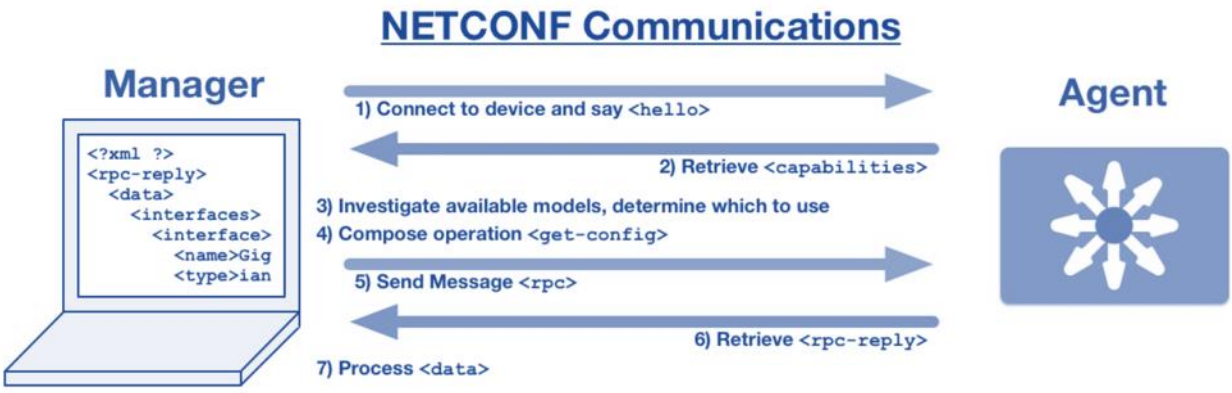
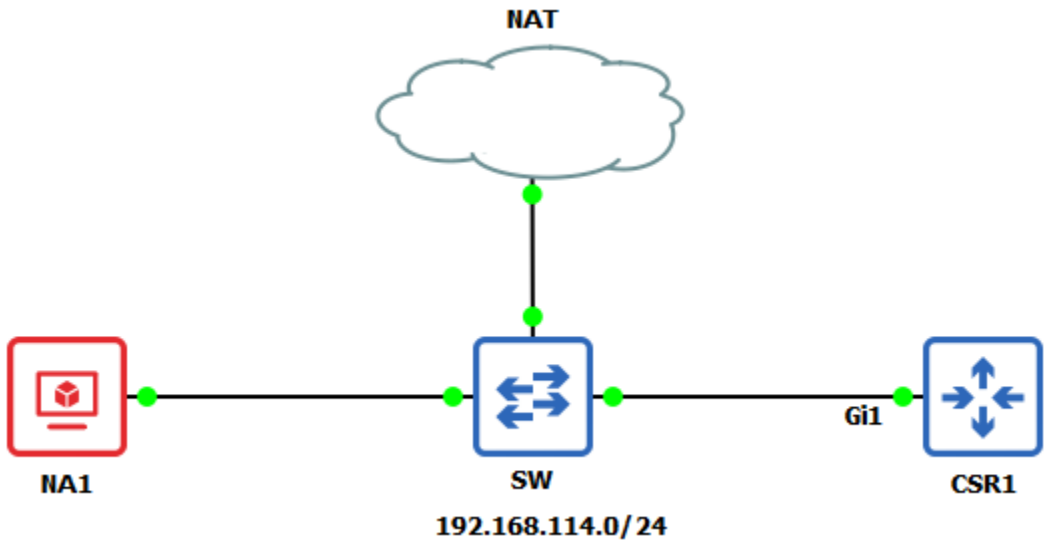
NETCONF:

- Network Configuration Protocol (NETCONF) is IETF standard developed to manage.
- Network Configuration Protocol (NETCONF) developed to manage network devices.
- The goal of Network Configuration Protocol is to make network automation easier.
- NETCONF uses XML for data encoding & Remote Procedure Call (RPC) for messages.
- Simple Network Management Protocol was also developed by IETF for same purpose.
- NETCONF & SNMP have many similarities & people referring to NETCONF as SNMPv4.
- It provides mechanisms to install, manipulate & delete configuration of network devices.
- YANG describes the data structures exchanged in NETCONF message like SNMP MIBs.
- NETCONF is based on XML messages exchanged via SSH protocol using TCP port 830.
- NETCONF is one of key APIs used by SDN Controller on Southbound to control devices.
- Uses XML-based data encoding for configuration data & protocol messages with YANG.
- It uses simple RPC-based mechanism to facilitate communication between client/server.
- RESTCONF provides subset of NETCONF functionality implemented on top of HTTP/HTTPS.
- Both NETCONF and RESTCONF were developed to manages devices in a standard way.
- Network devices running a NETCONF agent can be managed through five operations:

Operation	Description
<get>	Retrieve running configuration & device state information.
<get-config>	Retrieve all or part of a specified configuration datastore.
<edit-config>	The <edit-config> operation loads all or part of a specified configuration to the specified target configuration datastore.
<copy-config>	Create or replace an entire configuration datastore with the contents of another complete configuration datastore.
<delete-config>	Delete a configuration datastore. The <running> configuration datastore cannot be deleted.



NETCONF Lab Time:



Network Automation IP Configuration
DHCP config for eth0 auto eth0 iface eth0 inet dhcp

CSR Router Configuration
Router(config)#hostname CSR CSR(config)#interface gigabitEthernet 1 CSR(config)#ip address dhcp CSR(config-if)#no shutdown CSR(config)#username admin privilege 15 password 123 CSR# clock set 10:10:10 1 Jan 2019 CSR(config)#netconf-yang
CSR# show platform software yang-management process CSR# show netconf session CSR# clear netconf sessions

Network Automation Machine
root@NA1:~# ssh -s admin@192.168.114.224 -p 830 netconf

The router responds with a “hello” message:

```
root@NA1:~# ssh -s admin@192.168.114.224 -p 830 netconf
admin@192.168.114.224's password:
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
<capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
<capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
<capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
<capability>urn:ietf:params:netconf:capability:notification:1.0</capability>
<capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>
<capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=expli
<capability>urn:ietf:params:netconf:capability:yang-library:1.0?revision=2016-06-
<capability>http://tail-f.com/ns/netconf/actions/1.0</capability>
<capability>http://tail-f.com/ns/netconf/extensions</capability>
<capability>
        urn:ietf:params:netconf:capability:notification:1.1
      </capability>
</capabilities>
<session-id>88</session-id></hello>]]>]]>
```

This hello message contains the capabilities that the router supports. We need to reply to the router with a hello message. This hello message contains the capabilities that we support:

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>]]>]]>
```

```
<capability>
        urn:ietf:params:netconf:capability:notification:1.1
      </capability>
</capabilities>
<session-id>702</session-id></hello>]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <capabilities>
        <capability>urn:ietf:params:netconf:base:1.0</capability>
    </capabilities>
</hello>]]>]]>
```

The router won't show anything to acknowledge our hello message, but we now have an active NETCONF session. To end the session, send a message to close operation. Paste the text below on Terminal.

```
 <?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="559" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <close-session />
 </rpc>
 ]]>]]>
```

RPC:

NETCONF uses Remote Procedure Calls (RPCs), to send a request to a server and perform an action. XML-formatted message are sent to the server. Server returns results within an <rpc-reply>. The <rpc> message sends a message-id attribute with a unique value to identify the request it is responding to:

```
<rpc-reply message-id="urn:uuid:4ca456ec-ce2a-559c-cb68-1c70dd2227a5">
.
.
</rpc-reply>
```

Interacting with NETCONF manually over SSH work but hard job. Instead, we should use clients or code libraries that do most of the work. Ncclient is a popular python library we can use for NETCONF. Ncclient is a Python library that facilitates client-side scripting and application development around the NETCONF protocol.

Retrieve Capabilities of Router:

```
from ncclient import manager

# IOS XE Settings
ios_xe_host = "192.168.114.224"
ios_xe_port = 830
ios_xe_username = "admin"
ios_xe_password = "123"

m = manager.connect(
   host=ios_xe_host,
   port=ios_xe_port,
   username=ios_xe_username,
   password=ios_xe_password,
   hostkey_verify=False,
   look_for_keys=False
)

for capability in m.server_capabilities:
   print(capability)

m.close_session()
```

root@NA1:~# python cap.py

The router replies with a list of all its capabilities:

```
root@NA1:~# python cap.py
http://cisco.com/ns/yang/Cisco-IOS-XE-utd?module=Cisco-IOS-XE-utd&revision=2017-08-30
http://cisco.com/ns/yang/Cisco-IOS-XE-features?module=Cisco-IOS-XE-features&revision=20
,punt-num,parameter-map,multilink,l2vpn,l2,ezpm,eth-evc,esmc,efp,crypto
http://openconfig.net/yang/vlan?module=openconfig-vlan&revision=2016-05-26&deviations=c
http://openconfig.net/yang/network-instance-l3?module=openconfig-network-instance-l3&re
urn:ietf:params:netconf:capability:rollback-on-error:1.0
http://cisco.com/ns/yang/Cisco-IOS-XE-icmp?module=Cisco-IOS-XE-icmp&revision=2017-08-16
http://cisco.com/ns/yang/Cisco-IOS-XE-nbar?module=Cisco-IOS-XE-nbar&revision=2017-08-16
urn:ietf:params:xml:ns:yang:smiv2:CISCO-VOICE-DNIS-MIB?module=CISCO-VOICE-DNIS-MIB&revi
```

Retrieve Running Configuration of Router:

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

# IOS XE Settings
ios_xe_host = "192.168.114.224"
ios_xe_port = 830
ios_xe_username = "admin"
ios_xe_password = "123"

m = manager.connect(
   host=ios_xe_host,
   port=ios_xe_port,
   username=ios_xe_username,
   password=ios_xe_password,
   hostkey_verify=False,
   look_for_keys=False
)

netconf_reply = m.get_config("running")

print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())

m.close_session()
```

```
root@NA1:~# vi run.py
root@NA1:~# python run.py
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:2942d0d0-b888-451b-875b-0e732376fb04" xmlns="urn:ietf:
ns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
        <data>
                <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
                        <interface>
                                <name>GigabitEthernet1</name>
                                <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-
pe>
                                <enabled>true</enabled>
                                <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
                                <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
                        </interface>
                        <interface>
                                <name>GigabitEthernet2</name>
                                <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-
```

Create Loopback Interface on Router:

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

# IOS XE Settings
ios_xe_host = "192.168.114.224"
ios_xe_port = 830
ios_xe_username = "admin"
ios_xe_password = "123"

m = manager.connect(
   host=ios_xe_host,
   port=ios_xe_port,
   username=ios_xe_username,
   password=ios_xe_password,
   hostkey_verify=False,
   look_for_keys=False
)

netconf_interface_template = """
<config>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
            <interface>
                  <name>Loopback1</name>
                  <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-
type">ianaift:softwareLoopback</type>
                  <enabled>true</enabled>
                  <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
                        <address>
                              <ip>1.1.1.1</ip>
                              <netmask>255.255.255.255</netmask>
                        </address>
                  </ipv4>
            </interface>
      </interfaces>
</config>
"""
netconf_reply = m.edit_config(netconf_interface_template, target = "running")

print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
m.close_session()
```

```
root@NA1:~# vi loopback.py
root@NA1:~# python loopback.py
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:b5134e36-9143-4f49-98cf-608982a63219"
ns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
        <ok/>
</rpc-reply>
```

Lets verify in Router the loopback interface has been created.

```
CSR#show ip int br
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       192.168.114.224 YES DHCP   up                    up
GigabitEthernet2       unassigned      YES NVRAM  administratively down down
GigabitEthernet3       unassigned      YES NVRAM  administratively down down
GigabitEthernet4       unassigned      YES NVRAM  administratively down down
Loopback1              1.1.1.1         YES other  up                    up
CSR#
```

Delete Loopback Interface on Router:

```
from ncclient import manager
import xml.dom.minidom
# IOS XE Settings
ios_xe_host = "192.168.114.224"
ios_xe_port = 830
ios_xe_username = "admin"
ios_xe_password = "123"
m = manager.connect(
   host=ios_xe_host,
   port=ios_xe_port,
   username=ios_xe_username,
   password=ios_xe_password,
   hostkey_verify=False,
   look_for_keys=False
)
netconf_interface_template = """
<config>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
            <interface operation="delete">
                  <name>Loopback1</name>
            </interface>
      </interfaces>
</config>
"""
netconf_reply = m.edit_config(netconf_interface_template, target = "running")
print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
m.close_session()
```

```
root@NA1:~#
root@NA1:~# vi delloopback.py
root@NA1:~# python delloopback.py
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:93fcceb9-457d-4442-9607-1b653812e3fd"
ns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
        <ok/>
</rpc-reply>
```

Let's verify in Router the loopback interface has been deleted it is not there anymore.

```
CSR#
CSR#show ip int br
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       192.168.114.224 YES DHCP   up                    up
GigabitEthernet2       unassigned      YES NVRAM  administratively down down
GigabitEthernet3       unassigned      YES NVRAM  administratively down down
GigabitEthernet4       unassigned      YES NVRAM  administratively down down
CSR#
```