NETCONF/YANG Programming with YDK

Cisco Connect Croatia 2017

Djordje Vulovic
Consulting Systems Engineer
dvulovic@cisco.com

https://github.com/djordjevulovic/Cisco_Connect_HR_2017

Objective

- Get some real feeling about NETCONF/YANG programming with YANG Development Kit (YDK)
  - Install Python/YDK development environment
  - Run sample YDK application that:
    - Creates loopback interface
    - Creates BGP process
    - Create BGP neighbor
    - Advertise network over BGP
  - Use both IOS XR Native and OpenConfig YANG models

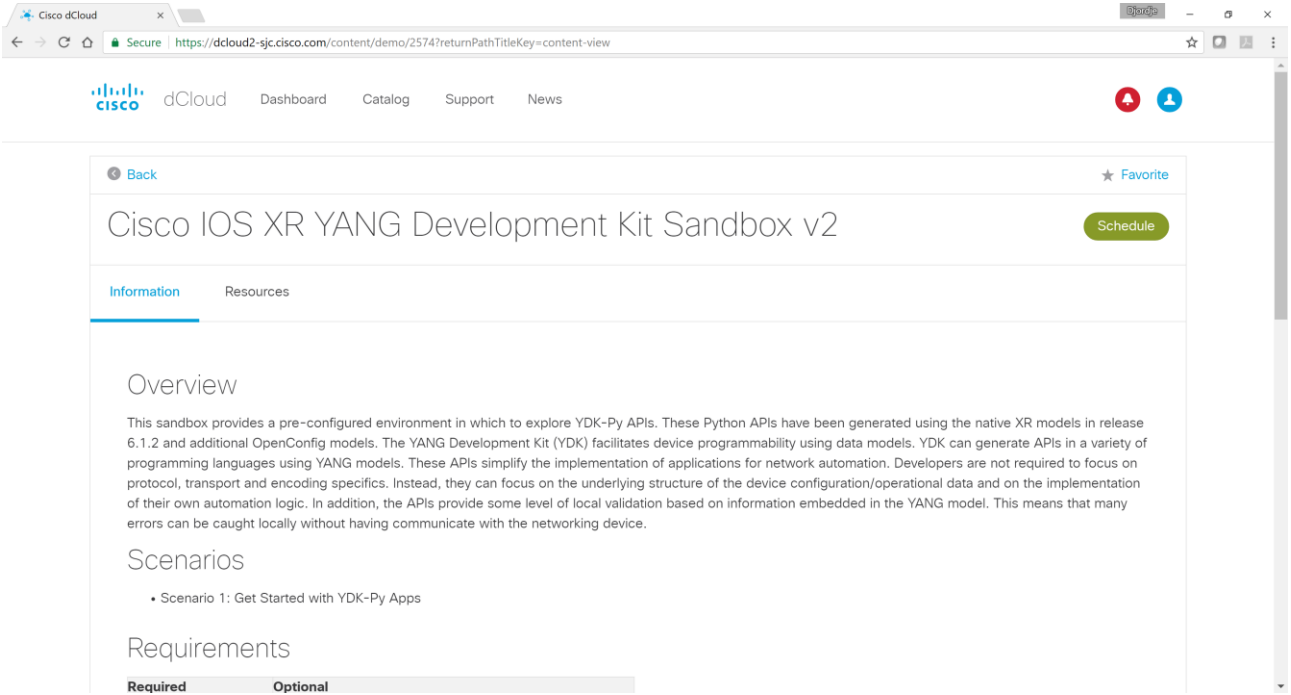
Requirements

- dCloud YDK Sandbox v2 Lab
  - https://dcloud2-sjc.cisco.com/content/demo/2574?returnPathTitleKey=content-view
- Python
  - https://www.python.org/downloads/
- PyCharm (optional by highly recommended)
  - https://www.jetbrains.com/pycharm/download/
- YDK
  - “pip install ydk-models-cisco-ios-xr” (see next slide for Windows)

Installing YDK on Windows

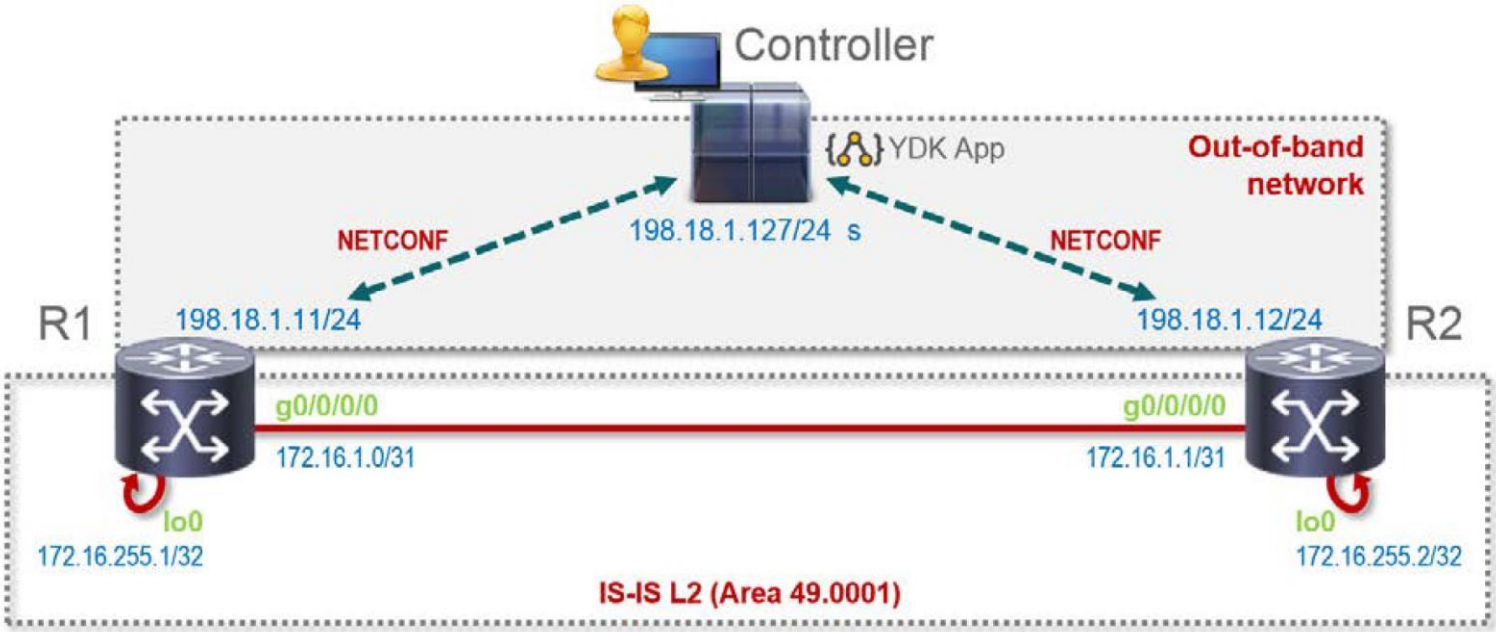
- YDK-PY by default requires exact version of lxml library (3.4.4)
- On Windows, building this package will likely fail
- Workaround:
  - Download YDK Source (e.g. https://github.com/CiscoDevNet/ydk-py/archive/master.zip)
  - In setup.py files, change from 'lxml==3.4.4‘ to 'lxml>=3.4.4‘
  - Install YDK from source (https://github.com/CiscoDevNet/ydk-py#installing-from-source)

dCloud Lab – YDK Sandbox v2 Lab



https://dcloud2-sjc.cisco.com/content/demo/2574?returnPathTitleKey=content-view

YDK SandBox Lab - Topology



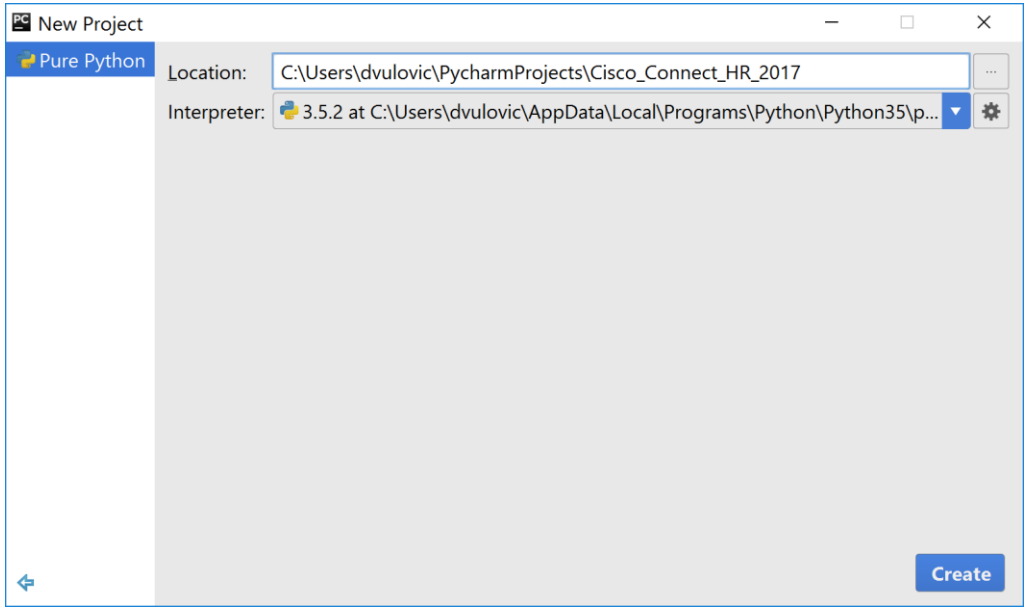
Application Workflow

- Step #1 – Create Loopback Interface
  - Lo 1111 with IP 1.1.1.1/24 on R1
  - Lo 2222 with IP 2.2.2.2/24 on R2
- Step #2 – Create BGP Process
  - AS 65000
- Step #3 – Create BGP Neighbor
  - Peer address is respective Loopback 0 IP address
- Step #4 – Advertise Network
  - 1.1.1.0/24 (R1) and 2.2.2.0/24 (R2)
- Step #5 – Add IPv4/Unicast SAFI to BGP Neighbor

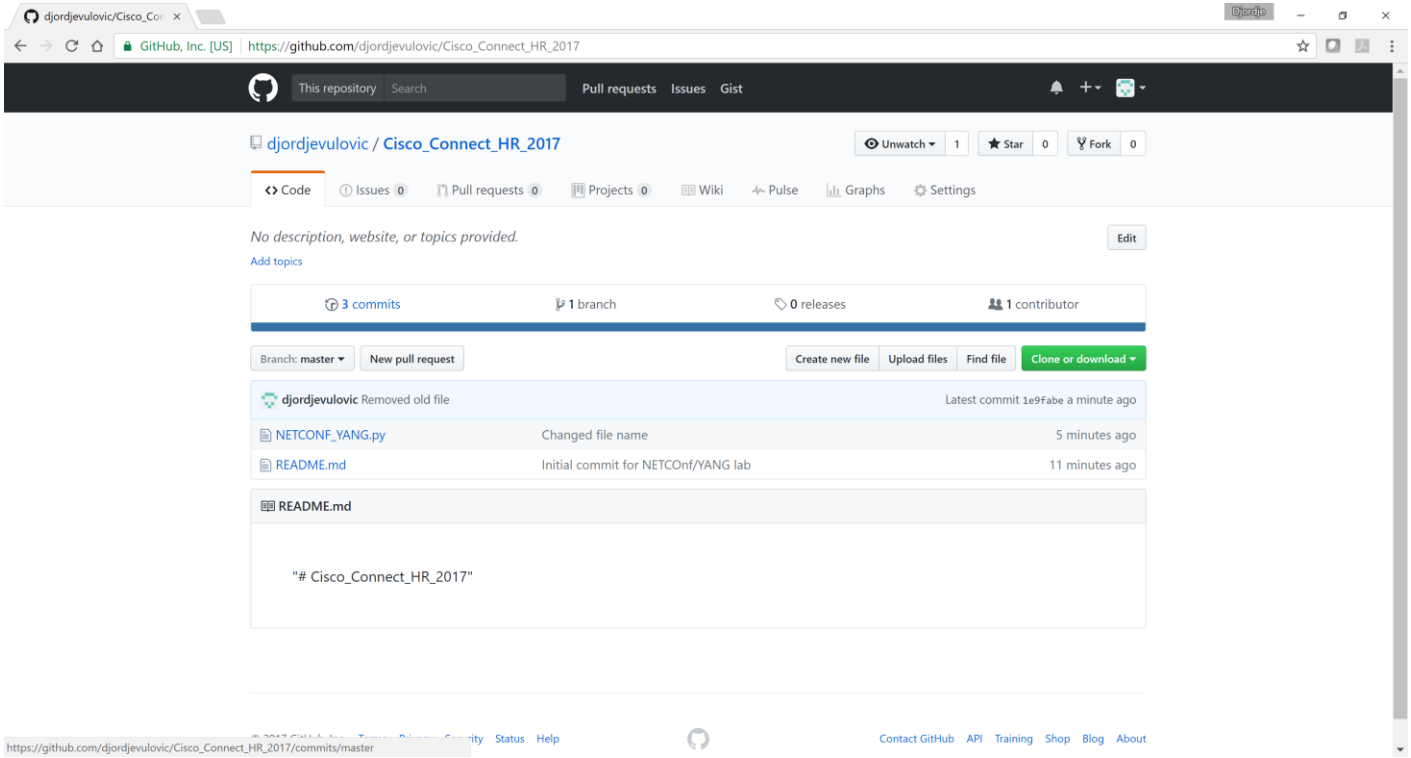
YANG Models Used for Configuration

Step	R1	R2
Step #1	IOS XR Native	OpenConfig
Step #2	IOS XR Native	OpenConfig
Step #3	IOS XR Native	OpenConfig
Step #4	IOS XR Native	IOS XR Native
Step #5	IOS XR Native	OpenConfig

PyCharm: Create New Project



GitHub: Download Application



https://github.com/djordjevulovic/Cisco_Connect_HR_2017/blob/master/NETCONF_YANG.py

PyCharm: App File into Project Directory

Cisco_Connect_HR_2017 - [C:\Users\dvulovic\PycharmProjects\Cisco_Connect_HR_2017] - ...\NETCONF_YANG.py - PyCharm Community Edition 2016.2

File Edit View Navigate Code Refactor Run Tools VCS Window Help

NETCONF YANG Lab

Cisco_Connect_HR_2017 / NETCONF_YANG.py

Project

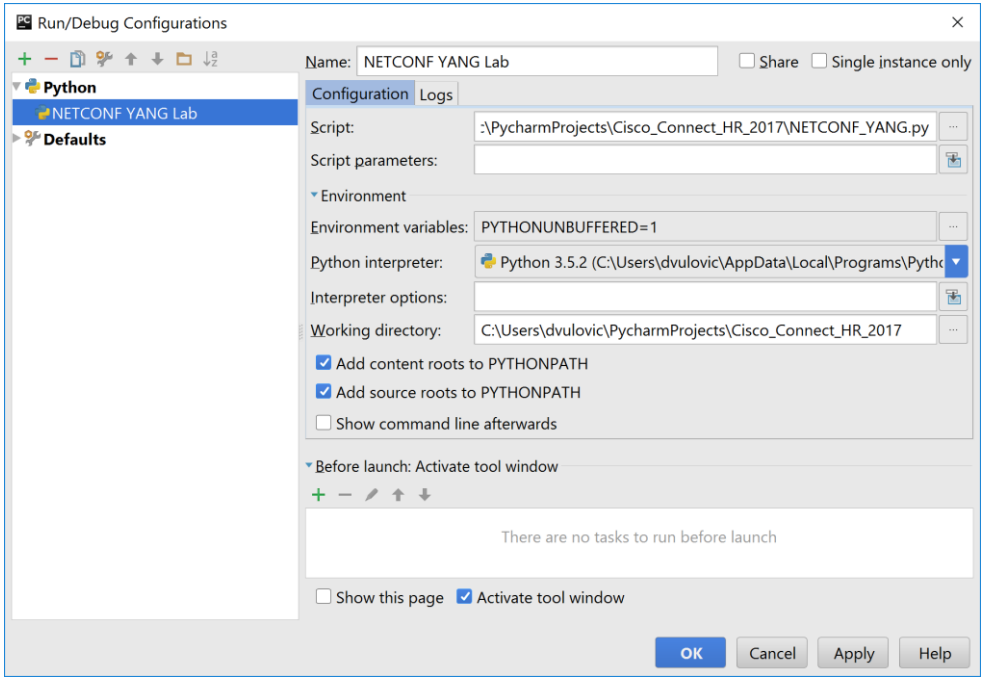
- Cisco_Connect_HR_2017 C:\Users\dvulovic\PycharmProjects\Cisco_
  - NETCONF_YANG.py
  - README.md
- External Libraries

NETCONF_YANG.py ×

```python
import logging
from ncclient import manager
from ydk.models.cisco_ios_xr import Cisco_IOS_XR_ifmgr_cfg, Cisco_IOS_XR_ipv4_bgp_cfg, Cisco_IOS_XR_ipv4_bgp_datatypes
from ydk.models.openconfig.openconfig_interfaces import Interfaces as OpenConfig_Interfaces
from ydk.models.openconfig.openconfig_bgp import Bgp as OpenConfig_Bgp
from ydk.models.openconfig.openconfig_bgp_types import Ipv4UnicastIdentity as OpenConfig_Ipv4UnicastIdentity
from ydk.models.ietf.iana_if_type import SoftwareloopbackIdentity
from ydk.models.cisco_ios_xr import Cisco_IOS_XR_ipv4_bgp_datatypes as xr_ipv4_bgp_datatypes
from ydk.services.crud_service import CRUDService
from ydk.providers.netconf_provider import NetconfServiceProvider
import ydk.types
from  ydk.models.ietf.ietf_interfaces import InterfaceTypeIdentity
##########################################################################
class dvulovic_NETCONF_Device:

    def __init__(self, host, port, username, password):

        self.host = host
        self.port = port
        self.username = username
        self.password = password

        self.m = manager.connect(host=self.host, port=self.port, username=self.username, password=self.password, hostkey_verify=False)

        self.ydk_provider = NetconfServiceProvider(address=self.host, port=self.port, username=self.username, password=self.password, protocol='ssh')

        self.ydk_crud = CRUDService()

class dvulovic_IOSXR(dvulovic_NETCONF_Device):

    def __init__(self, host, port, username, password):

        dvulovic_NETCONF_Device.__init__(self, host, port, username, password)

    def get_running_config(self, filter_string=None):
        if (filter_string is None):
            return self.m.get_config(source='running').data_xml
        else:
```

1:1 CRLF UTF-8 Git: master

PyCharm: Add Run Configuration



Step #1 – Create Loopback

Target Configuration in IOS XR CLI

```
interface Loopback1111
 ipv4 address 1.1.1.1 255.255.255.0
```

Step #1 – Create Loopback

Target Configuration in IOS XR Native Model

```
<interface-configurations xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg">
  <interface-configuration>
   <active>act</active>
   <interface-name>Loopback1111</interface-name>
   <interface-virtual/>
   <ipv4-network xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ipv4-io-cfg">
    <addresses>
     <primary>
      <address>1.1.1.1</address>
      <netmask>255.255.255.0</netmask>
     </primary>
    </addresses>
   </ipv4-network>
  </interface-configuration>
</interface-configurations>
```

Step #1 – Create Loopback

Python class method for IOS XR Native Model

```
class dvulovic_Native_IOSXR_Model_YDK(dvulovic_Generic_IOSXR_Model):

    def create_loopback(self, arg_loopbacknum, arg_ip, arg_mask):
        interface_configurations = Cisco_IOS_XR_ifmgr_cfg.InterfaceConfigurations()

        interface_configuration = interface_configurations.InterfaceConfiguration()

        interface_configuration.active = "act"
        interface_configuration.interface_name = "Loopback" + arg_loopbacknum
        interface_configuration.interface_virtual = ydk.types.Empty()

        primary_address = interface_configuration.ipv4_network.addresses.Primary()
        primary_address.address = arg_ip
        primary_address.netmask = arg_mask

        interface_configuration.ipv4_network.addresses.primary = primary_address

        interface_configurations.interface_configuration.append(interface_configuration)

        self.xr.ydk_crud.create(self.xr.ydk_provider, interface_configurations)
```

Step #1 – Create Loopback

Target Configuration in OpenConfig Model

```
<interfaces xmlns="http://openconfig.net/yang/interfaces">
  <interface>
   <name>Loopback1111</name>
   <config>
    <name>Loopback1111</name>
    <type xmlns:idx="urn:ietf:params:xml:ns:yang:iana-if-type">idx:softwareLoopback</type>
    <enabled>true</enabled>
   </config>
   <subinterfaces>
    <subinterface>
     <index>0</index>
     <ipv4 xmlns="http://openconfig.net/yang/interfaces/ip">
      <address>
       <ip>1.1.1.1</ip>
       <config>
        <ip>1.1.1.1</ip>
        <prefix-length>24</prefix-length>
       </config>
      </address>
     </ipv4>
    </subinterface>
   </subinterfaces>
  </interface>
</interfaces>
```

Step #1 – Create Loopback

Python class method for OpenConfig Model

```
class dvulovic_OpenConfig_IOSXR_Model_YDK(dvulovic_Generic_IOSXR_Model):

    def create_loopback(self, arg_loopbacknum, arg_ip, arg_prefixlen):
            oc_interface = OpenConfig_Interfaces.Interface()

            oc_interface.name = "Loopback" + arg_loopbacknum
            oc_interface.config.name = "Loopback" + arg_loopbacknum
            oc_interface.config.type = SoftwareloopbackIdentity()
            oc_interface.config.enabled = True

            oc_subinterface = oc_interface.subinterfaces.Subinterface()
            oc_subinterface.index = 0

            oc_subinterface_ipv4 = oc_subinterface.Ipv4()

            oc_subinterface_ipv4_address = oc_subinterface_ipv4.Address()
            oc_subinterface_ipv4_address.ip = arg_ip
            oc_subinterface_ipv4_address.config.ip = arg_ip
            oc_subinterface_ipv4_address.config.prefix_length = arg_prefixlen
            oc_subinterface_ipv4.address.append(oc_subinterface_ipv4_address)

            oc_subinterface.ipv4 = oc_subinterface_ipv4

            oc_interface.subinterfaces.subinterface.append(oc_subinterface)

            self.xr.ydk_crud.create(self.xr.ydk_provider, oc_interface)
```

Step #1 – Create Loopback

Python code to run step #1

```
# step 1 - create loopback interface
xr_native_model_r1.create_loopback("1111", "1.1.1.1", "255.255.255.0")
xr_oc_model_r2.create_loopback("2222","2.2.2.2", 24)
```

Step #1 – Create Loopback

Generated IOS XR CLI (R1)

```
RP/0/RP0/CPU0:r1#show configuration commit changes last 1
Thu Jan 26 13:21:12.899 UTC
Building configuration...
!! IOS XR Configuration version = 6.1.2
interface Loopback1111
 ipv4 address 1.1.1.1 255.255.255.0
!
end
```

Step #1 – Create Loopback

Generated IOS XR CLI (R2)

```
RP/0/RP0/CPU0:r2#show configuration commit changes last 1
Thu Jan 26 11:23:00.071 UTC
Building configuration...
!! IOS XR Configuration version = 6.1.2
interface Loopback2222
 ipv4 address 2.2.2.2 255.255.255.0
!
end
```

Step #2 – Create BGP Process

Python code to run step #2

```
# step 2 - create BGP process
xr_native_model_r1.create_bgp_process(65000)
xr_oc_model_r2.create_bgp_procces(65000)
```

Step #2 – Create BGP Process

Generated IOS XR CLI (R1)

```
RP/0/RP0/CPU0:r1#show configuration commit changes last 1
Thu Jan 26 13:24:44.804 UTC
Building configuration...
!! IOS XR Configuration version = 6.1.2
router bgp 65000
!
end

RP/0/RP0/CPU0:r1#
```

Step #2 – Create BGP Process

Generated IOS XR CLI (R2)

```
RP/0/RP0/CPU0:r2#show configuration commit changes last 1
Thu Jan 26 11:25:23.205 UTC
Building configuration...
!! IOS XR Configuration version = 6.1.2
router bgp 65000
!
end

RP/0/RP0/CPU0:r2#
```

Step #3 – Create BGP Neighbor

Target Configuration in IOS XR CLI

```
 router bgp 65000
 !
 neighbor 172.16.255.2
  remote-as 65000
  update-source Loopback0
  !
 !
!
```

Step #3 – Create BGP Neighbor

Target Configuration in IOS XR Native Model

```
<bgp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ipv4-bgp-cfg">
  <instance>
   <instance-name>default</instance-name>
   <instance-as>
    <as>0</as>
    <four-byte-as>
     <as>65000</as>
     <bgp-running/>
     <default-vrf>
      <bgp-entity>
       <neighbors>
        <neighbor>
         <neighbor-address>172.16.255.2</neighbor-address>
         <remote-as>
          <as-xx>0</as-xx>
          <as-yy>65000</as-yy>
         </remote-as>
         <update-source-interface>Loopback0</update-source-interface>
        </neighbor>
       </neighbors>
      </bgp-entity>
     </default-vrf>
    </four-byte-as>
   </instance-as>
  </instance>
 </bgp>
```

Step #3 – Create BGP Neighbor

Target Configuration in OpenConfig Model

```
<bgp xmlns="http://openconfig.net/yang/bgp">
  <global>
   <config>
    <as>65000</as>
   </config>
  </global>
  <neighbors>
   <neighbor>
    <neighbor-address>172.16.255.2</neighbor-address>
    <config>
     <neighbor-address>172.16.255.2</neighbor-address>
     <peer-as>65000</peer-as>
    </config>
    <transport>
     <config>
      <local-address>Loopback0</local-address>
     </config>
    </transport>
   </neighbor>
  </neighbors>
 </bgp>
```

Step #3 – Create BGP Neighbor

Python code to run step #3

```
# step 3 - create bgp neighbor
xr_native_model_r1.add_bgp_neighbor(65000,"172.16.255.2",65000, "Loopback0")
xr_oc_model_r2.add_bgp_neighbor(65000,"172.16.255.1",65000,"Loopback0")
```

Step #3 – Create BGP Neighbor

Generated IOS XR CLI (R1)

```
RP/0/RP0/CPU0:r1#show configuration commit changes last 1
Thu Jan 26 13:26:26.503 UTC
Building configuration...
!! IOS XR Configuration version = 6.1.2
router bgp 65000
 neighbor 172.16.255.2
  remote-as 65000
  update-source Loopback0
 !
!
end
```

Step #3 – Create BGP Neighbor

Generated IOS XR CLI (R2)

```
RP/0/RP0/CPU0:r2#show configuration commit changes last 1
Thu Jan 26 11:26:22.691 UTC
Building configuration...
!! IOS XR Configuration version = 6.1.2
router bgp 65000
 neighbor 172.16.255.1
  remote-as 65000
  update-source Loopback0
 !
!
end
```

Step #4 – Advertise Network

Python code to run step #4

```
# step 4 - advertise network
xr_native_model_r1.add_bgp_ipv4_unicast_network(65000,"1.1.1.0",24)
xr_native_model_r2.add_bgp_ipv4_unicast_network(65000,"2.2.2.0",24)
```

Step #4 – Advertise Network

Generated IOS XR CLI (R1)

```
RP/0/RP0/CPU0:r1#show configuration commit changes last 1
Thu Jan 26 13:30:43.825 UTC
Building configuration...
!! IOS XR Configuration version = 6.1.2
router bgp 65000
 address-family ipv4 unicast
  network 1.1.1.0/24
 !
!
end
```

Step #4 – Advertise Network

Generated IOS XR CLI (R2)

```
RP/0/RP0/CPU0:r2#show configuration commit changes last 1
Thu Jan 26 11:27:30.757 UTC
Building configuration...
!! IOS XR Configuration version = 6.1.2
router bgp 65000
 address-family ipv4 unicast
  network 2.2.2.0/24
 !
!
end
```

Step #5 – Add IPv4/Unicast SAFI

Python code to run step #5

```
# step 5 - add IPv4 Unicast SAFI to BGP neighbor
xr_native_model_r1.add_ipv4_unicast_SAFI_to_bgp_neighbor(65000, "172.16.255.2")
xr_oc_model_r2.add_ipv4_unicast_SAFI_to_bgp_neighbor(65000,"172.16.255.1")
```

Step #5 – Add IPv4/Unicast SAFI

Generated IOS XR CLI (R1)

```
RP/0/RP0/CPU0:r1#show configuration commit changes last 1
Thu Jan 26 13:31:26.675 UTC
Building configuration...
!! IOS XR Configuration version = 6.1.2
router bgp 65000
 neighbor 172.16.255.2
  address-family ipv4 unicast
  !
 !
!
end
```

Step #5 – Add IPv4/Unicast SAFI

Generated IOS XR CLI (R2)

```
RP/0/RP0/CPU0:r2#show configuration commit changes last 1
Thu Jan 26 11:29:00.024 UTC
Building configuration...
!! IOS XR Configuration version = 6.1.2
router bgp 65000
 neighbor 172.16.255.1
  address-family ipv4 unicast
  !
 !
!
end
```

The Result
BGP Table (R1)

```
RP/0/RP0/CPU0:r1#show bgp
Thu Jan 26 13:45:55.397 UTC
BGP router identifier 172.16.255.1, local AS number 65000
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 3
BGP main routing table version 3
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 1.1.1.0/24       0.0.0.0                  0         32768 i
*>i2.2.2.0/24       172.16.255.2             0    100      0 i

Processed 2 prefixes, 2 paths
```

The Result
BGP Table (R2)

```
RP/0/RP0/CPU0:r2#show bgp
Thu Jan 26 11:30:58.017 UTC
BGP router identifier 172.16.255.2, local AS number 65000
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000000   RD version: 3
BGP main routing table version 3
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
*>i1.1.1.0/24         172.16.255.1             0    100      0 i
*> 2.2.2.0/24         0.0.0.0                  0         32768 i

Processed 2 prefixes, 2 paths
```

The Result

Routing Tables (R1, R2)

```
RP/0/RP0/CPU0:r1#sh ip ro 2.2.2.0
Thu Jan 26 13:48:18.808 UTC

Routing entry for 2.2.2.0/24
  Known via "bgp 65000", distance 200, metric 0, type internal
  Installed Jan 26 13:44:44.525 for 00:03:35
  Routing Descriptor Blocks
    172.16.255.2, from 172.16.255.2
      Route metric is 0
  No advertising protos.
```

```
RP/0/RP0/CPU0:r2#sh ip ro 1.1.1.0
Thu Jan 26 11:33:10.905 UTC

Routing entry for 1.1.1.0/24
  Known via "bgp 65000", distance 200, metric 0, type internal
  Installed Jan 26 11:30:03.361 for 00:03:09
  Routing Descriptor Blocks
    172.16.255.1, from 172.16.255.1
      Route metric is 0
  No advertising protos.
```

The Result

Ping (R1, R2)

```
RP/0/RP0/CPU0:r1#ping 2.2.2.2 so 1.1.1.1
Thu Jan 26 13:48:49.520 UTC
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 3/3/5 ms
RP/0/RP0/CPU0:r1#
```

```
RP/0/RP0/CPU0:r2#ping 1.1.1.1 so 2.2.2.2
Thu Jan 26 11:34:48.511 UTC
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/5/9 ms
```

CISCO

Install lxml from Windows Binary

```
C:\Users\dvulovic\virtualenv\YDK>pip install lxml-3.7.3-cp35-cp35m-win_amd64.whl
Processing c:\users\dvulovic\virtualenv\ydk\lxml-3.7.3-cp35-cp35m-win_amd64.whl
Installing collected packages: lxml
  Found existing installation: lxml 3.7.2
    Uninstalling lxml-3.7.2:
      Successfully uninstalled lxml-3.7.2
Successfully installed lxml-3.7.3

C:\Users\dvulovic\virtualenv\YDK>
```

Libxml binaries for Windows can be found on
http://www.lfd.uci.edu/~gohlke/pythonlibs/#lxml

Activate virtualenv

```
C:\Users\dvulovic\virtualenv\YDK>.\Scripts\activate.bat
(YDK) C:\Users\dvulovic\virtualenv\YDK>
```

Upgrade PIP on Windows

```
(YDK) C:\Users\dvulovic\virtualenv\YDK>easy_install -U pip
Searching for pip
Reading https://pypi.python.org/simple/pip/
Best match: pip 9.0.1
Downloading
https://pypi.python.org/packages/11/b6/abcb525026a4be042b486df43905d6893fb04f05aac21c32c638e939e447/pip-
9.0.1.tar.gz#md5=35f01da33009719497f01a4ba69d63c9
Processing pip-9.0.1.tar.gz
Writing C:\Users\dvulovic\AppData\Local\Temp\easy_install-rh3_0uly\pip-9.0.1\setup.cfg
Running pip-9.0.1\setup.py -q bdist_egg --dist-dir C:\Users\dvulovic\AppData\Local\Temp\easy_install-
rh3_0uly\pip-9.0.1\egg-dist-tmp-3dh7l_wt
C:\Users\dvulovic\AppData\Local\Programs\Python\Python35\lib\distutils\dist.py:261: UserWarning: Unknown
distribution option: 'python_requires'
  warnings.warn(msg)
…
creating c:\users\dvulovic\virtualenv\ydk\lib\site-packages\pip-9.0.1-py3.5.egg
Extracting pip-9.0.1-py3.5.egg to c:\users\dvulovic\virtualenv\ydk\lib\site-packages
Adding pip 9.0.1 to easy-install.pth file
Installing pip-script.py script to C:\Users\dvulovic\virtualenv\YDK\Scripts
Installing pip.exe script to C:\Users\dvulovic\virtualenv\YDK\Scripts
Installing pip3-script.py script to C:\Users\dvulovic\virtualenv\YDK\Scripts
Installing pip3.exe script to C:\Users\dvulovic\virtualenv\YDK\Scripts
Installing pip3.5-script.py script to C:\Users\dvulovic\virtualenv\YDK\Scripts
Installing pip3.5.exe script to C:\Users\dvulovic\virtualenv\YDK\Scripts

Installed c:\users\dvulovic\virtualenv\ydk\lib\site-packages\pip-9.0.1-py3.5.egg
Processing dependencies for pip
Finished processing dependencies for pip
```

Install lxml from Windows Binary in VirtualEnv

```
(YDK) C:\Users\dvulovic\virtualenv\YDK>pip -v install lxml-3.7.3-cp35-cp35m-win_amd64.whl
Config variable 'Py_DEBUG' is unset, Python ABI tag may be incorrect
Config variable 'WITH_PYMALLOC' is unset, Python ABI tag may be incorrect
Processing c:\users\dvulovic\virtualenv\ydk\lxml-3.7.3-cp35-cp35m-win_amd64.whl
Installing collected packages: lxml

Successfully installed lxml-3.7.3
Cleaning up...

(YDK) C:\Users\dvulovic\virtualenv\YDK>
```

Libxml binaries for Windows can be found on http://www.lfd.uci.edu/~gohlke/pythonlibs/#lxml