

## Lab – NETCONF w/Python: List Capabilities

Gerardo Antonio Garcia Vazquez – 4 de diciembre de 2023

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

```
import ncclient
```

```

dukogg@dukogg: ~
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
dukogg@dukogg:~$ pip install ncclient
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: ncclient in ~/.local/lib/python3.10/site-packages (0.6.13)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from ncclient) (1.16.0)
Requirement already satisfied: paramiko>=1.15.0 in ~/.local/lib/python3.10/site-packages (from ncclient) (3.3.1)
Requirement already satisfied: lxml>=3.3.0 in ~/.local/lib/python3.10/site-packages (from ncclient) (4.9.3)
Requirement already satisfied: setuptools>0.6 in /usr/lib/python3/dist-packages (from ncclient) (59.6.0)
Requirement already satisfied: cryptography>=3.3 in /usr/lib/python3/dist-packages (from paramiko>=1.15.0->ncclient) (3.4.8)
Requirement already satisfied: pynacl>=1.5 in /usr/lib/python3/dist-packages (from paramiko>=1.15.0->ncclient) (1.5.0)
Requirement already satisfied: bcrypt>=3.2 in ~/.local/lib/python3.10/site-packages (from paramiko>=1.15.0->ncclient) (4.0.1)
Requirement already satisfied: cffi>=1.4.1 in ~/.local/lib/python3.10/site-packages (from pynacl>=1.5->paramiko>=1.15.0->ncclient) (1.16.0)
Requirement already satisfied: pycparser in ~/.local/lib/python3.10/site-packages (from cffi>=1.4.1->pynacl>=1.5->paramiko>=1.15.0->ncclient) (2.21)
dukogg@dukogg:~$

```

## 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.

- In Python IDLE, create a new Python script file:
- In the new Python script file editor, import the “manager” class from the ncclient module:

```
from ncclient import manager
```

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

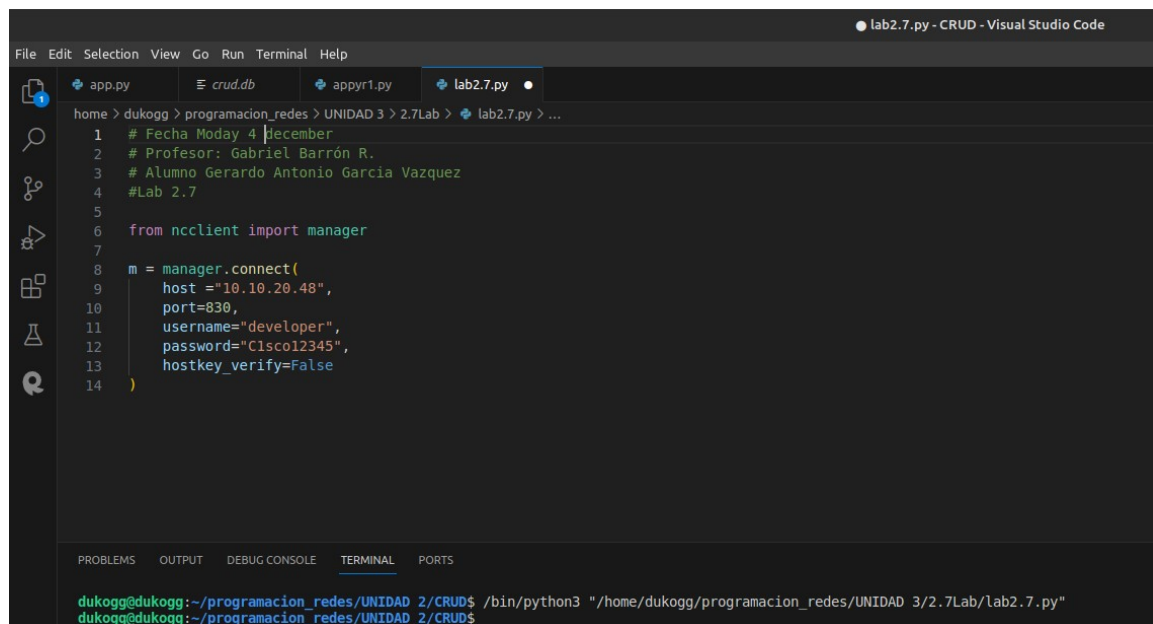
```

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)



The screenshot shows a Visual Studio Code editor window titled "lab2.7.py - CRUD - Visual Studio Code". The editor displays a Python script with the following content:

```
1 # Fecha Moday 4 December
2 # Profesor: Gabriel Barrón R.
3 # Alumno Gerardo Antonio Garcia Vazquez
4 #Lab 2.7
5
6 from ncclient import manager
7
8 m = manager.connect(
9     host="10.10.20.48",
10    port=830,
11    username="developer",
12    password="C1sco12345",
13    hostkey_verify=False
14 )
```

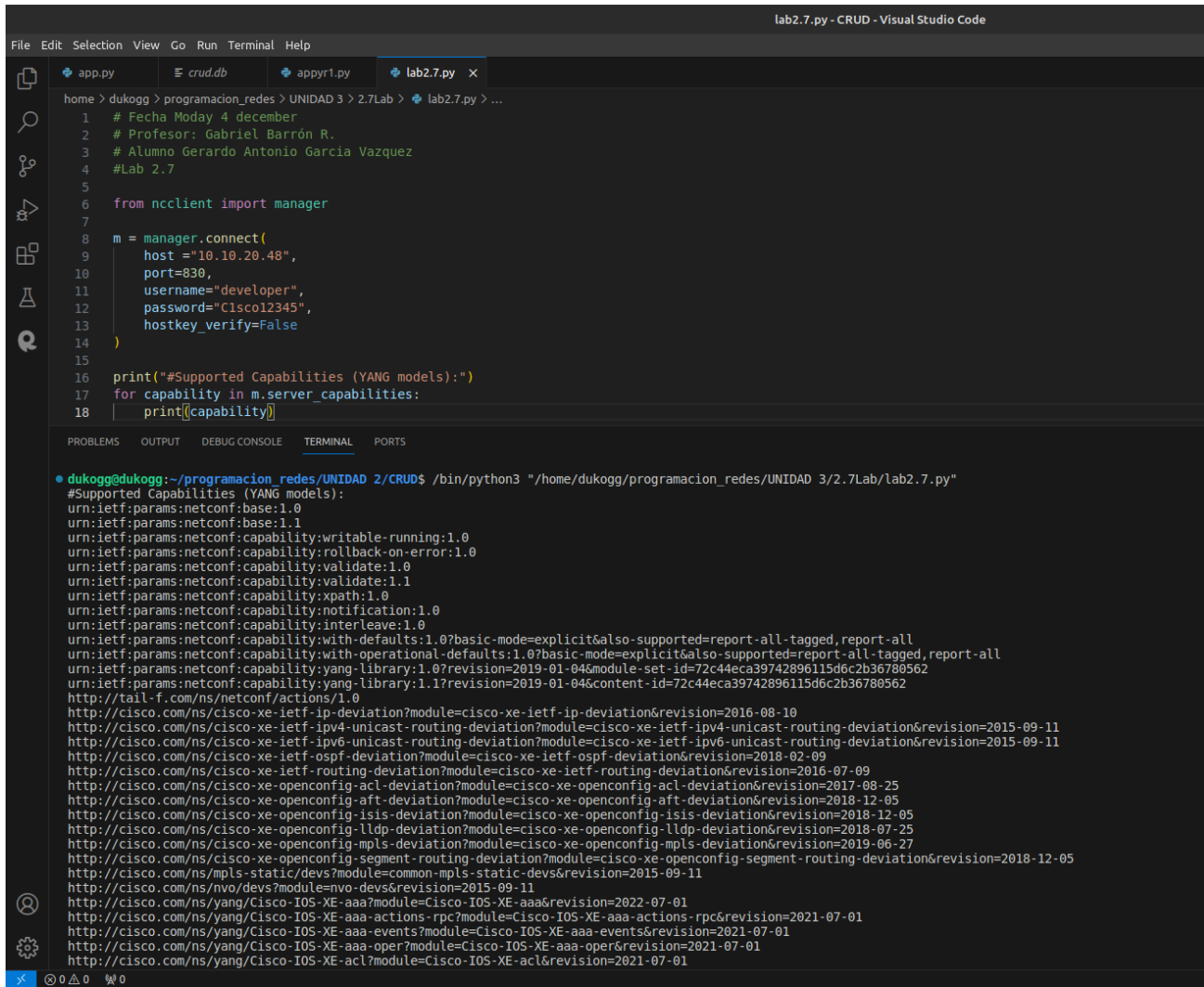
The terminal at the bottom shows the command prompt for a user named "dukogg" in the directory `~/programacion_redes/UNIDAD 2/CRUD`, running the script `lab2.7.py` using Python 3.

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

### Step 1: Send show commands and display the output

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

```
print("#Supported Capabilities (YANG models):")
for capability in m.server_capabilities:
    print(capability)
```
- Execute the Python script file to see the results.



The screenshot shows a Visual Studio Code editor with a file named `lab2.7.py` open. The script uses the `ncclient` library to connect to a device and list its supported YANG capabilities. The terminal output shows the connection details and a list of capabilities, including various Cisco IOS-XE modules.

```

1 # Fecha Moday 4 december
2 # Profesor: Gabriel Barrón R.
3 # Alumno Gerardo Antonio Garcia Vazquez
4 #Lab 2.7
5
6 from ncclient import manager
7
8 m = manager.connect(
9     host="10.10.20.48",
10    port=830,
11    username="developer",
12    password="C1scol2345",
13    hostkey_verify=False
14 )
15
16 print("#Supported Capabilities (YANG models):")
17 for capability in m.server_capabilities:
18     print(capability)

```

```

• dukogg@dukogg:~/programacion_redes/UNIDAD 2/CRUD$ /bin/python3 "/home/dukogg/programacion_redes/UNIDAD 3/2.7Lab/lab2.7.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:rollback-on-error:1.0
urn:ietf:params:netconf:capability:validate:1.0
urn:ietf:params:netconf:capability:validate:1.1
urn:ietf:params:netconf:capability:xpath: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,report-all
urn:ietf:params:netconf:capability:with-operational-defaults:1.0?basic-mode=explicit&also-supported=report-all-tagged,report-all
urn:ietf:params:netconf:capability:yang-library:1.0?revision=2019-01-04&module-set-id=72c44eca39742896115d6c2b36780562
urn:ietf:params:netconf:capability:yang-library:1.1?revision=2019-01-04&content-id=72c44eca39742896115d6c2b36780562
http://tail-f.com/ns/netconf/actions/1.0
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-aft-deviation?module=cisco-xe-openconfig-aft-deviation&revision=2018-12-05
http://cisco.com/ns/cisco-xe-openconfig-isis-deviation?module=cisco-xe-openconfig-isis-deviation&revision=2018-12-05
http://cisco.com/ns/cisco-xe-openconfig-ldp-deviation?module=cisco-xe-openconfig-ldp-deviation&revision=2018-07-25
http://cisco.com/ns/cisco-xe-openconfig-mpls-deviation?module=cisco-xe-openconfig-mpls-deviation&revision=2019-06-27
http://cisco.com/ns/cisco-xe-openconfig-segment-routing-deviation?module=cisco-xe-openconfig-segment-routing-deviation&revision=2018-12-05
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=2022-07-01
http://cisco.com/ns/yang/Cisco-IOS-XE-aaa-actions-rpc?module=Cisco-IOS-XE-aaa-actions-rpc&revision=2021-07-01
http://cisco.com/ns/yang/Cisco-IOS-XE-aaa-events?module=Cisco-IOS-XE-aaa-events&revision=2021-07-01
http://cisco.com/ns/yang/Cisco-IOS-XE-aaa-oper?module=Cisco-IOS-XE-aaa-oper&revision=2021-07-01
http://cisco.com/ns/yang/Cisco-IOS-XE-acl?module=Cisco-IOS-XE-acl&revision=2021-07-01

```

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