



# UNIVERSIDAD TECNOLÓGICA DEL NORTE DE GUANAJUATO

Tecnologías de la Información y Comunicación  
**Programa educativo:**

TSU en Infraestructura de Redes Digitales  
**Área académica:**

Programación de Redes  
**Asignatura:**

**Unidad III:** Programación de Redes

**Grupo:** GIR0441

NETCONF w/Python: Get Operational Data  
**Laboratorio 9:**

Venado Soria German Emiliano  
**Alumno:**

Gabriel Barrón Rodríguez  
**Docente:**

Dolores Hidalgo, C.I.N., Gto., Miércoles 14 de Diciembre de 2022  
**Lugar y fecha:**

## Lab – NETCONF w/Python: Get Operational Data

### Objectives

#### Part 1: Retrieve the IOS XE VM's Operational Data - Statistics

### Background / Scenario

In this lab, you will learn how to use NETCONF to retrieve operational data from the network device.

### Required Resources

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

### Instructions

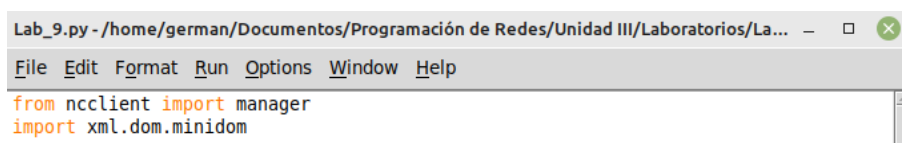
#### Part 1: Retrieve Interface Statistics

In this part, you will use the ncclient module to retrieve the device's operational data. The data are returned back in XML form. In the following steps this data will be transformed into a tabular output.

#### Step 1: Use ncclient to retrieve the device's running configuration.

- In Python IDLE, create a new Python script file:
- In the new Python script file editor, import the "manager" class from the ncclient module and the xml.dom.minidom module:

```
from ncclient import manager
import xml.dom.minidom
```



En un nuevo archivo Python en IDLE comenzamos a importar los módulos con los que hemos venido trabajando en los laboratorios anteriores.

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

```
Lab_9.py - /home/german/Documentos/
File Edit Format Run Options W
from ncclient import manager
import xml.dom.minidom

m = manager.connect(
    host= "10.10.20.48",
    port= 830,
    username= "developer",
    password= "Cisco12345",
    hostkey_verify= False
)
```

También proseguimos con la configuración de la sesión para la conexión remota.

- d. After a successful NETCONF connection, using the “get ()” function of the “m” NETCONF session object to retrieve and print the device’s operational data. The get () function expects a “filter” string parameter that defines the NETCONF filter.

The following filter retrieves the interfaces-state operational data (statistics), as defined in the ietf-interfaces YANG model:

**Note:** Executing the get() function without a filter is similar to execute “debug all”.

```
netconf_filter = """
<filter>
  <interfaces-state xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"/>
</filter>
"""

netconf_reply = m.get(filter = netconf_filter)
print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
```

```
Lab_9.py - /home/german/Documentos/Programación de Redes/Unidad III/Laboratorios/Lab...
File Edit Format Run Options Window Help

from ncclient import manager
import xml.dom.minidom

m = manager.connect(
    host= "10.10.20.48",
    port= 830,
    username= "developer",
    password= "C1sc0l2345",
    hostkey_verify= False
)

netconf_filter = """
<filter>
  <interfaces-state xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"/>
</filter>
"""

netconf_reply = m.get(filter = netconf_filter)
'''print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())'''
```

Empleamos la función “get ()” para devolver y mostrar los datos operativos del dispositivo, agregando un parámetro de cadena de “filter” que indica el filtro NETCONF, el cual recupera los datos operativos del estado de las interfaces (estadísticas), tal como se define en el modelo YANG de interfaces ietf.

- e. Execute the Python script and explore the output.

```
= RESTART: /home/german/Documentos/Programación de Redes/Unidad III/Laboratorios/Lab-9/Lab_9.py
<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:102eeac6-6971-420b-b003-f2817014a149">
  <data>
    <interfaces-state xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>GigabitEthernet1</name>
        <type xmlns:ianaif="urn:ietf:params:xml:ns:yang:iana-if-type">ianaif:ethernetCsmacd</type>
        <admin-status>up</admin-status>
        <oper-status>up</oper-status>
        <last-change>2022-12-15T16:51:17.000776+00:00</last-change>
        <if-index>1</if-index>
        <phys-address>00:50:56:bf:75:bb</phys-address>
        <speed>1024000000</speed>
        <statistics>
          <discontinuity-time>2022-12-15T16:40:59.000399+00:00</discontinuity-time>
          <in-octets>15319</in-octets>
          <in-unicast-pkts>132</in-unicast-pkts>
          <in-broadcast-pkts>0</in-broadcast-pkts>
          <in-multicast-pkts>0</in-multicast-pkts>
          <in-discards>0</in-discards>
          <in-errors>0</in-errors>
          <in-unknown-protos>0</in-unknown-protos>
          <out-octets>88078</out-octets>
          <out-unicast-pkts>214</out-unicast-pkts>
          <out-broadcast-pkts>0</out-broadcast-pkts>
          <out-multicast-pkts>0</out-multicast-pkts>
          <out-discards>0</out-discards>
          <out-errors>0</out-errors>
        </statistics>
      </interface>
    </interfaces-state>
  </data>
</rpc-reply>
```

La salida que nos da es exitosa, nos muestra el estado de las interfaces del Router.

- f. Convert the XML netconf\_reply data to a Python dictionary using the “xmldict” module. You can use a simple **for** loop to print a summary view of the statistical data:

```
import xmldict

netconf_reply_dict = xmldict.parse(netconf_reply.xml)

for interface in netconf_reply_dict["rpc-reply"]["data"]["interfaces-state"]["interface"]:

    print("Name: {} MAC: {} Input: {} Output {}".format(

        interface["name"],

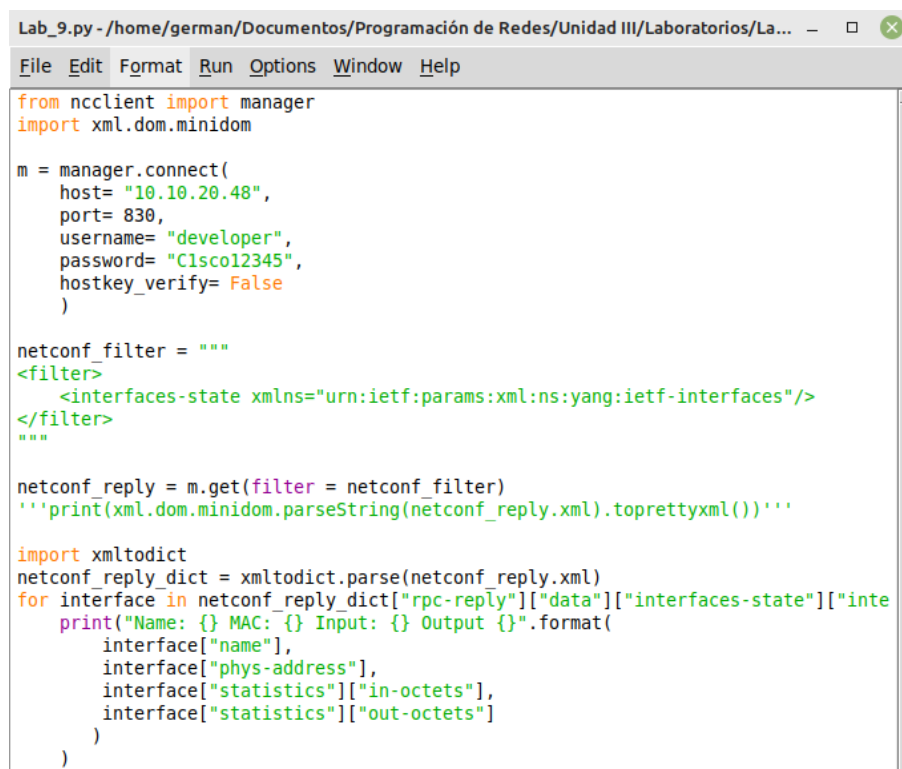
        interface["phys-address"],

        interface["statistics"]["in-octets"],

        interface["statistics"]["out-octets"]

    )

)
```



```
Lab_9.py - /home/german/Documentos/Programación de Redes/Unidad III/Laboratorios/La...
File Edit Format Run Options Window Help

from ncclient import manager
import xml.dom.minidom

m = manager.connect(
    host= "10.10.20.48",
    port= 830,
    username= "developer",
    password= "C1scol2345",
    hostkey_verify= False
)

netconf_filter = """
<filter>
  <interfaces-state xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"/>
</filter>
"""

netconf_reply = m.get(filter = netconf_filter)
'''print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())'''

import xmldict
netconf_reply_dict = xmldict.parse(netconf_reply.xml)
for interface in netconf_reply_dict["rpc-reply"]["data"]["interfaces-state"]["interface"]:
    print("Name: {} MAC: {} Input: {} Output {}".format(
        interface["name"],
        interface["phys-address"],
        interface["statistics"]["in-octets"],
        interface["statistics"]["out-octets"]
    )
)
```

En este apartado debemos agregar configuración para convertir los datos XML en un diccionario de Python, de manera que debemos importar un módulo llamado “xmldict”. Al parecer este módulo lo debemos instalar aparte, ya que puede que no venga por defecto en algunas distribuciones. También debemos utilizar un ciclo for para mostrar una vista resumida de los datos estadísticos.

- g. Execute the script and explore the output.

```
= RESTART: /home/german/Documentos/Programación de Redes/Unidad III/Laboratorios
/Lab-9/Lab_9.py
Name: GigabitEthernet1 MAC: 00:50:56:bf:75:bb Input: 50160 Output 436483
Name: GigabitEthernet2 MAC: 00:50:56:bf:e5:e1 Input: 0 Output 0
Name: GigabitEthernet3 MAC: 00:50:56:bf:87:d7 Input: 0 Output 0
Name: Control Plane MAC: 00:00:00:00:00:00 Input: 0 Output 0
```

Esta es la salida del diccionario que solicitamos, que nos muestra de una manera más ordenada y específica de algunas interfaces del Router.

### Conclusiones

De este último laboratorio me quedo más tranquilo y satisfecho de todos los conocimientos que adquirí al estar trabajando con cada uno de ellos, la mayoría tienen el mismo objetivo de interactuar, pero desde diferentes maneras y aplicaciones, ya sean por entorno gráfico como POSTMAN o un entorno de terminal como lo es IDLE, pero finalmente nos enlazamos en un mismo lenguaje, solicitando datos y convirtiéndolos en diferentes formatos.