

**NOMBRE:** Isidro Lara Lopez

**CARRERA:** INFRAESTRUCTURA DE REDES DIGITALES

**DOCENTE:** BARRON RODRIGUEZ GABRIEL

**NO. CONTROL:** 1221100381

**FECHA:** 14 DE DICIEMBRE DEL 2022

**GRUPO:** GIR0441

## 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

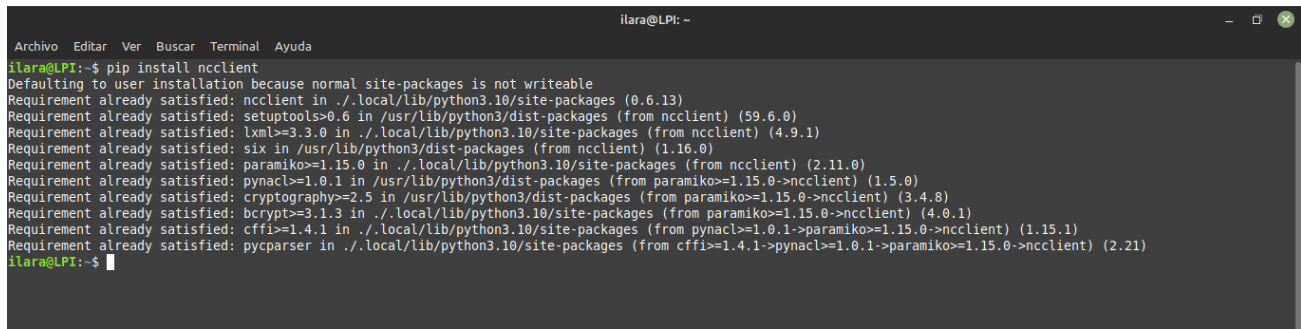
In this part, you will install ncclient module into your Python environment. ncclient is a python module that simplifies NETCONF operations with built in functions that deal with the XML messages and RPC calls.

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
```

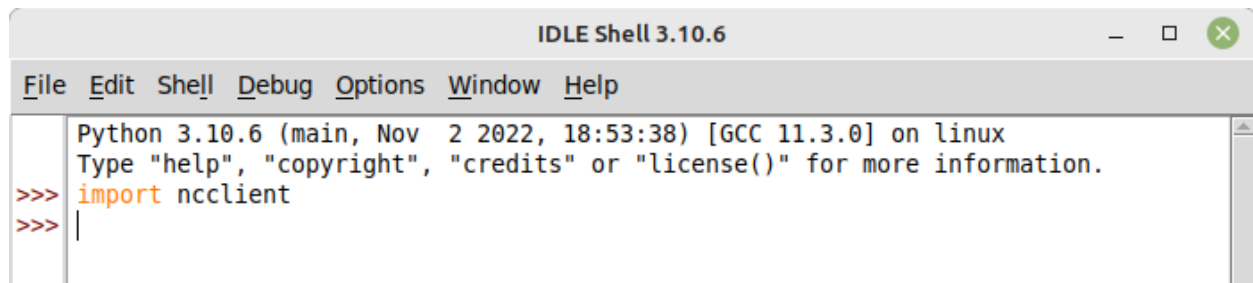


```
ilara@LPI: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
ilara@LPI:~$ pip install ncclient  
Defaulting to user installation because normal site-packages is not writeable  
Requirement already satisfied: ncclient in ./local/lib/python3.10/site-packages (0.6.13)  
Requirement already satisfied: setuptools>=0.6 in /usr/lib/python3/dist-packages (from ncclient) (59.6.0)  
Requirement already satisfied: lxml>=3.3.0 in ./local/lib/python3.10/site-packages (from ncclient) (4.9.1)  
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) (2.11.0)  
Requirement already satisfied: pynacl>=1.0.1 in /usr/lib/python3/dist-packages (from paramiko>=1.15.0->ncclient) (1.5.0)  
Requirement already satisfied: cryptography>=2.5 in /usr/lib/python3/dist-packages (from paramiko>=1.15.0->ncclient) (3.4.8)  
Requirement already satisfied: bcrypt>=3.1.3 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>=1.0.1->paramiko>=1.15.0->ncclient) (1.15.1)  
Requirement already satisfied: pycparser in ./local/lib/python3.10/site-packages (from cffi>=1.4.1->pynacl>=1.0.1->paramiko>=1.15.0->ncclient) (2.21)  
ilara@LPI:~$
```

Aquí instalamos el módulo `ncclient` en nuestra laptop con el comando anterior en una terminal.

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

```
import ncclient
```



```
IDLE Shell 3.10.6  
File Edit Shell Debug Options Window Help  
Python 3.10.6 (main, Nov 2 2022, 18:53:38) [GCC 11.3.0] on linux  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> import ncclient  
>>> |
```

en este paso importamos el módulo `ncclient` en el idle de Python para crear el script y que funcione.

## 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.

A screenshot of a Python script editor window titled "2.7 - /home/ilara/2.7 (3.10.6)". The script contains the following code:

```
from ncclient import manager
m = manager.connect(
    host="10.10.20.48",
    port=830,
    username="developer",
    password="Cisco12345",
    hostkey_verify=False
)
print("#Supported Capabilities (YANG models):")
for capability in m.server_capabilities:
    print(capability)
```

En este paso creamos el script solicitado cambiándole

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

- 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.

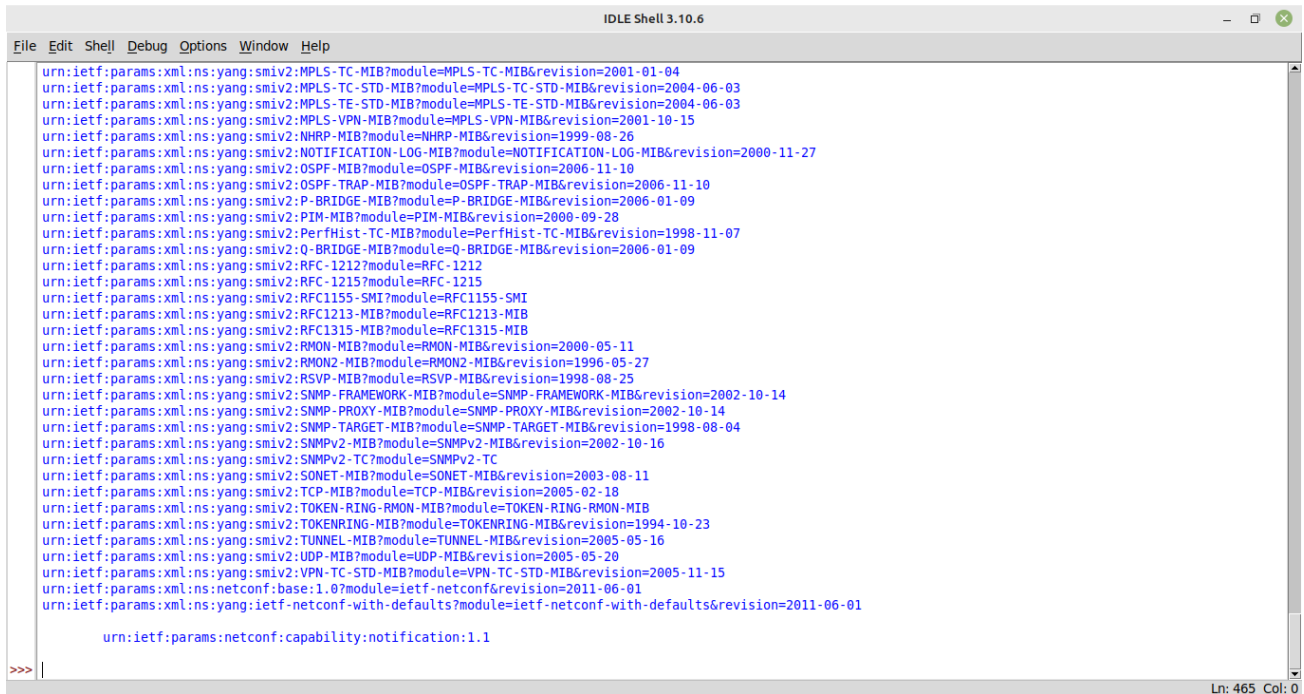
- b. Use a for loop and a print function to print the device capabilities:

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

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



```
IDLE Shell 3.10.6
File Edit Shell Debug Options Window Help
Python 3.10.6 (main, Nov 2 2022, 18:53:38) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> import ncclient
>>>
===== RESTART: /home/ilara/2.7 =====
#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: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=736825758336af65af9606c071685c05
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-ldp-deviation?module=cisco-xe-openconfig-ldp-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
Ln: 465 Col: 0
```



```
urn:ietf:params:xml:ns:yang:smiv2:MPLS-TC-MIB?module=MPLS-TC-MIB&revision=2001-01-04
urn:ietf:params:xml:ns:yang:smiv2:MPLS-TC-STD-MIB?module=MPLS-TC-STD-MIB&revision=2004-06-03
urn:ietf:params:xml:ns:yang:smiv2:MPLS-TE-STD-MIB?module=MPLS-TE-STD-MIB&revision=2004-06-03
urn:ietf:params:xml:ns:yang:smiv2:MPLS-VPN-MIB?module=MPLS-VPN-MIB&revision=2001-10-15
urn:ietf:params:xml:ns:yang:smiv2:NHRP-MIB?module=NHRP-MIB&revision=1999-08-26
urn:ietf:params:xml:ns:yang:smiv2:NOTIFICATION-LOG-MIB?module=NOTIFICATION-LOG-MIB&revision=2000-11-27
urn:ietf:params:xml:ns:yang:smiv2:OSPF-MIB?module=OSPF-MIB&revision=2006-11-10
urn:ietf:params:xml:ns:yang:smiv2:OSPF-TRAP-MIB?module=OSPF-TRAP-MIB&revision=2006-11-10
urn:ietf:params:xml:ns:yang:smiv2:P-BRIDGE-MIB?module=P-BRIDGE-MIB&revision=2006-01-09
urn:ietf:params:xml:ns:yang:smiv2:PIM-MIB?module=PIM-MIB&revision=2000-09-28
urn:ietf:params:xml:ns:yang:smiv2:PerfHist-TC-MIB?module=PerfHist-TC-MIB&revision=1998-11-07
urn:ietf:params:xml:ns:yang:smiv2:Q-BRIDGE-MIB?module=Q-BRIDGE-MIB&revision=2006-01-09
urn:ietf:params:xml:ns:yang:smiv2:RFC-1212?module=RFC-1212
urn:ietf:params:xml:ns:yang:smiv2:RFC-1215?module=RFC-1215
urn:ietf:params:xml:ns:yang:smiv2:RFC1155-SMI?module=RFC1155-SMI
urn:ietf:params:xml:ns:yang:smiv2:RFC1213-MIB?module=RFC1213-MIB
urn:ietf:params:xml:ns:yang:smiv2:RFC1315-MIB?module=RFC1315-MIB
urn:ietf:params:xml:ns:yang:smiv2:RMON-MIB?module=RMON-MIB&revision=2000-05-11
urn:ietf:params:xml:ns:yang:smiv2:RMON2-MIB?module=RMON2-MIB&revision=1996-05-27
urn:ietf:params:xml:ns:yang:smiv2:RSVP-MIB?module=RSVP-MIB&revision=1998-08-25
urn:ietf:params:xml:ns:yang:smiv2:SNMP-FRAMEWORK-MIB?module=SNMP-FRAMEWORK-MIB&revision=2002-10-14
urn:ietf:params:xml:ns:yang:smiv2:SNMP-PROXY-MIB?module=SNMP-PROXY-MIB&revision=2002-10-14
urn:ietf:params:xml:ns:yang:smiv2:SNMP-TARGET-MIB?module=SNMP-TARGET-MIB&revision=1998-08-04
urn:ietf:params:xml:ns:yang:smiv2:SNMPv2-MIB?module=SNMPv2-MIB&revision=2002-10-16
urn:ietf:params:xml:ns:yang:smiv2:SNMPv2-TC?module=SNMPv2-TC
urn:ietf:params:xml:ns:yang:smiv2:SONET-MIB?module=SONET-MIB&revision=2003-08-11
urn:ietf:params:xml:ns:yang:smiv2:TCP-MIB?module=TCP-MIB&revision=2005-02-18
urn:ietf:params:xml:ns:yang:smiv2:TOKEN-RING-RMON-MIB?module=TOKEN-RING-RMON-MIB
urn:ietf:params:xml:ns:yang:smiv2:TOKENRING-MIB?module=TOKENRING-MIB&revision=1994-10-23
urn:ietf:params:xml:ns:yang:smiv2:TUNNEL-MIB?module=TUNNEL-MIB&revision=2005-05-16
urn:ietf:params:xml:ns:yang:smiv2:UDP-MIB?module=UDP-MIB&revision=2005-05-20
urn:ietf:params:xml:ns:yang:smiv2:VPN-TC-STD-MIB?module=VPN-TC-STD-MIB&revision=2005-11-15
urn:ietf:params:xml:ns:netconf:base:1.0?module=ietf-netconf&revision=2011-06-01
urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults?module=ietf-netconf-with-defaults&revision=2011-06-01

urn:ietf:params:netconf:capability:notification:1.1
```

observamos los resultados que genero el script.

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

**nccliente:** es una biblioteca de Python para clientes NETCONF. Su objetivo es ofrecer una API intuitiva que mapee sensiblemente la naturaleza codificada por XML de NETCONF a las construcciones y expresiones idiomáticas de Python, y facilitar la escritura de scripts de administración de red.

características de ncclient son:

Admite todas las operaciones y capacidades definidas en RFC 6241.

Solicitar canalización.

Solicitudes de 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.

El Protocolo de configuración de red (NETCONF) es un protocolo de gestión de red con base en XML que da un procedimiento programable para configurar y regir dispositivos de red. NETCONF ha sido determinado en RFC 4741 por el Conjunto laboral de ingeniería de Internet (IETF) y inspeccionado en RFC 6241.

NETCONF da estándares por medio de los cuales los administradores de red y los desarrolladores de aplicaciones tienen la posibilidad de regir configuraciones de dispositivos de red y obtener el estado del dispositivo de red inmediatamente.

Los paquetes NETCONF permanecen en formato XML y el protocolo NETCONF tiene una poderosa capacidad de filtrado. Cada campo de datos tiene un nombre de factor fijo y una postura. Por lo que los dispositivos del mismo abastecedor tienen la posibilidad de usar el mismo modo de ingreso y modo de visualización de resultados. Los dispositivos de diferentes proveedores tienen la posibilidad de conseguir el mismo impacto por medio del mapeo XML. Esta funcionalidad permite el desarrollo de programa de terceros y la personalización del programa NMS en el ámbito de diversas proveedores y dispositivos, dicho programa NMS, NETCONF simplifica la configuración del dispositivo y optimización la eficiencia de la configuración del dispositivo.

En esta práctica de laboratorio, aprenderá a usar el módulo de Python ncclient para interactuar fácilmente con dispositivos de red mediante NETCONF.

ncclient proporciona una clase de "administrador" con la función de conectar () para configurar la conexión NETCONF remota. Después de una conexión exitosa, el objeto devuelto representa la conexión NETCONF al dispositivo que estamos conectados remotamente.

El objeto devuelto por la función `manager.connect()` que representa la sesión remota de NETCONF. En cada sesión de NETCONF, el servidor primero envía su lista de capacidades: modelos YANG admitidos. Con el módulo ncclient, la lista de capacidades recibida se almacena en la lista `m.server_capabilities`.