NETCONF und Python

1.) Bauen Sie über myExperTeach die Verbindung zur VM auf.
2.) Starten Sie eine Shell und von dort aus mit *telnet 10.196.0.201* eine Telnet-Sitzung zu dem Router auf mit Logins: lab/lab123. Diese Sitzung bitte bestehen lassen.
3.) Konfigurieren Sie auf dem Juniper Router NETCONF wie folgt:

```
lab@vSRX> config
[edit]
lab@vSRX# set system services netconf ssh

[edit]
lab@vSRX# commit
```

4.) Starten Sie eine weitere Shell und wechseln Sie in das Verzeichnis /NETY. Sie finden dort zwei Python-Programme vor:

```
et@ubuntu:~$ cd NETY
et@ubuntu:~/NETY$ ls
junos_editconfig.py  junos_getconfig.py
```

Beide Programme bieten ein programmatisches Framework zum lesenden und schreibenden Zugriff auf den Juniper Router. Dabei wird die ncclient Library verwendet. Das Framework hat an strategisch wichtigen Stellen Platzhalter, die im Zuge der Übung mit Inhalt gefüllt werden. In der Übung soll zunächst aus dem YANG-Modell des JUNOS der XML-Ausdruck für die Konfiguration einer Loopback-Adresse erzeugt werden. Dieser wird dann zum Konfigurieren und Auslesen der Loopback-Adresse von Python aus verwendet.

5.) Öffnen Sie eine dritte Shell und wechseln Sie in das Verzeichnis API/NETCONF/Other_YANG/yang/vendor/juniper/16.1:

```
et@ubuntu:~$ cd API/NETCONF/Other_YANG/yang/vendor/juniper/16.1
et@ubuntu:~/API/NETCONF/Other_YANG/yang/vendor/juniper/16.1$ ls
configuration.yang  junos-extension.yang  operational
```

In der Datei configuration.yang befindet sich das komplette YANG-Modell von JUNOS 16.1. Der Teil mit der Konfiguration einer IP-Adresse auf einem Interface lässt sich mit pyang folgendermaßen analysieren:

```
et@ubuntu:~/API/NETCONF/Other_YANG/yang/vendor/juniper/16.1$ pyang -f tree
configuration.yang --tree-
path=/configuration/interfaces/interface/unit/family/inet/address/name
```

Analysieren Sie die YANG-Statements und überlegen Sie wie der dazu passende XML Code strukturiert sein muss.

6.) Sie können mit pyang auch direkt eine YANG-kompatibles XML-Skelett erzeugen:

```
et@ubuntu:~/API/NETCONF/Other_YANG/yang/vendor/juniper/16.1$ pyang -f sample-xml-
skeleton configuration.yang --sample-xml-skeleton-
```

path=/configuration/interfaces/interface/unit/family/inet/address/name

```
<?xml version='1.0' encoding='UTF-8'?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <configuration xmlns="http://yang.juniper.net/yang/1.1/jc">
  <interfaces>
   <interface>
     <unit>
      <family>
       <inet>
        <address>
         <name/>
        </address>
       </inet>
      </family>
     </unit>
    </interface>
  </interfaces>
 </configuration>
</data>
```

Öffnen Sie mit Sublime das Programm junos_editconfig.py und fügen Sie das XML-Skelett (ohne den XML Prologue und das Element <data>) anstelle von <MY XML CONFIG> in die lokale Variabel cfg1 ein. Ergänzen sie unter <interface> und <unit> ein zusätzliches Empty Element <name/>. Es handelt sich um die Keys der entsprechenden Listen im YANG-Modell. Tragen Sie in die Empty Elements die richtigen Werte für die unit 0 auf dem Interface lo0 ein:

```
<configuration xmlns="http://yang.juniper.net/yang/1.1/jc">
  <interfaces>
   <interface>
   <name>lo0</name>
     <unit>
     <name>0</name>
      <family>
       <inet>
        <address>
         <name/>
        </address>
       </inet>
      </family>
     </unit>
    </interface>
  </interfaces>
 </configuration>
```

Das Empty Element <name/> unter <address> fungiert als Träger der Loopback-IP-Adresse in Prefix-Notation. Hier soll für den vSRX_x die Addresse 1.1.1.x/32 (x = 01, 02, …, 12) eingetragen werden:

<name/> -> <name>1.1.1.x/32</name>

7.) Fügen Sie unter manager.connect() hinter host, username und password die richtigen Parameter ein, die hier zunächst nur durch Variablen vertreten werden:

    <MY IP ADDRESS> -> 10.196.0.x
    <MY USERNAME> -> lab
    <MY PASSWORD> -> lab123

8.) Speichern Sie mit Str-S ab und rufen Sie das Programm auf der Shell mit Python 3 auf:

```
et@ubuntu:~ /NETY $ python3 junos_editconfig.py
<rpc-reply message-id="urn:uuid:a79c5351-45b6-4614-bc1f-5320262c6f67">
 <ok/>
</rpc-reply>
```

    Verifizieren Sie über die CLI Session, dass der Router die Konfiguration in die Candidate Configuration übernommen hat.

9.) Was geschieht, wenn Sie eine zusätzliche Loopback-Adresse 2.2.2.x/32 per NETCONF pushen?? Was müsste in dem Python-Programm geändert werden, um eine Ersetzung (replace) der Adresse zu erreichen? ACHTUNG: Ändern Sie bitte nicht die default_operation='merge'! Fügen Sie stattdessen das Attribut operation="replace" an die richtige Stelle im XML Code ein.

10.) Mit operation="delete" kann die IP-Adresse auf dem Loopback-interface gezielt gelöscht werden. Verifizieren Sie dies, indem Sie zunächst mehrere parallele IP-Adressen konfigurieren, um dann eine davon gezielt zu löschen!

11.) Bislang ist die Candidate Configuration uncommited. Das merkt man daran, dass beim Versuch, den Configuration Mode mit exit zu verlassen, eine diesbezügliche Warnmeldung ausgegeben wird. Das Commitment ist per NETCONF mit der Operation commit möglich. Kommentieren Sie die Variable resp unter try: mit # aus und definieren Sie eine neue:

```
resp = junos_nc.commit()
```

    Achten Sie hierbei auf die korrekte Indentation (Einrückung, Standard in Python 4 Zeichen).

    Rufen Sie das Programm junos_editconfig.py damit nach Abspeichern erneut auf. Verifizieren Sie über die CLI Session, dass die Candidate Configuration commited wurde. Der Router sollte Sie nun mit exit kommentarlos aus dem Configuration Mode entlassen.

12.) Spielen Sie (mit der ursprünglichen Definition der Variablen resp) verschiedene Fehlerbilder durch, und beobachten Sie die Reaktion von Python:
- falsche Router IP-Adresse
- falsche Credentials
- falsch formatierter XML Code
- falscher Namespace
- falscher Element Name
- falscher Content (Interface-Kennung, Unit-Nummer, IP-Adresse)
- target='running'. Warum reagiert der Router so wiederwillig??

13.) Editieren Sie durch eine analoge Vorgehensweise das Programm junos_getconfig.py, um damit gezielt die Loopback-Adresse(n) auszulesen. Sie können die in junos_editconfig.py verwendete XML-Struktur hierbei als Subtree Filter in der lokalen Variablen f1 weiterverwenden (ohne das

Element <config>!!), indem Sie für die IP-Adresse ein Empty Element <name/> ersetzten.
Ändern Sie unter get_config() das Statement 'running' in 'candidate'. Bemerken Sie irgendeinen Unterschied? Wenn nein, warum nicht?