Lab 14 Use the Cisco IOS XE NETCONF API

Do Lab 14 Tasks 1 and 2 in full before starting the following task 3.

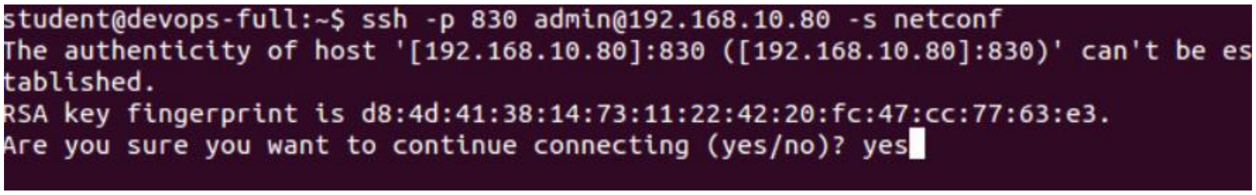
Task 3: NETCONF on IOS XE initial configuration with complete XML configuration data.

Ensure that Labs **9, 10, 13, 14** and Lab **14** tasks 1 and 2 are completed in full *before* starting this task.

In this task you will use **NETCONF** on your **CSR1000v** to extract the entire XML configuration and use that to perform configuration changes with Python.

How-to Steps

1. Connect to your **Ubuntu Configured** VM. Be *SURE* you do these steps from **Ubuntu Configured** and *not* **Ubuntu Unconfigured** as you will be relying on some Sublime XML pre-configurations.
2. Open a Terminal widow.
3. Type in the command to connect to your **CSR1000v** router with NETCONF.
```
ssh -p 830 admin@192.168.10.80 -s netconf
```
4. Respond with a **yes** if prompted about an RSA key. (you will likely not be prompted for later ssh sessions)

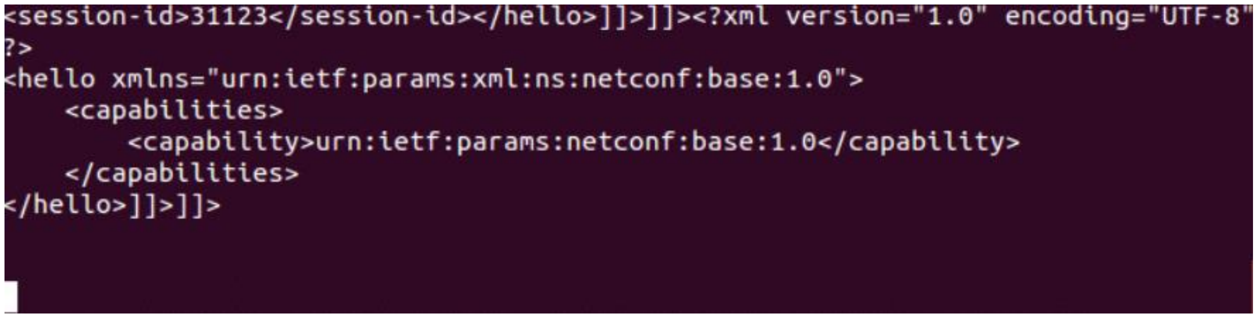


```
student@devops-full:~$ ssh -p 830 admin@192.168.10.80 -s netconf
The authenticity of host '[192.168.10.80]:830 ([192.168.10.80]:830)' can't be es
tablished.
RSA key fingerprint is d8:4d:41:38:14:73:11:22:42:20:fc:47:cc:77:63:e3.
Are you sure you want to continue connecting (yes/no)? yes
```

5. Enter the password **cisco** when prompted. Simply note all the XML output of this device capabilities. Maximize your terminal screen.
6. Enter in this exact XML code directly after the NETCONF prompt. This is stating your NETCONF client capabilities. You can type this in exactly or copy the identical content from the file in your **Class_Share** link on your Ubuntu desktop: **XML\NetconfClient.xml**.
```
<?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>]]>]]>
```

Note: Additional capabilities can be entered above obtained from the NETCONF device during the initial capabilities exchange.

7. You can hit enter a few times. Note how you *don't* have a left indented prompt at this point.



```
<session-id>31123</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>]]>]]>
```

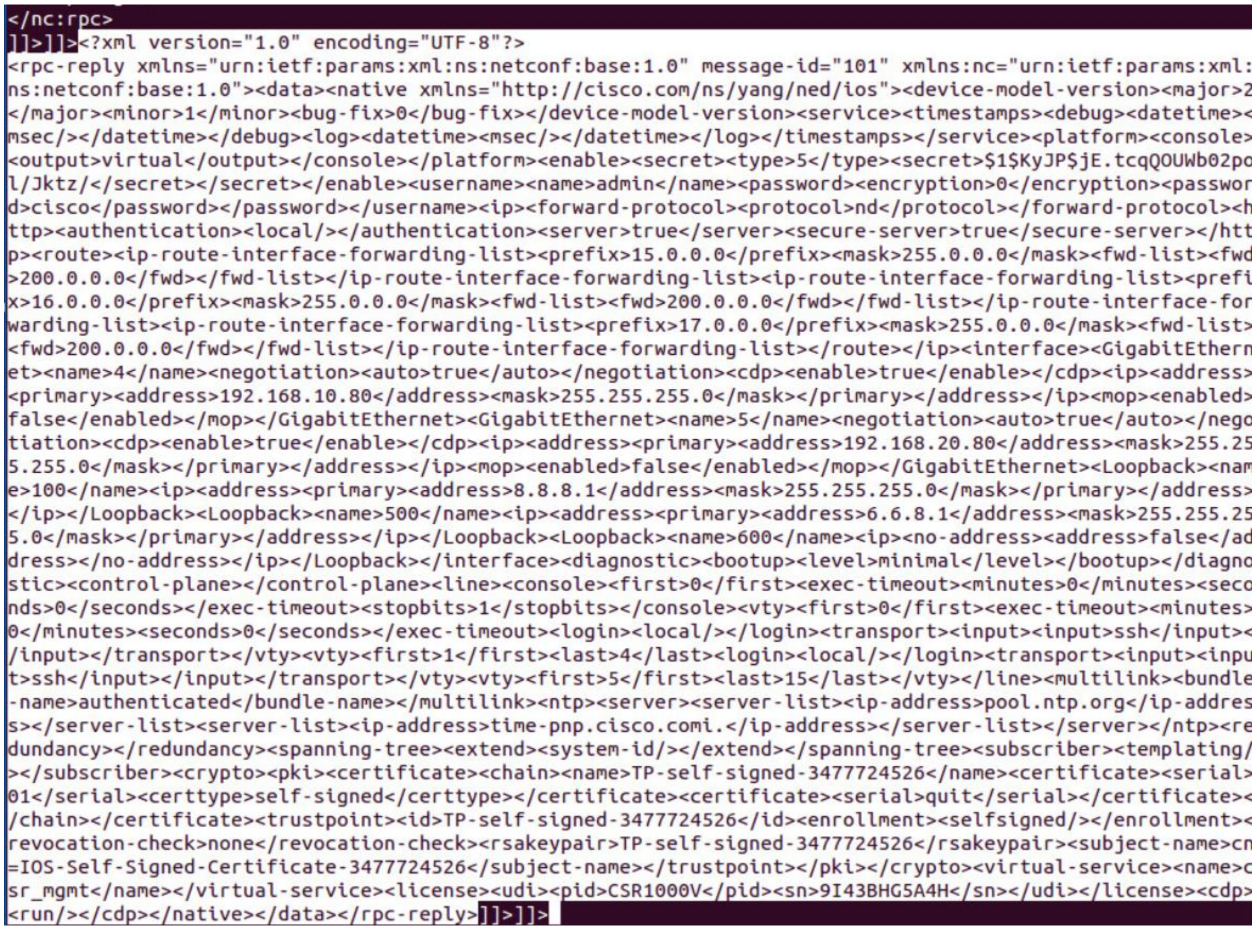
8. Maximize your terminal window to minimize the character returns within longer XML lines.

Note: NETCONF messages always end with the unique prompt of **]]>]]>**. This prompt occurs after each NETCONF communication, and is not itself valid XML.

9. To get the entire device config in XML, enter in this entire NETCONF **get** command after the prompt. You can type this in exactly, or copy the identical content from the file: **Z:\XML\NetconfConfig.xml**

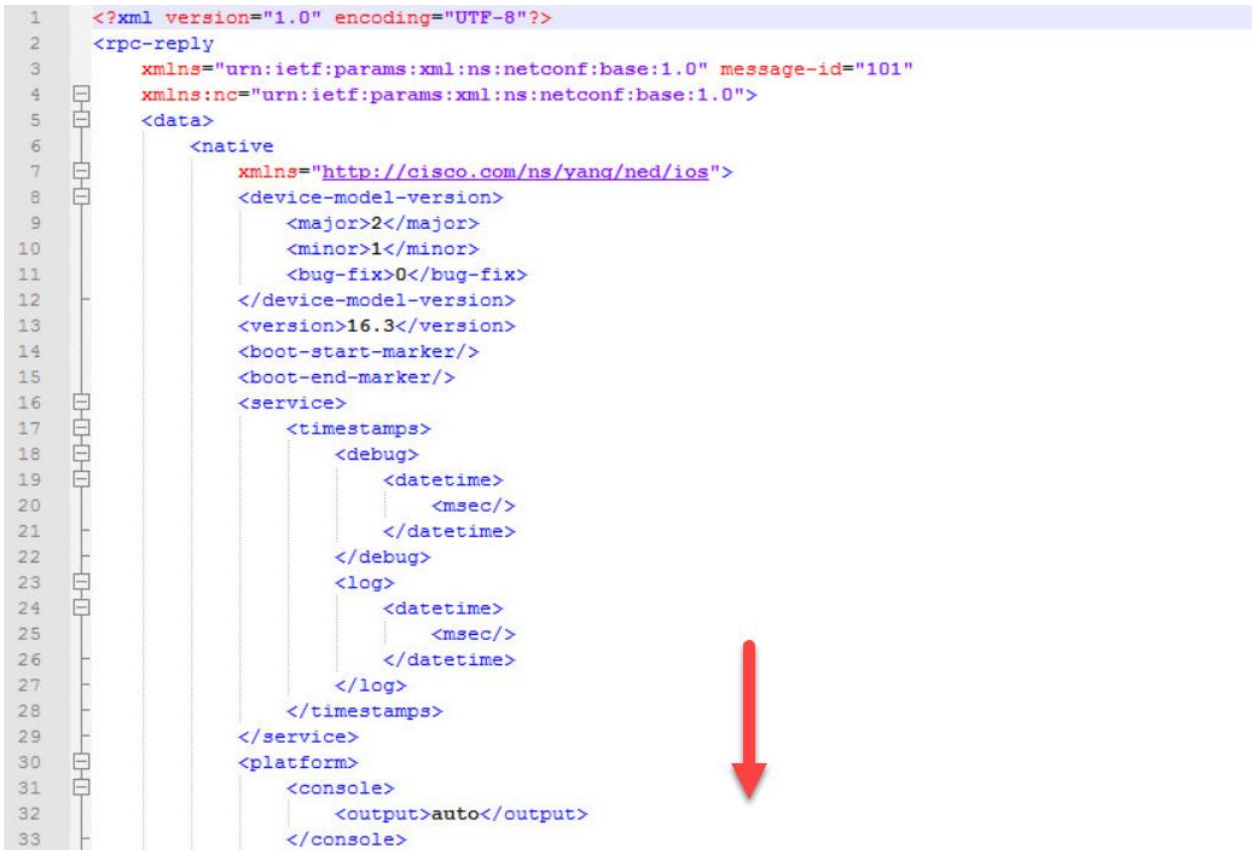
```
<?xml version="1.0"?>
<nc:rpc message-id="101" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <nc:get>
        <nc:filter type="subtree">
            <native xmlns="http://cisco.com/ns/yang/ned/ios">
            </native>
        </nc:filter>
    </nc:get>
</nc:rpc>
]]>]]>
```

10. Note all the output which is your entire IOS XE router running configuration in XML. Copy the entire XML output code starting right *after* the NETCONF prompt to the very end *excluding* the trailing NETCONF prompt of:
]]>]]>



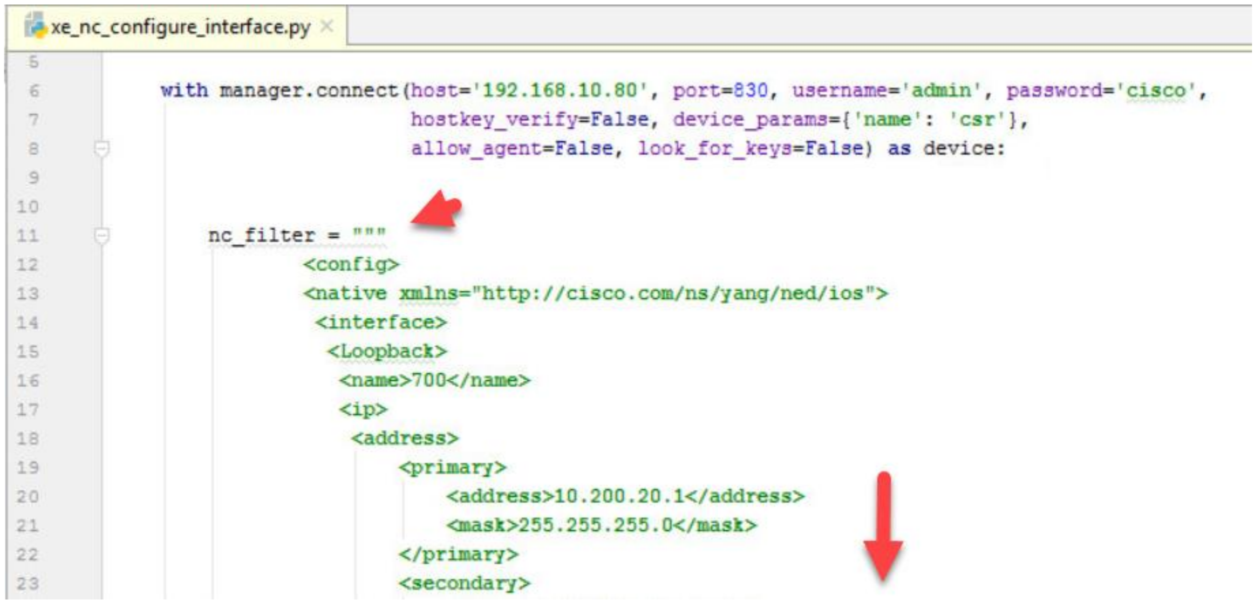
```
</nc:rpc>
]]>]]><?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101" xmlns:nc="urn:ietf:params:xml:
ns:netconf:base:1.0"><data><native xmlns="http://cisco.com/ns/yang/ned/ios"><device-model-version><major>2
</major><minor>1</minor><bug-fix>0</bug-fix></device-model-version><service><timestamps><debug><datetime><
msec/></datetime></debug><log><datetime><msec/></datetime></log></timestamps></service><platform><console>
<output>virtual</output></console></platform><enable><secret><type>5</type><secret>$1$KyJP$jE.tcqQOUWb02po
l/Jktz/</secret></secret></enable><username><name>admin</name><password><encryption>0</encryption><passwor
d>cisco</password></password></username><ip><forward-protocol><protocol>nd</protocol></forward-protocol><h
ttp><authentication><local/></authentication><server>true</server><secure-server>true</secure-server></htt
p><route><ip-route-interface-forwarding-list><prefix>15.0.0.0</prefix><mask>255.0.0.0</mask><fwd-list><fwd
>200.0.0.0</fwd></fwd-list></ip-route-interface-forwarding-list><ip-route-interface-forwarding-list><prefi
x>16.0.0.0</prefix><mask>255.0.0.0</mask><fwd-list><fwd>200.0.0.0</fwd></fwd-list></ip-route-interface-for
warding-list><ip-route-interface-forwarding-list><prefix>17.0.0.0</prefix><mask>255.0.0.0</mask><fwd-list>
<fwd>200.0.0.0</fwd></fwd-list></ip-route-interface-forwarding-list></route></ip><interface><GigabitEthern
et><name>4</name><negotiation><auto>true</auto></negotiation><cdp><enable>true</enable></cdp><ip><address>
<primary><address>192.168.10.80</address><mask>255.255.255.0</mask></primary></address></ip><mop><enabled>
false</enabled></mop></GigabitEthernet><GigabitEthernet><name>5</name><negotiation><auto>true</auto></nego
tiation><cdp><enable>true</enable></cdp><ip><address><primary><address>192.168.20.80</address><mask>255.25
5.255.0</mask></primary></address></ip><mop><enabled>false</enabled></mop></GigabitEthernet><Loopback><nam
e>100</name><ip><address><primary><address>8.8.8.1</address><mask>255.255.255.0</mask></primary></address>
</ip></Loopback><Loopback><name>500</name><ip><address><primary><address>6.6.8.1</address><mask>255.255.25
5.0</mask></primary></address></ip></Loopback><Loopback><name>600</name><ip><no-address><address>false</ad
dress></no-address></ip></Loopback></interface><diagnostic><bootup><level>minimal</level></bootup></diagno
stic><control-plane></control-plane><line><console><first>0</first><exec-timeout><minutes>0</minutes><seco
nds>0</seconds></exec-timeout><stopbits>1</stopbits></console><vty><first>0</first><exec-timeout><minutes>
0</minutes><seconds>0</seconds></exec-timeout><login><local/></login><transport><input><input>ssh</input><
/input></transport></vty><vty><first>1</first><last>4</last><login><local/></login><transport><input><inpu
t>ssh</input></input></transport></vty><vty><first>5</first><last>15</last></vty></line><multilink><bundle
-name>authenticated</bundle-name></multilink><ntp><server><server-list><ip-address>pool.ntp.org</ip-addres
s></server-list><server-list><ip-address>time-pnp.cisco.comi.</ip-address></server-list></server></ntp><re
dundancy></redundancy><spanning-tree><extend><system-id/></extend></spanning-tree><subscriber><templating/
></subscriber><crypto><pki><certificate><chain><name>TP-self-signed-3477724526</name><certificate><serial>
01</serial><certtype>self-signed</certtype></certificate><certificate><serial>quit</serial></certificate><
/chain></certificate><trustpoint><id>TP-self-signed-3477724526</id><enrollment><selfsigned/></enrollment><
revocation-check>none</revocation-check><rsakeypair>TP-self-signed-3477724526</rsakeypair><subject-name>cn
=IOS-Self-Signed-Certificate-3477724526</subject-name></trustpoint></pki></crypto><virtual-service><name>c
sr_mgmt</name></virtual-service><license><udi><pid>CSR1000V</pid><sn>9I43BHG5A4H</sn></udi></license><cdp>
<run/></cdp></native></data></rpc-reply>]]>]]>
```

11. Within your lab environment, go to any online XML formatter such as:
    https://xmlvalidator.com
    https://www.webtoolkitonline.com
12. Paste in the XML code to any validator tool and ensure that the XML is validated.
13. Copy and paste the formatted XML back into **Sublime**. You now have the entire configuration of your IOS XE device in properly formatted XML.
14. Save the file as **iosxe.xml** to your **Class_Share** directory.
15. Open **Win7** VM and open the saved **iosxe.xml** file in **Notepad++.** Read and study your entire router configuration in XML. Note carefully the hierarchy of XML entries.



```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <native
            xmlns="http://cisco.com/ns/yang/ned/ios">
            <device-model-version>
                <major>2</major>
                <minor>1</minor>
                <bug-fix>0</bug-fix>
            </device-model-version>
            <version>16.3</version>
            <boot-start-marker/>
            <boot-end-marker/>
            <service>
                <timestamps>
                    <debug>
                        <datetime>
                            <msec/>
                        </datetime>
                    </debug>
                    <log>
                        <datetime>
                            <msec/>
                        </datetime>
                    </log>
                </timestamps>
            </service>
            <platform>
                <console>
                    <output>auto</output>
                </console>
```

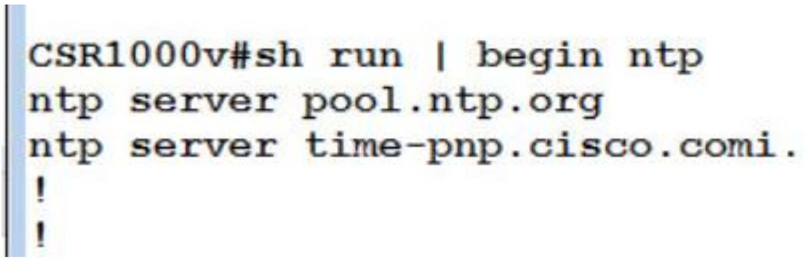
16. In **PyCharm**, open the Python file as needed from **Z:\Python**.
    `xe_nc_configure_interface.py`
17. Ensure you have the proper router credentials in your Python code from prior lab steps.
18. Take careful note of the **XML** that exists between the triple quotes **“””**.



xe_nc_configure_interface.py

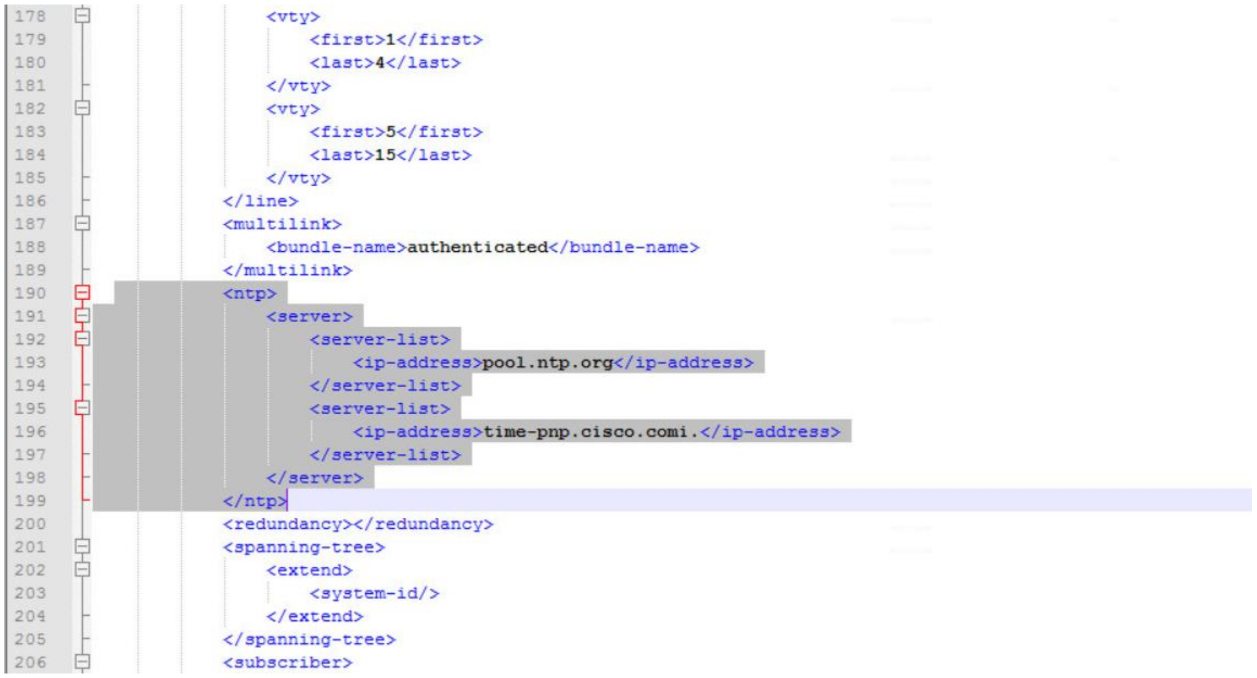
```

    with manager.connect(host='192.168.10.80', port=830, username='admin', password='cisco',
                         hostkey_verify=False, device_params={'name': 'csr'},
                         allow_agent=False, look_for_keys=False) as device:


        nc_filter = """
                <config>
                <native xmlns="http://cisco.com/ns/yang/ned/ios">
                 <interface>
                  <Loopback>
                   <name>700</name>
                   <ip>
                    <address>
                        <primary>
                            <address>10.200.20.1</address>
                            <mask>255.255.255.0</mask>
                        </primary>
                        <secondary>
```

19. At this point, you have all you need to make any router configuration change, based on your XML content and your Python script. As an example, review your existing NTP configurations on the **CSR1000v** device.



```
CSR1000v#sh run | begin ntp
ntp server pool.ntp.org
ntp server time-pnp.cisco.comi.
!
!
```

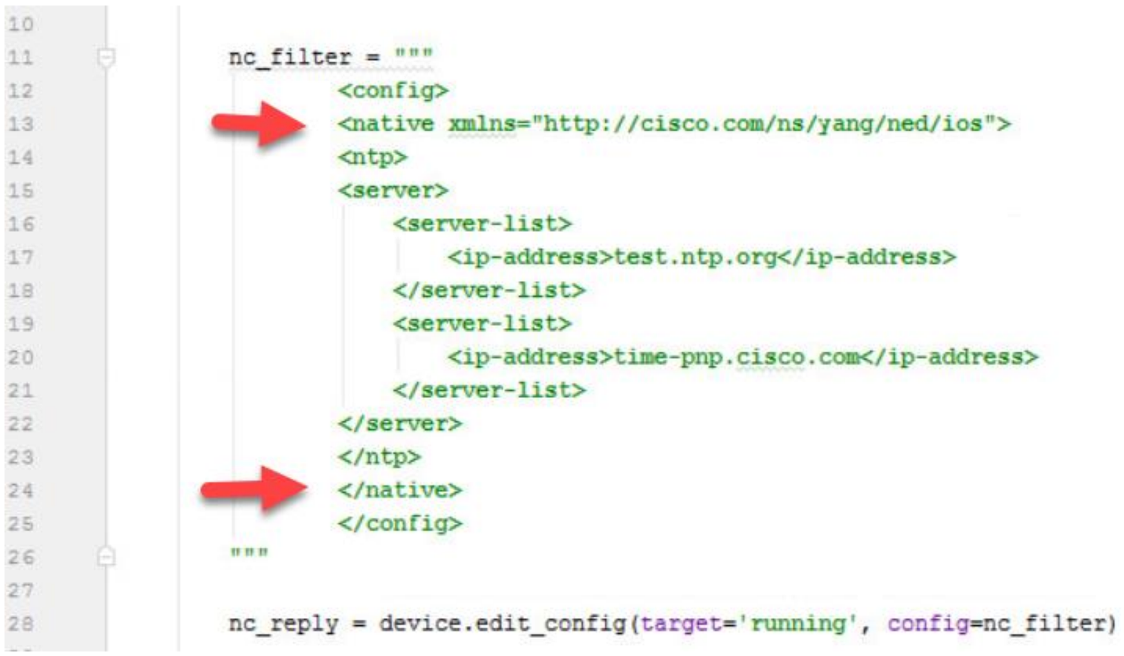
20. In **Notepad++**, highlight and copy exactly the NTP configuration in **XML**. Pay very close attention to every character that relates to NTP. Depending on your existing configuration, you may have to scroll down further then the line numbers in this screen shot.



```
178             <vty>
179                 <first>1</first>
180                 <last>4</last>
181             </vty>
182             <vty>
183                 <first>5</first>
184                 <last>15</last>
185             </vty>
186         </line>
187         <multilink>
188             <bundle-name>authenticated</bundle-name>
189         </multilink>
190         <ntp>
191             <server>
192                 <server-list>
193                     <ip-address>pool.ntp.org</ip-address>
194                 </server-list>
195                 <server-list>
196                     <ip-address>time-pnp.cisco.comi.</ip-address>
197                 </server-list>
198             </server>
199         </ntp>
200         <redundancy></redundancy>
201         <spanning-tree>
202             <extend>
203                 <system-id/>
204             </extend>
205         </spanning-tree>
206         <subscriber>
```

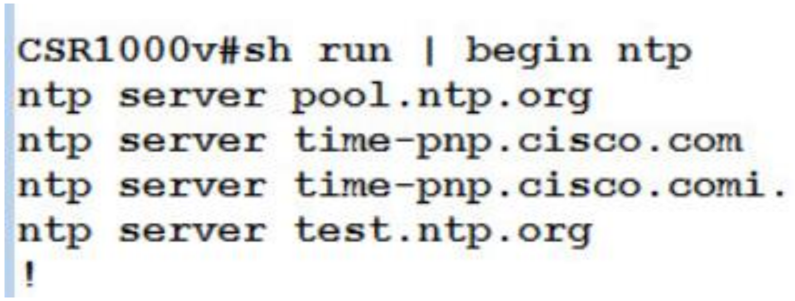
21. Paste the XML into your Python script after the first 2 lines and before the last two XML lines. You may have to clean up your indenting after pasting. As an example, change the ntp settings to:

```
test.ntp.org

time-pnp.cisco.com
```



```
nc_filter = """
        <config>
        <native xmlns="http://cisco.com/ns/yang/ned/ios">
        <ntp>
        <server>
            <server-list>
                <ip-address>test.ntp.org</ip-address>
            </server-list>
            <server-list>
                <ip-address>time-pnp.cisco.com</ip-address>
            </server-list>
        </server>
        </ntp>
        </native>
        </config>
"""

nc_reply = device.edit_config(target='running', config=nc_filter)
```

22. From **PyCharm**, run or step through your code in full and ensure no errors.
23. Return to the **CSR1000v** device and verify the two new added NTP entries.



```
CSR1000v#sh run | begin ntp
ntp server pool.ntp.org
ntp server time-pnp.cisco.com
ntp server time-pnp.cisco.comi.
ntp server test.ntp.org
!
```

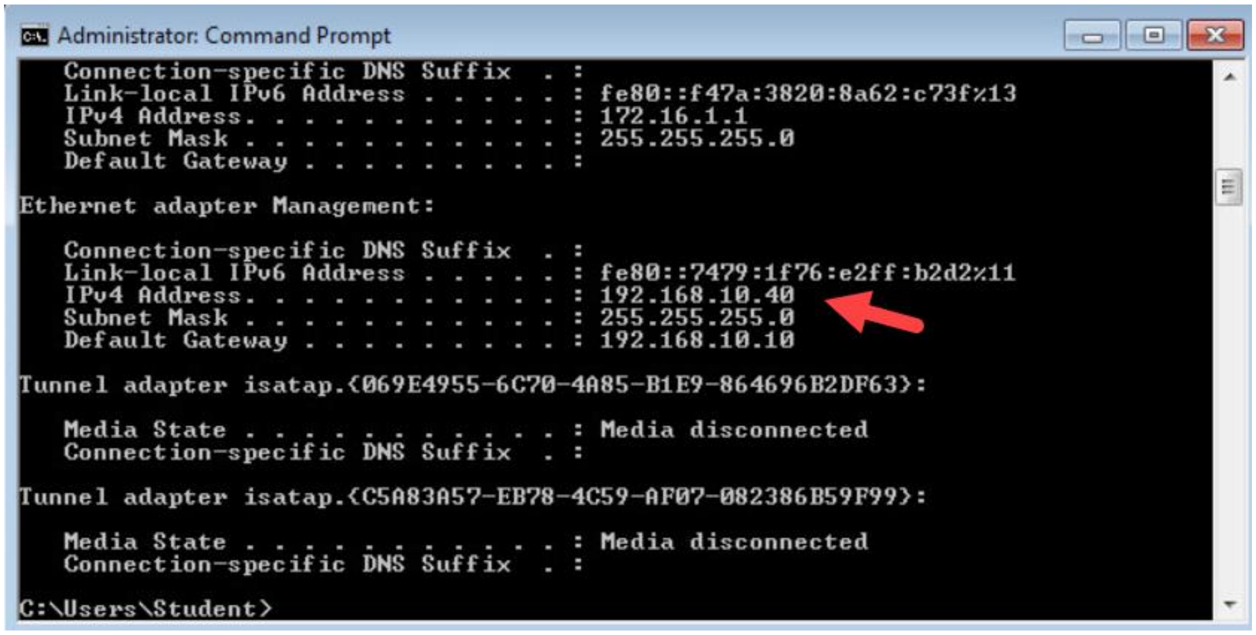
24. Optionally, take any other XML code from Notepad++ and repeat these steps to perform any other configuration change to your **CSR1000v**. Be very careful to get the *exact* XML code for each configuration change.

Task 4: Use Wireshark to view the network traffic from configuration.

In this task you will use **Wireshark** to view the network traffic of **NETCONF** and **RESTCONF.**

How-to Steps

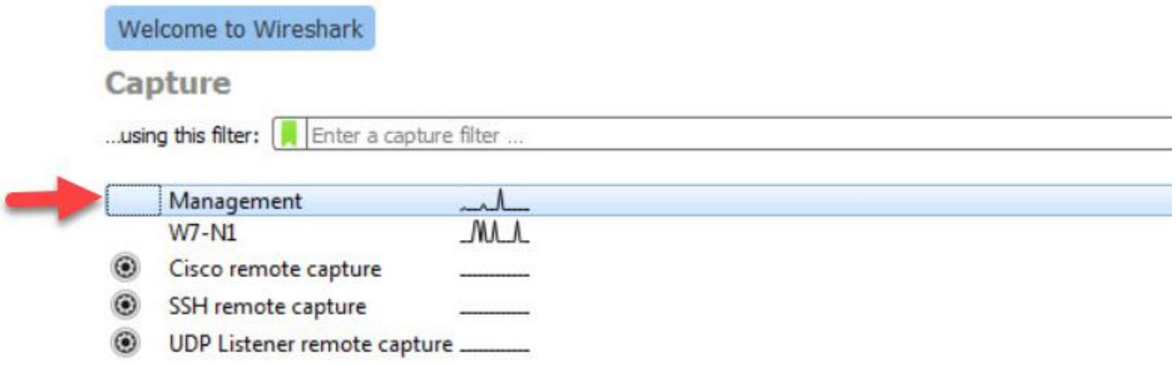
1. Connect to your **Win7** VM. Note the existing IP address of your Management interface with **ipconfig** from a **Command Prompt**.



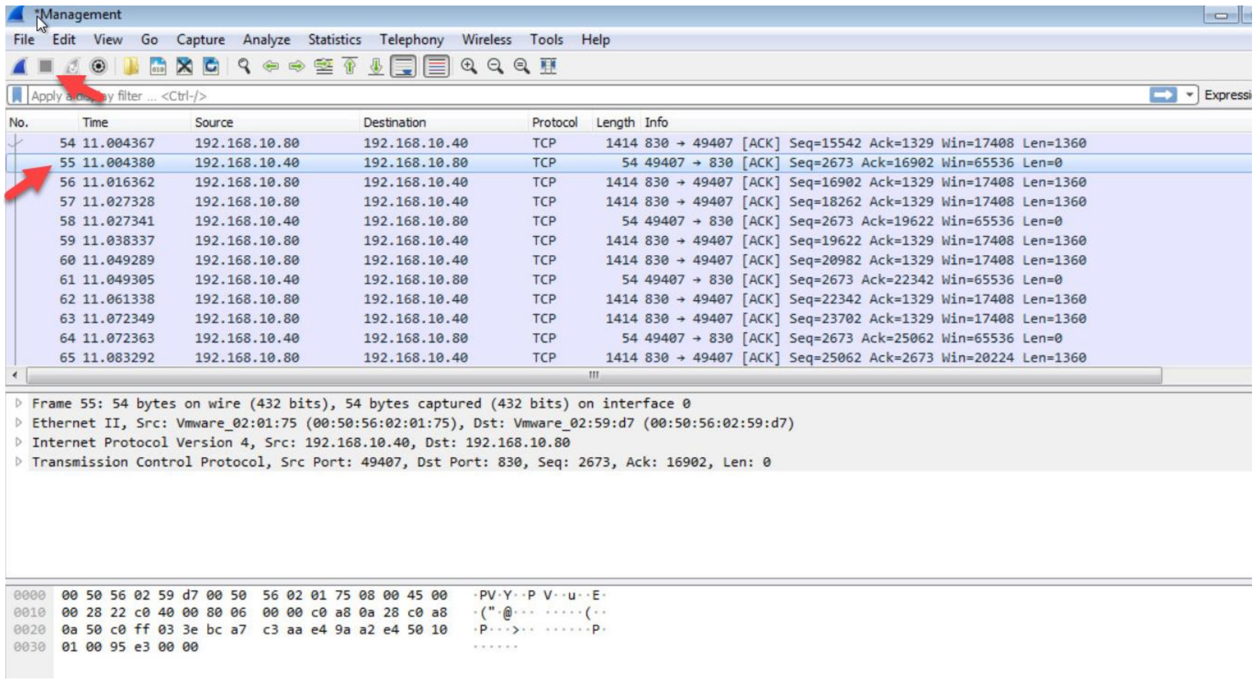
```
   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::f47a:3820:8a62:c73f%13
   IPv4 Address. . . . . . . . . . . : 172.16.1.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Ethernet adapter Management:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::7479:1f76:e2ff:b2d2%11
   IPv4 Address. . . . . . . . . . . : 192.168.10.40
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.10.10

Tunnel adapter isatap.{069E4955-6C70-4A85-B1E9-864696B2DF63}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter isatap.{C5A83A57-EB78-4C59-AF07-082386B59F99}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\Student>
```

2. Start **Wireshark**. You can download and install **Wireshark** on your **Win7** VM if it is not already installed.

3. Double click the **Management** interface.

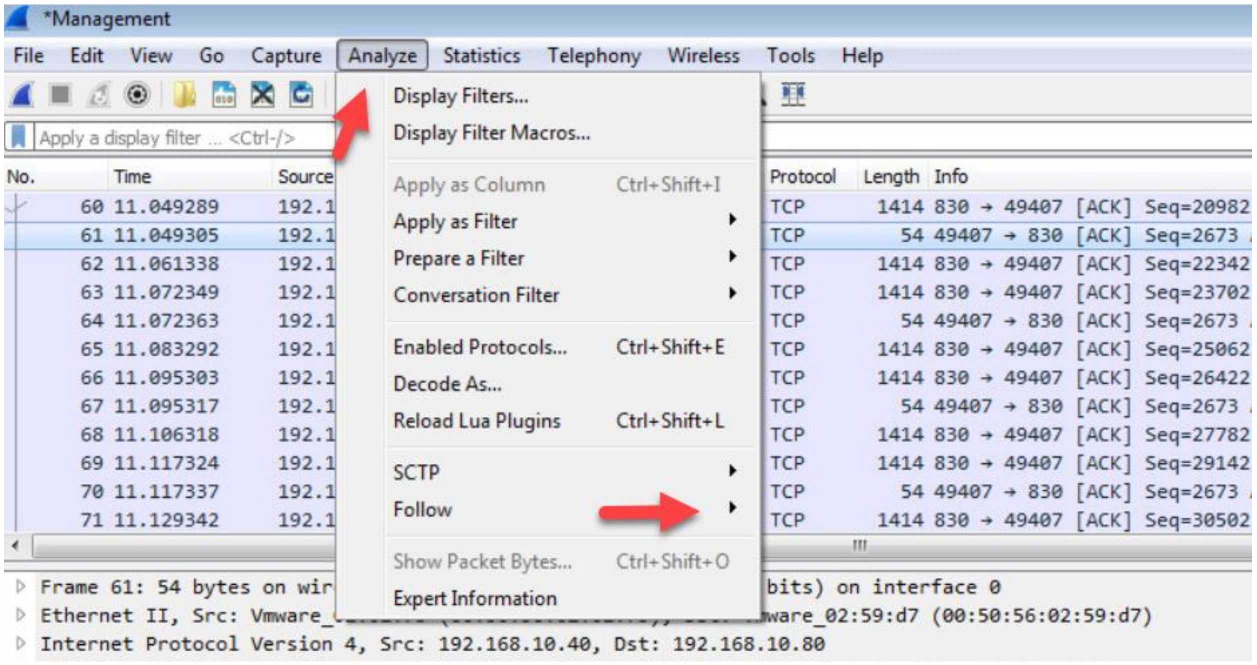


4. Return to **PyCharm**, and run your **NETCONF** script again from the last task.
5. Return to **Wireshark** and stop your capture. Select any packet in the **NETCONF** traffic.


*Management
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expressi

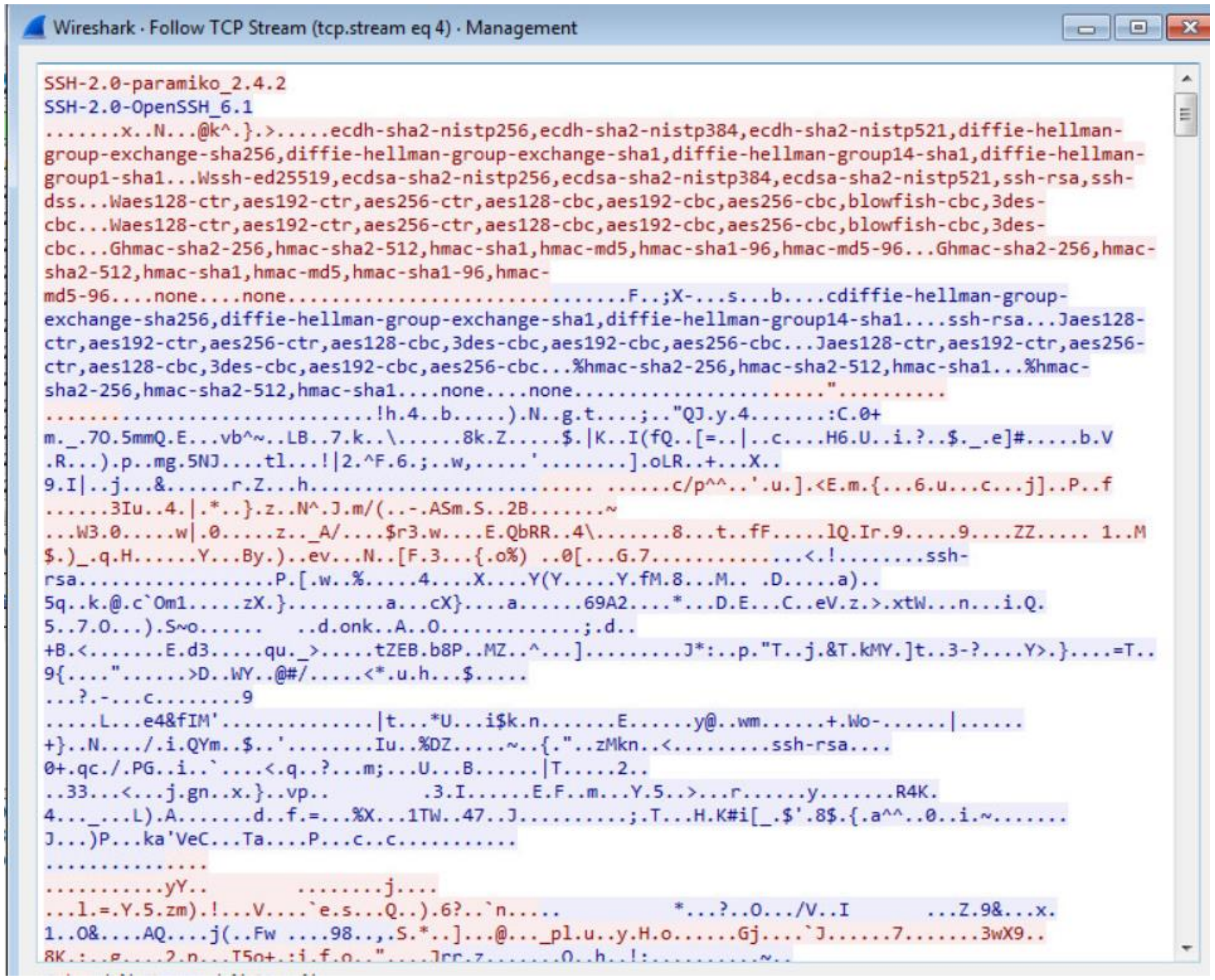
```
No.      Time        Source          Destination     Protocol Length Info
     54 11.004367    192.168.10.80   192.168.10.40   TCP      1414 830 → 49407 [ACK] Seq=15542 Ack=1329 Win=17408 Len=1360
     55 11.004380    192.168.10.40   192.168.10.80   TCP        54 49407 → 830 [ACK] Seq=2673 Ack=16902 Win=65536 Len=0
     56 11.016362    192.168.10.80   192.168.10.40   TCP      1414 830 → 49407 [ACK] Seq=16902 Ack=1329 Win=17408 Len=1360
     57 11.027328    192.168.10.80   192.168.10.40   TCP      1414 830 → 49407 [ACK] Seq=18262 Ack=1329 Win=17408 Len=1360
     58 11.027341    192.168.10.40   192.168.10.80   TCP        54 49407 → 830 [ACK] Seq=2673 Ack=19622 Win=65536 Len=0
     59 11.038337    192.168.10.80   192.168.10.40   TCP      1414 830 → 49407 [ACK] Seq=19622 Ack=1329 Win=17408 Len=1360
     60 11.049289    192.168.10.80   192.168.10.40   TCP      1414 830 → 49407 [ACK] Seq=20982 Ack=1329 Win=17408 Len=1360
     61 11.049305    192.168.10.40   192.168.10.80   TCP        54 49407 → 830 [ACK] Seq=2673 Ack=22342 Win=65536 Len=0
     62 11.061338    192.168.10.80   192.168.10.40   TCP      1414 830 → 49407 [ACK] Seq=22342 Ack=1329 Win=17408 Len=1360
     63 11.072349    192.168.10.80   192.168.10.40   TCP      1414 830 → 49407 [ACK] Seq=23702 Ack=1329 Win=17408 Len=1360
     64 11.072363    192.168.10.40   192.168.10.80   TCP        54 49407 → 830 [ACK] Seq=2673 Ack=25062 Win=65536 Len=0
     65 11.083292    192.168.10.80   192.168.10.40   TCP      1414 830 → 49407 [ACK] Seq=25062 Ack=2673 Win=20224 Len=1360

▷ Frame 55: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
▷ Ethernet II, Src: Vmware_02:01:75 (00:50:56:02:01:75), Dst: Vmware_02:59:d7 (00:50:56:02:59:d7)
▷ Internet Protocol Version 4, Src: 192.168.10.40, Dst: 192.168.10.80
▷ Transmission Control Protocol, Src Port: 49407, Dst Port: 830, Seq: 2673, Ack: 16902, Len: 0

0000  00 50 56 02 59 d7 00 50  56 02 01 75 08 00 45 00   ·PV·Y··P V··u··E·
0010  00 28 22 c0 40 00 80 06  00 00 c0 a8 0a 28 c0 a8   ·("·@··· ·····(··
0020  0a 50 c0 ff 03 3e bc a7  c3 aa e4 9a a2 e4 50 10   ·P···>·· ······P·
0030  01 00 95 e3 00 00                                  ······
```

6. From the **Wireshark** menu, select **Analyze > Follow > TCP Stream.**


*Management
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Display Filters...
Display Filter Macros...
Apply as Column Ctrl+Shift+I
Apply as Filter
Prepare a Filter
Conversation Filter
Enabled Protocols... Ctrl+Shift+E
Decode As...
Reload Lua Plugins Ctrl+Shift+L
SCTP
Follow
Show Packet Bytes... Ctrl+Shift+O
Expert Information

```
No.      Time        Source   Protocol Length Info
     60 11.049289    192.1    TCP      1414 830 → 49407 [ACK] Seq=20982
     61 11.049305    192.1    TCP        54 49407 → 830 [ACK] Seq=2673
     62 11.061338    192.1    TCP      1414 830 → 49407 [ACK] Seq=22342
     63 11.072349    192.1    TCP      1414 830 → 49407 [ACK] Seq=23702
     64 11.072363    192.1    TCP        54 49407 → 830 [ACK] Seq=2673
     65 11.083292    192.1    TCP      1414 830 → 49407 [ACK] Seq=25062
     66 11.095303    192.1    TCP      1414 830 → 49407 [ACK] Seq=26422
     67 11.095317    192.1    TCP        54 49407 → 830 [ACK] Seq=2673
     68 11.106318    192.1    TCP      1414 830 → 49407 [ACK] Seq=27782
     69 11.117324    192.1    TCP      1414 830 → 49407 [ACK] Seq=29142
     70 11.117337    192.1    TCP        54 49407 → 830 [ACK] Seq=2673
     71 11.129342    192.1    TCP      1414 830 → 49407 [ACK] Seq=30502

▷ Frame 61: 54 bytes on wir                    bits) on interface 0
▷ Ethernet II, Src: Vmware_                    ware_02:59:d7 (00:50:56:02:59:d7)
▷ Internet Protocol Version 4, Src: 192.168.10.40, Dst: 192.168.10.80
```

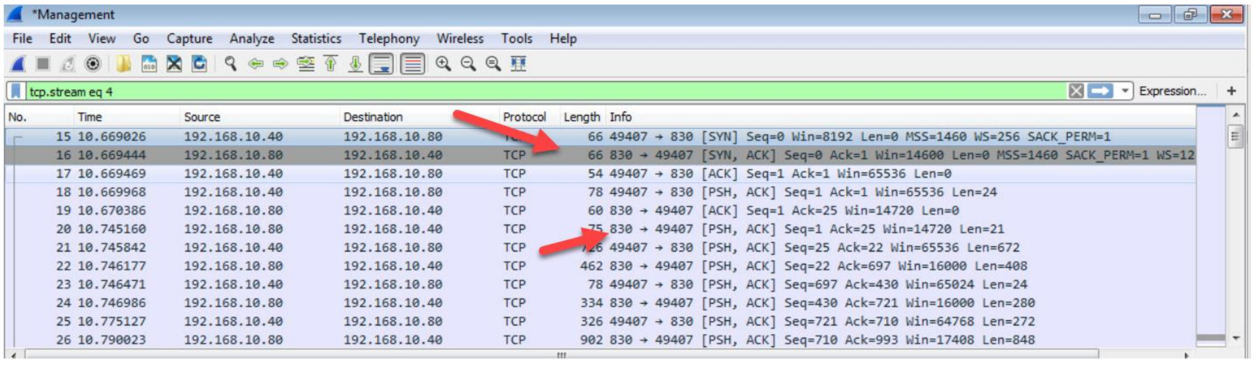
7. Since **NETCONF** must use encrypted data, you can see the SSH negotiation protocols, but the entire content of the configuration is encrypted.



Wireshark · Follow TCP Stream (tcp.stream eq 4) · Management

```
SSH-2.0-paramiko_2.4.2
SSH-2.0-OpenSSH_6.1
.......x..N...@k^.}.>.....ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-
group-exchange-sha256,diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1,diffie-hellman-
group1-sha1...Wssh-ed25519,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,ssh-rsa,ssh-
dss...Waes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,aes192-cbc,aes256-cbc,blowfish-cbc,3des-
cbc...Waes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,aes192-cbc,aes256-cbc,blowfish-cbc,3des-
cbc...Ghmac-sha2-256,hmac-sha2-512,hmac-sha1,hmac-md5,hmac-sha1-96,hmac-md5-96...Ghmac-sha2-256,hmac-
sha2-512,hmac-sha1,hmac-md5,hmac-sha1-96,hmac-
md5-96....none....none..............................F..;X-...s...b....cdiffie-hellman-group-
exchange-sha256,diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1....ssh-rsa...Jaes128-
ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc,aes192-cbc,aes256-cbc...Jaes128-ctr,aes192-ctr,aes256-
ctr,aes128-cbc,3des-cbc,aes192-cbc,aes256-cbc...%hmac-sha2-256,hmac-sha2-512,hmac-sha1...%hmac-
sha2-256,hmac-sha2-512,hmac-sha1....none....none........................."..........
..............................!h.4..b.....).N..g.t....;.."QJ.y.4.......:C.0+
m._.7O.5mmQ.E...vb^~..LB..7.k..\......8k.Z.....$.|K..I(fQ..[=..|..c....H6.U..i.?..$._.e]#.....b.V
.R...).p..mg.5NJ....tl...!|2.^F.6.;..w,.....'........].oLR..+...X..
9.I|..j...&......r.Z...h......................... ......c/p^^..'.u.].<E.m.{...6.u...c...j]..P..f
......3Iu..4.|.*..}.z..N^.J.m/(..-.ASm.S..2B.......~
...W3.0.....w|.0.....z.._A/....$r3.w....E.QbRR..4\.......8...t..fF.....1Q.Ir.9.....9....ZZ..... 1..M
$.)_.q.H......Y...By.)..ev...N..[F.3...{.o%) ..0[...G.7.............<.!........ssh-
rsa...................P.[.w..%.....4....X....Y(Y.....Y.fM.8...M.. .D.....a)..
5q..k.@.c`Om1.....zX.}.........a...cX}....a......69A2....*...D.E...C..eV.z.>.xtW...n...i.Q.
5..7.O...).S~o......   ..d.onk..A..O.............;.d..
+B.<.......E.d3.....qu._>.....tZEB.b8P..MZ..^...].........J*:..p."T..j.&T.kMY.]t..3-?....Y>.}....=T..
9{...."......>D..WY..@#/.....<*.u.h...$.....
...?.-...c........9
.....L...e4&fIM'..............|t...*U...i$k.n.......E......y@..wm......+.Wo-......|......
+}..N..../.i.QYm..$..'........Iu..%DZ.....~..{."..zMkn..<.........ssh-rsa....
0+.qc./.PG..i..`....<.q..?...m;...U...B......|T.....2..
..33...<...j.gn..x.}..vp..        .3.I......E.F..m...Y.5..>...r......y.......R4K.
4..._...L).A.......d..f.=...%X...1TW..47..J..........;.T...H.K#i[_.$'.8$.{.a^^..0..i.~.......
J...)P...ka'VeC...Ta....P...c..c...........
..............
...........yY..        ........j....
...1.=.Y.5.zm).!...V....`e.s...Q..).6?..`n.....          *...?..O.../V..I       ...Z.9&...x.
1..O&....AQ....j(..Fw ....98..,.S.*..]...@..._pl.u..y.H.o......Gj....`J......7.......3wX9..
8K.:..g....2.n...T5o+.:i.f.o.."....Jrr.z.......O..h..!:..........~..
```

8. **Close** the window when done. Note how you can see the **NETCONF** TCP 3 way handshake and the TCP port 830 traffic.



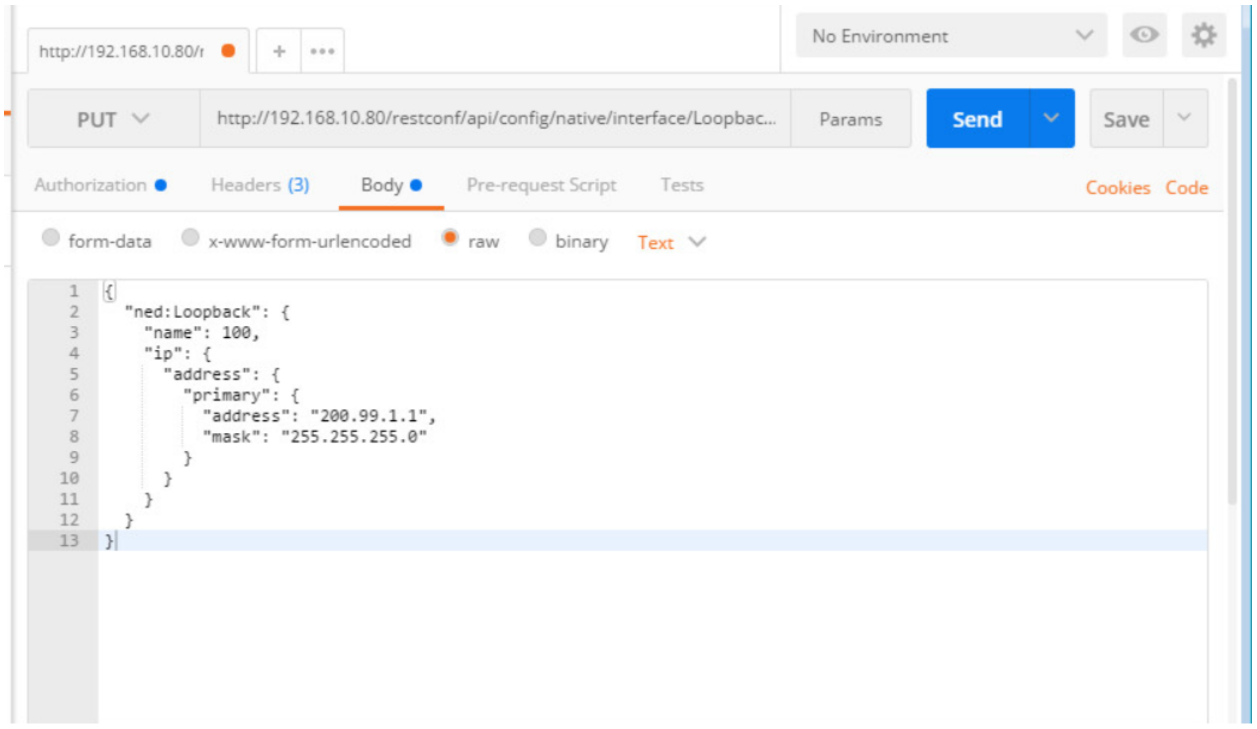
*Management

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...

tcp.stream eq 4

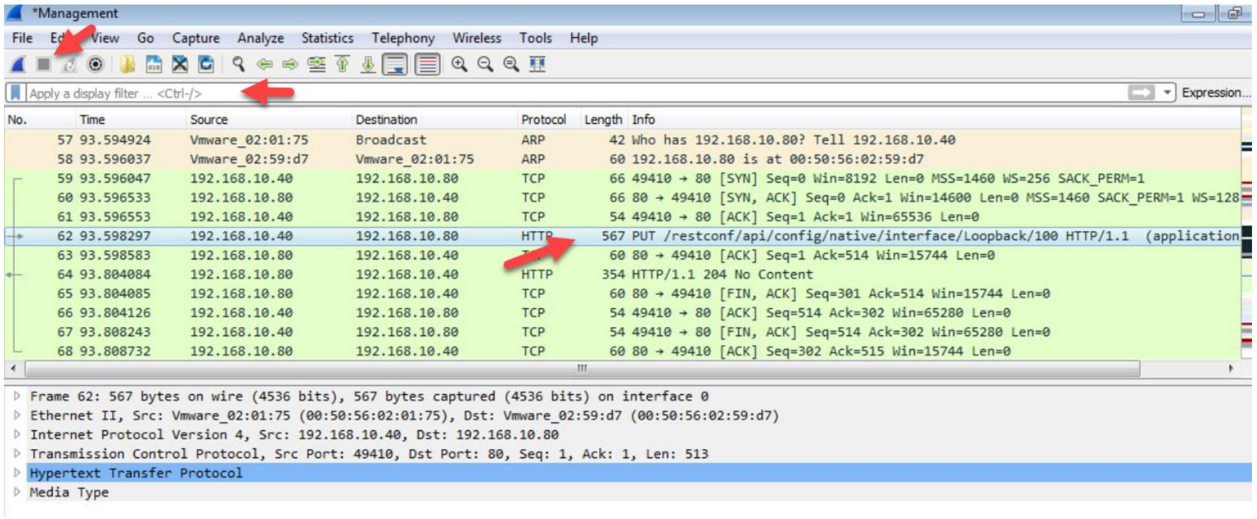
```
No.  Time       Source          Destination     Protocol  Length Info
  15 10.669026  192.168.10.40   192.168.10.80   TCP           66 49407 → 830 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
  16 10.669444  192.168.10.80   192.168.10.40   TCP           66 830 → 49407 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM=1 WS=12
  17 10.669469  192.168.10.40   192.168.10.80   TCP           54 49407 → 830 [ACK] Seq=1 Ack=1 Win=65536 Len=0
  18 10.669968  192.168.10.40   192.168.10.80   TCP           78 49407 → 830 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=24
  19 10.670386  192.168.10.80   192.168.10.40   TCP           60 830 → 49407 [ACK] Seq=1 Ack=25 Win=14720 Len=0
  20 10.745160  192.168.10.80   192.168.10.40   TCP           75 830 → 49407 [PSH, ACK] Seq=1 Ack=25 Win=14720 Len=21
  21 10.745842  192.168.10.40   192.168.10.80   TCP          726 49407 → 830 [PSH, ACK] Seq=25 Ack=22 Win=65536 Len=672
  22 10.746177  192.168.10.80   192.168.10.40   TCP          462 830 → 49407 [PSH, ACK] Seq=22 Ack=697 Win=16000 Len=408
  23 10.746471  192.168.10.40   192.168.10.80   TCP           78 49407 → 830 [PSH, ACK] Seq=697 Ack=430 Win=65024 Len=24
  24 10.746986  192.168.10.80   192.168.10.40   TCP          334 830 → 49407 [PSH, ACK] Seq=430 Ack=721 Win=16000 Len=280
  25 10.775127  192.168.10.40   192.168.10.80   TCP          326 49407 → 830 [PSH, ACK] Seq=721 Ack=710 Win=64768 Len=272
  26 10.790023  192.168.10.80   192.168.10.40   TCP          902 830 → 49407 [PSH, ACK] Seq=710 Ack=993 Win=17408 Len=848
```

9. Start a new Capture in **Wireshark**. You can save or delete the prior capture as you wish.
10. Return to Postman from the prior lab and run Postman again to change the Loopback interface.



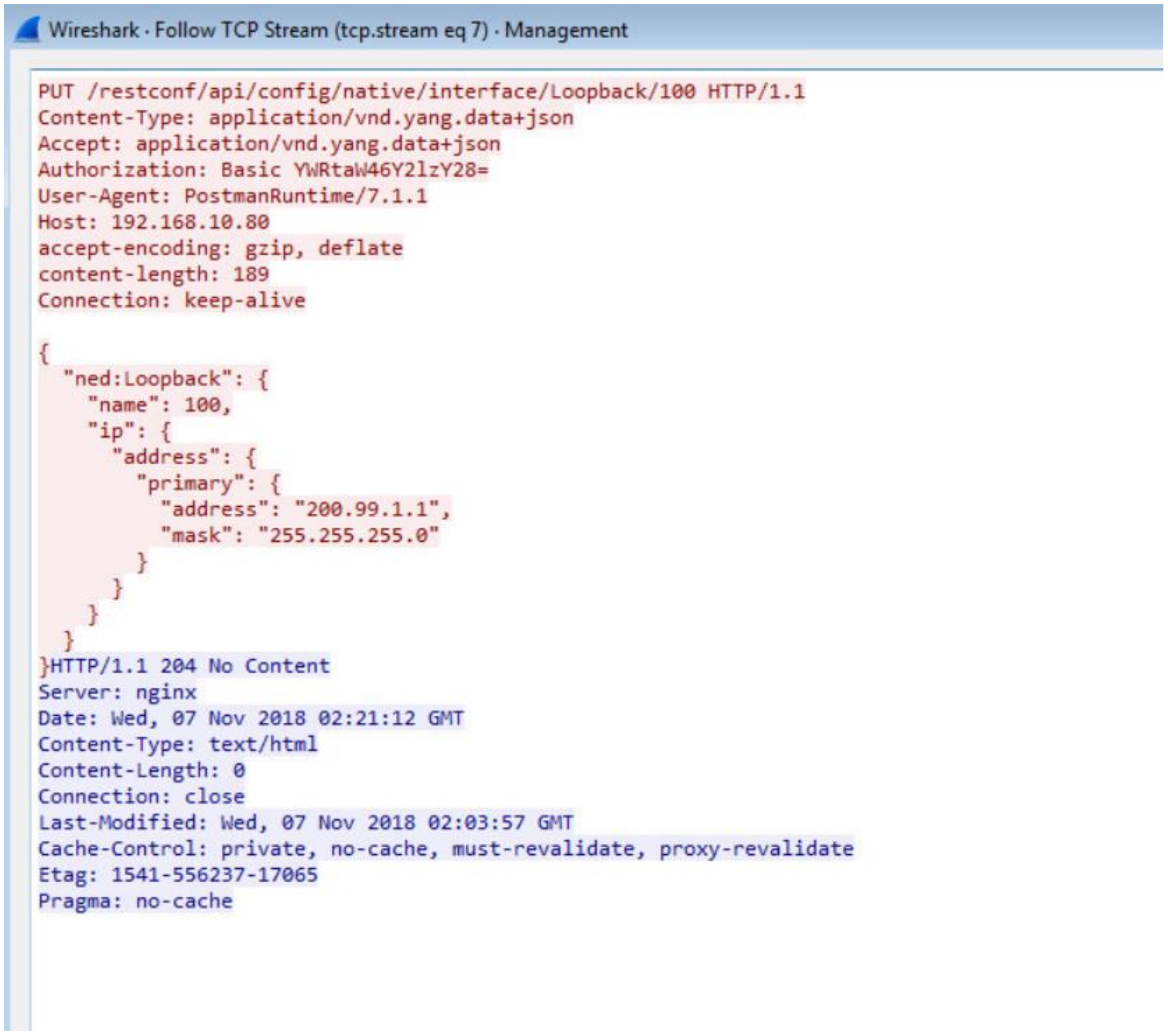
```
{
  "ned:Loopback": {
    "name": 100,
    "ip": {
      "address": {
        "primary": {
          "address": "200.99.1.1",
          "mask": "255.255.255.0"
        }
      }
    }
  }
}
```

11. Ensure you get a 2xx response.
12. Return to Wireshark and stop your capture. Delete the prior filter. Highlight any one frame in the **RESTCONF** configuration.



```
No.  Time       Source           Destination      Protocol Length Info
  57 93.594924  Vmware_02:01:75  Broadcast        ARP       42 Who has 192.168.10.80? Tell 192.168.10.40
  58 93.596037  Vmware_02:59:d7  Vmware_02:01:75  ARP       60 192.168.10.80 is at 00:50:56:02:59:d7
  59 93.596047  192.168.10.40    192.168.10.80    TCP       66 49410 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
  60 93.596533  192.168.10.80    192.168.10.40    TCP       66 80 → 49410 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM=1 WS=128
  61 93.596553  192.168.10.40    192.168.10.80    TCP       54 49410 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
  62 93.598297  192.168.10.40    192.168.10.80    HTTP     567 PUT /restconf/api/config/native/interface/Loopback/100 HTTP/1.1  (application
  63 93.598583  192.168.10.80    192.168.10.40    TCP       60 80 → 49410 [ACK] Seq=1 Ack=514 Win=15744 Len=0
  64 93.804084  192.168.10.80    192.168.10.40    HTTP     354 HTTP/1.1 204 No Content
  65 93.804085  192.168.10.80    192.168.10.40    TCP       60 80 → 49410 [FIN, ACK] Seq=301 Ack=514 Win=15744 Len=0
  66 93.804126  192.168.10.40    192.168.10.80    TCP       54 49410 → 80 [ACK] Seq=514 Ack=302 Win=65280 Len=0
  67 93.808243  192.168.10.40    192.168.10.80    TCP       54 49410 → 80 [FIN, ACK] Seq=514 Ack=302 Win=65280 Len=0
  68 93.808732  192.168.10.80    192.168.10.40    TCP       60 80 → 49410 [ACK] Seq=302 Ack=515 Win=15744 Len=0

▷ Frame 62: 567 bytes on wire (4536 bits), 567 bytes captured (4536 bits) on interface 0
▷ Ethernet II, Src: Vmware_02:01:75 (00:50:56:02:01:75), Dst: Vmware_02:59:d7 (00:50:56:02:59:d7)
▷ Internet Protocol Version 4, Src: 192.168.10.40, Dst: 192.168.10.80
▷ Transmission Control Protocol, Src Port: 49410, Dst Port: 80, Seq: 1, Ack: 1, Len: 513
▷ Hypertext Transfer Protocol
▷ Media Type
```

13. From the **Wireshark** menu, select **Analyze > Follow > TCP Stream.**
14. As this **RESTCONF** query was based on HTTP port 80, note how you can see the entire content of the configuration change. The router credentials are encrypted.


Wireshark · Follow TCP Stream (tcp.stream eq 7) · Management

```
PUT /restconf/api/config/native/interface/Loopback/100 HTTP/1.1
Content-Type: application/vnd.yang.data+json
Accept: application/vnd.yang.data+json
Authorization: Basic YWRtaW46Y2lzY28=
User-Agent: PostmanRuntime/7.1.1
Host: 192.168.10.80
accept-encoding: gzip, deflate
content-length: 189
Connection: keep-alive

{
  "ned:Loopback": {
    "name": 100,
    "ip": {
      "address": {
        "primary": {
          "address": "200.99.1.1",
          "mask": "255.255.255.0"
        }
      }
    }
  }
}HTTP/1.1 204 No Content
Server: nginx
Date: Wed, 07 Nov 2018 02:21:12 GMT
Content-Type: text/html
Content-Length: 0
Connection: close
Last-Modified: Wed, 07 Nov 2018 02:03:57 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Etag: 1541-556237-17065
Pragma: no-cache
```

15. While http is ideal to see the packet content for learning and troubleshooting, it is of course unsecure. HTTPS would be the secure RESTful choice.

Challenge results

This lab demonstrated extracting the source XML code of your **CSR1000v** router and editing the configuration of an IOS XE device with Python using NETCONF.