp-deviation?module=cisco-xe-openconfig-lldp-deviation&revision=2018-07-25
http://cisco.com/ns/mpls-static/devs?module=common-mpls-static-devs&revision=2015-09-11
http://cisco.com/ns/nvo/devs?module=nvo-devs&revision=2015-09-11
http://cisco.com/ns/yang/Cisco-IOS-XE-aaa?module=Cisco-IOS-XE-aaa&revision=2018-12-06
http://cisco.com/ns/yang/Cisco-IOS-XE-aaa-oper?module=Cisco-IOS-XE-aaa-oper&revision=2018-10-29
http://cisco.com/ns/yang/Cisco-IOS-XE-acl?module=Cisco-IOS-XE-acl&revision=2019-01-29
http://cisco.com/ns/yang/Cisco-IOS-XE-acl-oper?module=Cisco-IOS-XE-acl-oper&revision=2018-10-29
http://cisco.com/ns/yang/Cisco-IOS-XE-app-hosting-cfg?module=Cisco-IOS-XE-app-hosting-cfg&revision=2019-01-11
http://cisco.com/ns/yang/Cisco-IOS-XE-app-hosting-oper?module=Cisco-IOS-XE-app-hosting-oper&revision=2018-11-29
http://cisco.com/ns/yang/Cisco-IOS-XE-arp?module=Cisco-IOS-XE-arp&revision=2018-06-28
http://cisco.com/ns/yang/Cisco-IOS-XE-arp-oper?module=Cisco-IOS-XE-arp-oper&revision=2018-07-13
http://cisco.com/ns/yang/Cisco-IOS-XE-atm?module=Cisco-IOS-XE-atm&revision=2018-05-19
http://cisco.com/ns/yang/Cisco-IOS-XE-bba-group?module=Cisco-IOS-XE-bba-group&revision=2017-02-07
http://cisco.com/ns/yang/Cisco-IOS-XE-bfd?module=Cisco-IOS-XE-bfd&revision=2019-01-22
http://cisco.com/ns/yang/Cisco-IOS-XE-bfd-oper?module=Cisco-IOS-XE-bfd-oper&revision=2018-10-29
```

c. Execute the Python script file to see the results.

d. Is the Cisco-IOS-XE-cdp YANG model supported by the device?

Type your answer here

End of Document

Conclusion

En la practica de este laboratorio ejecutaremos nos mostrara una lista de los sistemas que cisco admite dentro del router. En el codijo nos conectamos todo los necesario para que pueda conectarse y poder ejecutar el comando `capabilities` para después mostrarlo en l consola y esto es por medio de nnclient que moniterea la codificación de XMl del router y después buscar coincidencias del sistema.

¿Qué es *ncclient* ?

ncclient es una biblioteca de Python para clientes NETCONF. Su objetivo es ofrecer una API intuitiva que mapee con sensatez la naturaleza codificada en XML de NETCONF a las construcciones y modismos de Python, y facilite la escritura de scripts de administración de red.

Caracteristicas de ncclient

- Solicitud de canalización.
- Solicitudes RPC asíncronas.
- Mantener XML fuera del camino a menos que sea realmente necesario.
- Extensible. Se pueden agregar fácilmente nuevas asignaciones de transporte y capacidades/operaciones.

Los tipos de modelos soportados

- CSR: device_params={'nombre':'csr'}
- Nexo: device_params={'nombre':'nexo'}
- IOS XR: device_params={'nombre':'iosxr'}
- IOS XE: device_params={'nombre':'iosxe'}

¿Que hace la siguiente parte del codijo `m.server_capabilities?`

Dado un URI que tiene una cadena de consulta de esquema (es decir , :url URI de capacidad), devolverá una lista de esquemas admitidos.

Representa el conjunto de capacidades disponibles para un cliente o servidor NETCONF. Se inicializa con una lista de URI de capacidad.

`¿para que sirve manager.connect()?`

Permite a un administrador crear un paquete de configuración de acceso remoto que se distribuya a los usuarios remotos del administrador.