**Unit Six Lab**

Directions: Complete each of the scripts below and submit them by the date specified in the assignment sheet and Blackboard. Submit **copies of your code** and **screenshots of the code running with each task**. Also, be sure to use the document, **Script Requirements as a guide** to writing good code. **Full credit will not be earned if you do not meet these script requirements.** **40 points**

**Note: Since we are running APIs on the NXOS switches in this lab, be sure to run the feature nxapi command from global configuration on each switch before attempting to send and API request.**

1. **Create a script with a nested dictionary for your two NXOS switches, dist-sw01 and dist-sw02. Have your key/value pairs be, hostname: nameOfDevice, deviceType: switch or router, mgmtIP: x.x.x.x For example:**

**hostname: dist-sw01 deviceType: switch mgmtIP: 10.10.20.177**

**Then have your script iterate the dictionary and print the results in a table:**

Host Type Mgmt IP

dist-sw01 switch 10.10.20.177

dist-sw02 switch 10.10.20.178

1. **Log into CML and make a reservation. Run the NX-API Sandbox and generate code for a show version command. Copy the code to your new script and modify the url, username, password, and request line appropriately as shown in your earlier lab. Test that your code runs for a single switch. If it does, copy the above script to your new script and modify it so that you print out a table that displays the hostname from the show version output, the memory and memory type, the chassis that is being used, and the boot file name (kick\_file\_name).**
2. **Using the NX-API Sandbox, generate code for a show ip ospf neighbor command on one of your NEXUS switches. Write a script that uses the nested dictionary in number 1 to show a useful printout of your ospf neighbors by device. Each device should display its neighbors and the router ID ,IP address of each neighbor, and interface. A sample output suggestion is shown below (note: Your addresses and interfaces will be different. This is simply a format suggestion):**

dist-sw01 OSPF Neighbors

Router-ID Neighbor IP Int

172.16.1.1 172.13.2.1 Eth1/1

172.16.2.1 172.16.12.3 Eth1/2

dist-sw02 OSPF Neighbors

Router-ID Neighbor IP Int

172.16.1.2 172.13.2.2 Eth1/1

172.16.2.2 172.16.12.4 Eth1/2

**Hint: Test your script in small parts. Try to get the printout for one device first, like dist-sw01. You will be iterating objects on a single device. Because in the end, you will be iterating your nested dictionary as well, you will have nested loops in your script. You take your working code for one device, place it in your final loop. For review of nested loops, see nested for loops in the For Loop section of the w3schools website we used earlier in the semester.**

**Your script structure will be something like the following:**

import requests

import json

def getOSPFNeighbor(mgmtIPAddress):

#This code is mostly generated by the NXOSAPI and gets a dictionary of OSPF neighbors. The dictionary (response) should be returned from this function to be used by the print function. mgmtIPaddress is received and used to ,modify the URL so that the code can be run on whatever device’s IP is received

def printOSPFNeighbor(OSPF\_Neighbor\_dict) :

#You generate this code which receives the response dictionary from a show ip ospf neighor NEXOSAPI call and prints it in a desired format (the format specified for the assignment). This function does not need to return anything. It just prints results by iterating the nested dictionary.

#Below is the main script

devices = {

#This is your nested dictionary created in step 1

}

#This for loop returns a mgmtIP for each device in the nested dictionary, devices (shown above) and sends it to a function that you define called, getOSPFNeighbor(). This function uses the IP address to modify the URL and uses the NXOSAPI code to generate a dictionary, OSPF\_Neighbor, that you return and send to a function to print out in the format shown above in the directions.

for device in devices:

mgmtIP = device[“mgmtIP’]

OSPF\_Neighbor = getOSPFNeighbor(mgmtIP)

printOSPFNeighbor(OSPF\_Neighbor)