



Universidad Tecnológica  
del Norte de Guanajuato  
Organismo Público Descentralizado del Gobierno del Estado

"Educación y progreso para la vida"

Tecnologías de la Información y Comunicación

Programa educativo: TSU en Infraestructura de Redes Digitales

Área académica: Programación de Redes

Asignatura:

Unidad 3

Grupo: GIR0441

### **2.8 Lab - NETCONF wPython Device Configuration**

Alumna Gómez Luna Cinthia Valeria:

Docente: Gabriel Barrón Rodríguez

Dolores Hidalgo, C.I.N., Gto., Jueves 24 de Noviembre de 2022

En el presente documento se logrará ver algunas capturas realizadas las cuales nos ayudan a mostrar cómo se estará trabajando con el NETCONF se podrá lograr ver cómo se puede recuperar la configuración de los dispositivos y así también como poder crearla y veremos cómo realizar el soporte transaccional del NETconf.

Deberemos agregar nuestro Script el cual contiene lo siguiente

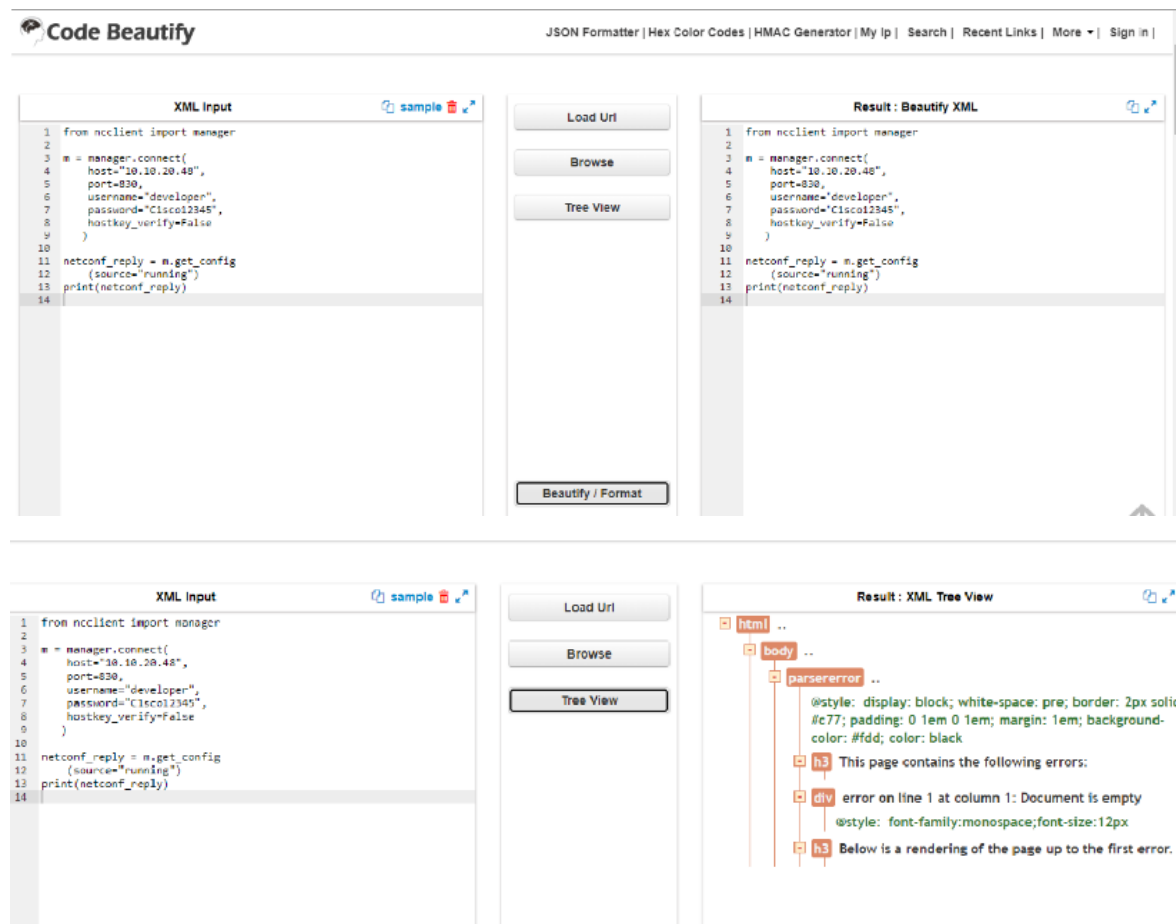
```
from ncclient import manager
```

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

File Edit Format Run Options Window Help

```
from ncclient import manager  
  
m = manager.connect(  
    host="10.10.20.48",  
    port=830,  
    username="developer",  
    password="C1sco12345",  
    hostkey_verify=False  
)  
  
netconf_reply = m.get_config  
    (source="running")  
print(netconf_reply)  
|
```

Una vez Realizado este paso se deberá implementar en la página CodeBeautify.com para poder realizar lo siguiente



The top screenshot shows the Code Beautify website interface. On the left, the 'XML Input' field contains a Python script for connecting to a Netconf manager. In the center, there are buttons for 'Load Uri', 'Browse', 'Tree View', and 'Beautify / Format'. On the right, the 'Result: Beautify XML' field shows the same script formatted.

The bottom screenshot shows the same interface, but the 'Result: XML Tree View' field displays an XML tree structure with error messages. The tree structure is as follows:

```
html ..
├── body ..
│   └── parsererror ..
│       @style: display: block; white-space: pre; border: 2px solid #c77; padding: 0 1em 0 1em; margin: 1em; background-color: #fdd; color: black
│       └── h3 This page contains the following errors:
│           └── div error on line 1 at column 1: Document is empty
│               @style: font-family: monospace; font-size: 12px
│               └── h3 Below is a rendering of the page up to the first error.
```

Una vez Realizado el paso anterior se deberá actualizar la configuración y se deberá crear un nuevo SCRIP el cual contendrá lo siguiente

```
from ncclient import manager
```

```
import xml.dom.minidom
```

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

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="C1sco12345",
    hostkey_verify=False
)
```

Se deberá crear una interfaz de bucle invertido el cual se deberá ver de la siguiente manera

```

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="Cisc012345",
    hostkey_verify=False
)

netconf_data = """
<config>
  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
    <interface>
      <Loopback>
        <name>111</name>
        <description>TEST1</description>
        <ip>
          <address>
            <primary>
              <address>100.100.100.100</address>
              <mask>255.255.255.0</mask>
            </primary>
          </address>
        </ip>
      </Loopback>
    </interface>
  </native>
</config>
"""

netconf_reply = m.edit_config(target="running", config=netconf_data)
print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())

<?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:64b489dd-4473-4676-aa09-ea2cfe2b890">
  <ok/>
</rpc-reply>

```

Una vez logrado este punto se deberá Ejecutar el comando s hip int brief

```

csr1000v-1#sh ip int brief

```

Interface	IP-Address	OK?	Method	Status	Protocol
GigabitEthernet1	10.10.20.48	YES	NVRAM	up	up
GigabitEthernet2	unassigned	YES	NVRAM	administratively down	down
GigabitEthernet3	unassigned	YES	NVRAM	administratively down	down
Loopback111	100.100.100.100	YES	other	up	up

```

csr1000v-1#

```

Después de realizar esto se deberá de agregar el comando sh int desc

```
csr1000v-1#sh int desc
Interface          Status      Protocol Description
Gi1                up          up      MANAGEMENT INTERFACE - I
N'T TOUCH ME
Gi2                admin down  down    Network Interface
Gi3                admin down  down    Network Interface
Lo111             up          up      TEST1
```

### Conclusión

Para poder lograr concluir esta práctica tuvimos que a ver ejecutado la 2.7 la cual conocimos y pusimos a prueba el ncclient y en esta prueba la volvimos a poner a prueba en esta práctica se realizó una práctica más completa, anterior mente logramos comprender que ncclient nos proporciona API's las cuales pueden lograr a ser mapeadas de una forma adecuada así puedes lograr facilitar los trabajos de los administradores se puede llegar a concluir que se llegó al resultado esperado de una manera buena