2022

Lab – NETCONF w/Python: Get Operational Data

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

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.

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 and the xml.dom.minidom module:

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

c. Set up 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
        )
```

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

m=manager.connect(
    host="10.10.20.48",
    port=830,
    username="developer",
    password="C1sco12345",
    hostkey_verify=False
)
```

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)
```

```
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())
```

e. Execute the Python script and explore the output.

f. Convert the XML netconf_reply data to a Python dictionary using the "xmltodict" module. You can use a simple **for** loop to print a summary view of the statistical data:

```
import xmltodict

netconf_reply_dict = xmltodict.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"]
```

```
        )
    )
```

```
netconf_reply_dict = xmltodict.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"]
        )
    )
```

g. Execute the script and explore the output.

```
========== RESTART: C:\Users\pablo\Documents\UTNG Cuatrimestre 4\Programación de Redes\Unidad III\Ejercicios\2.9_lab.py =
Name: GigabitEthernet1 MAC: 00:50:56:bf:ff:31 Input: 167348 Output 1197392
Name: GigabitEthernet2 MAC: 00:50:56:bf:aa:82 Input: 0 Output 0
Name: GigabitEthernet3 MAC: 00:50:56:bf:f5:15 Input: 0 Output 0
Name: Loopback1 MAC: 00:1e:e6:b3:25:00 Input: 0 Output 608
Name: Loopback99 MAC: 00:1e:e6:b3:25:00 Input: 0 Output 232
Name: Loopback111 MAC: 00:1e:e6:b3:25:00 Input: 0 Output 0
Name: Control Plane MAC: 00:00:00:00:00:00 Input: 0 Output 0
```

Conclusión

Dentro del código que ejecutamos dentro del código tendremos que mostrar las direcciones MAC en conjunto con otras parámetros como el nombre, input , output y el tipo de formato. Se pueda usar igual de otras formas para obtener datos de operaciones de diferentes parámetros y información relevante en otras aspectos que necesites saber.

NETOCONF dentro de un router

Dentro de una sesión de NETCONF, una aplicación cliente puede solicitar información sobre el estado actual de un dispositivo que ejecuta dentro del router. Para solicitar información operativa, una aplicación cliente emite el elemento de etiqueta de solicitud específico de la API XML del router que devuelve la información deseada.

La aplicación cliente puede especificar el formato de la información devuelta por el servidor NETCONF. Al establecer el atributo de formato opcional en la etiqueta de solicitud operativa de apertura, una aplicación cliente puede especificar el formato de la respuesta como formato de etiqueta XML, que es el texto ASCII formateado predeterminado, o notación de objetos de JavaScript (JSON).

¿Para qué sirve xmltodict?

xml2dict nos permite tratar archivos XML de una manera muy fácil, ya que convierte los archivos XML en una estructura de datos de tipo diccionario. En nuestro codijo que ejecutamos obtenemos la dirección MAC que esta dentro de un diccionario.

¿Para qué sirve la librería `xml.dom.minidom`?

xml. dom. minidom es una implementación mínima de la interfaz Document Object Model (Modelo de objetos del documento), con una API similar a la de otros lenguajes. Está destinada a

ser más simple que una implementación completa del DOM y también significativamente más pequeña.