1.) Verbinden Sie sich mit dem Jump Host auf dem Guacamole Server mit den vom Trainer zur Verfügung gestellten Zugangsdaten. Dort sollte bereits ein Fenster mit einer RDP Sitzung zu einer Ubuntu VM in der Testnetz-Umgebung geöffnet sein, das Sie maximieren oder auf Vollbild einstellen sollten. Im Verzeichnis /NSO1 unterhalb des Home-Verzeichnisses finden Sie diverse für die Übungen nützliche Dateien.
2.) Ziel der Übung ist die die Einrichtung eines Service, der einer Gruppe von JUNOS Devices eine Loopback-Adresse und RID zuteilt sowie OSPF in einer Area mit vorgegebener (für alle Devices übereinstimmender) Area ID auf allen Interfaces aktiviert. Die Service-Parameter sind demnach:
    - Device-Name (String)
    - Loopback-Adresse = RID (Dotted Decimal)
    - Area ID (Decimal)

    Begleitend soll eine Action eingerichtet werde, die in einer Service-Instanz die Konnektivität der Loopback-Adressen aller Devices testet.
3.) Onboarden Sie die unter ihrer Kontrolle stehenden Juniper Devices, falls dies noch nicht geschehen sein sollte.

4.) Erzeugen Sie ein Package vom Typ Python only mit Action Example. Der Name des Package soll LOOP lauten.
5.) Richten Sie das YANG-Modell für den Service ein. Der in das YANG Skeleton einzufügende Teil könnte wie folgt gestaltet sein :

```
list LOOP{

   key name;
   leaf name {
    type string;
   }

   uses ncs:service-data;
   ncs:servicepoint LOOPservicepoint;

   leaf area {type uint8;}

   list device {
     key name;
     leaf name {
        type leafref {   path "/ncs:devices/ncs:device/ncs:name";}

     leaf loopback {type inet:ipv4-address;}
    }
}
```

6.) Ergänzen Sie das YANG-Modell um Statements für den Test:

```
container Test {

  tailf:action LOOPTest {
   tailf:actionpoint LOOPTest-actionpoint;
    input {
     leaf ServiceInstance {
      type leafref {
       path “/LOOP/name";
      }
     }}

    output {
     leaf Result {
      type string;
     }
    }
  }
 }
```

Compilieren Sie das fertige YANG-Modul mit

et@ubuntu:~/nso-5.1/Lab/packages/LOOP/src$ make clean all

7.) Editieren Sie das Python Skeleton main.py unter /home/et/nso-5.1/Lab/packages/LOOP/python/LOOP wie folgt:

Fügen Sie import Anweisungen für MAAPI und MAAGIC hinzu:

```
from ncs import maapi
from ncs import maagic
```

Im Bereich für den Service Callback:

- MAAGIC-Ausdrücke zum Auslesen der Service-Parameter area, device und loopback
- MAAGIC-Ausdrücke zum Konfigurieren der betroffenen Device-Instanzen durch Iteration über die Devices in der Service-Instanz. TIP: Erzeugen Sie eine Liste dieser Devices mit service.device.keys()

TIP: Erzeugen Sie eine fiktive JUNOS-Konfiguration für die RID, Loopback-Adresse und OSPF-Konfiguration. In etwa folgendermaßen:

```
admin@ncs% set devices device j1 config junos:configuration routing-options router-id 1.2.3.4
admin@ncs% set devices device j1 config junos:configuration interfaces interface lo0 unit 0 family inet address …
                                                                                                … 1.2.3.4/32
admin@ncs% set devices device j1 config junos:configuration protocols ospf area 5 interface all
```

Rufen Sie die Konfiguration mit der Option display keypath oder diplay xpath ab, um daraus die MAAGIC Nodes zu gewinnen. TIP: area, interface (unter area) und address werden im NED formal als list behandelt. Löschen sie im Vorfeld die vorhandene OSPF-Konfiguration und Loopback-Adresse. TIP: Nutzen Sie die Built-in Funktion del. Der

resultierende Code könnte am Ende wie folgt gestaltet sein:

```
areaid = service.area
devlist = service.device.keys()

for dev in devlist:

  lb = service.device[dev].loopback
  self.log.info('Loopback: ' , lb)
  del root.devices.device[dev].config.junos__configuration.protocols.ospf.area
  root.devices.device[dev].config.junos__configuration.protocols.ospf.area.create(areaid)
 root.devices.device[dev].config.junos__configuration.protocols.ospf.area[areaid].interface.create('all')
 del root.devices.device[dev].config.junos__configuration.interfaces.interface['lo0'].unit[0].family.inet.address
 root.devices.device[dev].config.junos__configuration.interfaces.interface['lo0'].unit[0].family.inet. ….
                                                                                  …    address.create(lb + '/32')
  root.devices.device[dev].config.junos__configuration.routing_options.router_id = lb
```

Im Bereich für den Action Callback:

- Ändern Sie nach eigenem Ermessen den vorgegebenen Namen der Action Class ab.
- Passen Sie entsprechend die Parameter unter self.register_action() an. Insbesondere muss hier die Registrierung der Action Class bei dem Action Point LOOPTest-actionpoint erfolgen.
- MAAGIC-Ausdrücke zum Auslesen des Instanz-Namens aus dem Leaf ServiceInstance unter input und Ablage unter einer lokalen Variablen. Z.B. so:

```
servinst = input.ServiceInstance
```

- Auslesen der Service-Parameter device und loopback und Zuweisung der Werte an lokale Variablen aus der Service-Instanz. Hierzu muss zunächst eine MAAPI Transaction eingerichtet werden, um den Root Node der CDB greifbar zu machen. TIP: Nutzen Die die Methode keys(), um die Device-Name in einer Liste abzulegen. Das könnte in etwa so aussehen:

```
with maapi.single_write_trans('admin', 'python') as t:
    root = maagic.get_root(t)
    servinst = input.ServiceInstance
    devlist = root.LOOP[servinst].device.keys()
```

- Für das spätere Anweisend der Pings, ist es sinnvoll, alle Loopback-adressen in einer Liste abzulegen:

```
alllb = []
for dev in devlist:
    lb = root.LOOP[servinst].device[dev].loopback
    alllb.append(lb)
```

- MAAGIC-Ausdrücke zum Anweisen der Action in geschachtelten for Loops über alle Devices und alle Loopback-Adressen in der Service-Instanz. Für den JUNOS NED lauten

diese (eingebettet in die for Loops) wie folgt:

```
result = '\n'
  for dev in devlist:
    lb = root.LOOP[servinst].device[dev].loopback
    for lo in alllb:
      if lo != lb:
        action = root.devices.device[dev].rpc.jrpc__rpc_ping.ping
        inp = action.get_input()
        inp.host = lo
        inp.count = 2
        outp = action(inp)
        resp = outp.ping_results.probe_results_summary.responses_received
        if resp == 2:
          result += 'Ping von %s with %s -> %s successful\n' % (dev  ,lb ,lo)
        else:
          result += 'Ping von %s with %s -> %s NOT successful\n' % (dev , lb, lo)
```

Die Variable resp enthält die Anzahl der beantworteten Pings. Sie sollte identisch inp.count = 2 (Sie können hier auch eine andere Zahl wählen; bei größeren Werten dauert der Test entsprechend länger) sein. Dies wird hier als Kriterium für die Erreichbarkeit. Verifizieren Sie die Pfad-Angaben in den MAAGIC-Ausdrücken (in rot) durch Analyse des korrespondierenden Teils im YANG Modell junos-rpc.yang im JUNOS NED:

```
et@ubuntu:~/nso-5.1/Lab/packages/juniper-junos-nc-3.0/src/yang$ ls
junos-rpc.yang  junos.yang

et@ubuntu:~/nso-5.1/Lab/packages/juniper-junos-nc-3.0/src/yang$ subl junos-rpc.yang
```

Randbemerkung: Dieses YANG-Modell ist im gesamten YANG-Modell der CDB an den Node root/devcies/device/rpc „gemounted“ worden, sodass die Pfade zu den Schema Nodes mit root/devices/device/rpc beginnen müssen

- Der aus den Resultaten des Pings erzeugte String result wird an den Leaf Result unter output übergeben:

```
output.Result = result
```

- Am Ende aller zu der Transaction t gehörenden Anweisungen muss noch ein t.apply() angebracht werden.

8.) Aktivieren Die das Package LOOP durch einen Reload aller Packages im CLI des NSO:

```
admin@ncs> request packages reload
```

9.) Erzeugen Die eine Service-Instanz von LOOP

```
LOOP A {
      area 3;
      device j1 {
         loopback 1.1.1.1;
      }
      device j2 {
         loopback 2.2.2.2;
       }

      device j3 {

        loopback 3.3.3.3;
      }
}
```

(der der Name der Service-Instanz, die Namen der Device-Instanzen und IP-Adressen sind hier nur exemplarisch zu verstehen) und verifizieren Sie über eine CLI-Session mit den betroffenen Devices deren Wirksamkeit. Testen Die danach über einen Aufruf des Tests mit

request Test LOOPTest ServiceInstance <Instance Name>

die Funktionstüchtigkeit der Service-Instanz und der Test Action. Sollte der Name der Action „Test“ auch in den YANG-Modellen anderer Service Packages Verwendung finden, ist der der Präfix des Namespace LOOP als Unterscheidungskriterium anzubringen (LOOP:Test). Dies wird einem dann aber auch mit der kontextsensitiven Hilfe (?) so angeboten.