Lab – NETCONF w/Python: Get Operational Data

Objectives

Part 1: Retrieve the IOS XE VMs' existing running configuration

Background / Scenario

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

Required Resources

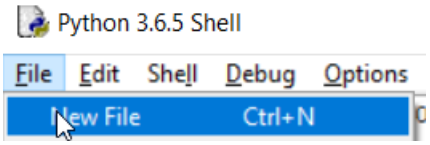
- Access to a router with the IOS XE operating system version 16.6 or higher
- Python 3.x environment

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 that in the following steps is being transformed into a tabular output.

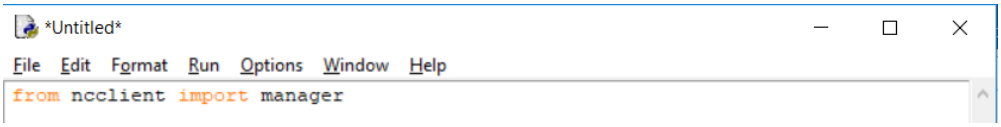
Step 1: Use ncclient to retrieve the device's running configuration.

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:

```
from ncclient import manager
```



c. 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
        )
```

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

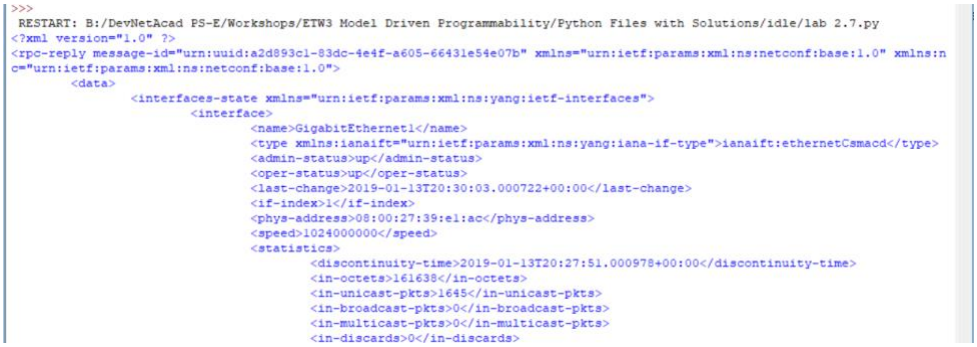
The following filter retrieves the interfaces-state operational data (statistics), as defined in the ietf-interfaces YANG model.

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())
```

e. Execute the Python script and explore the output:



```
>>>
 RESTART: B:/DevNetAcad PS-E/Workshops/ETW3 Model Driven Programmability/Python Files with Solutions/idle/lab 2.7.py
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:a2d893c1-83dc-4e4f-a605-66431e54e07b" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:n
c="urn:ietf:params:xml:ns:netconf:base:1.0">
        <data>
                <interfaces-state xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
                        <interface>
                                <name>GigabitEthernet1</name>
                                <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
                                <admin-status>up</admin-status>
                                <oper-status>up</oper-status>
                                <last-change>2019-01-13T20:30:03.000722+00:00</last-change>
                                <if-index>1</if-index>
                                <phys-address>08:00:27:39:e1:ac</phys-address>
                                <speed>1024000000</speed>
                                <statistics>
                                        <discontinuity-time>2019-01-13T20:27:51.000978+00:00</discontinuity-time>
                                        <in-octets>161638</in-octets>
                                        <in-unicast-pkts>1645</in-unicast-pkts>
                                        <in-broadcast-pkts>0</in-broadcast-pkts>
                                        <in-multicast-pkts>0</in-multicast-pkts>
                                        <in-discards>0</in-discards>
                                                ...
```

f. Converting 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"]
        )
)
```

g. Execute the script and explore the output: Take a screenshot of the result.

```
Name: GigabitEthernet1 MAC: 08:00:27:80:a5:ed Input: 11770371 Output 2091466
Name: Loopback1 MAC: 00:1e:49:7b:e4:00 Input: 0 Output 0
Name: Loopback2 MAC: 00:1e:49:7b:e4:00 Input: 0 Output 0
Name: Loopback100 MAC: 00:1e:49:7b:e4:00 Input: 0 Output 0
Name: Control Plane MAC: 00:00:00:00:00:00 Input: 0 Output 0

[Done] exited with code=0 in 0.633 seconds
```