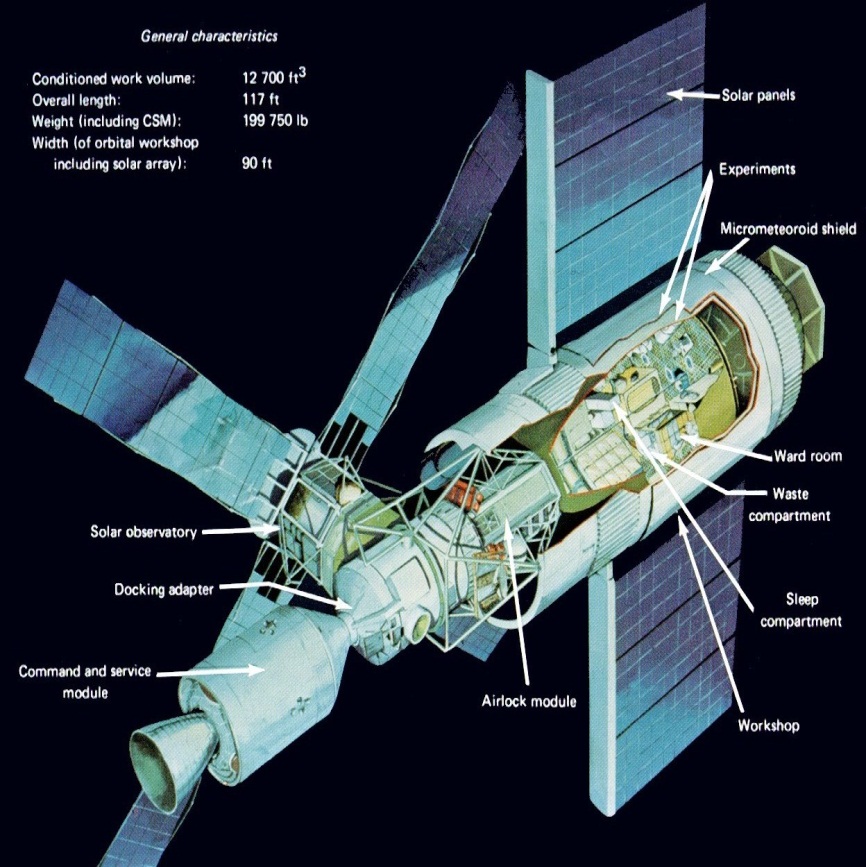
|  |
| --- |
| Nexus Access  Programer’s Guide |
| Programer’s Guide |
| A Guide to the NexusAccess\_m.py python interface. Nexus Access Software is provides to illustrate some of the uses of the Python Library on the Nexus 3K platform. Software is provided “As Is” and there is no implied warrantee. |
|  |



Nexus Access – Programer’s Guide

Contents

[I. Overview 3](#_Toc339527678)

[II. Socket Server 3](#_Toc339527679)

[1. Modules: 3](#_Toc339527680)

[2. Initialization 4](#_Toc339527681)

[3. Main Code loop 5](#_Toc339527682)

[III. Socket Client 6](#_Toc339527683)

[1. Program Structure 6](#_Toc339527684)

[a. Main file (NexusAccess\_m.py) 6](#_Toc339527685)

[b. Nexus Object file (Nexus\_Object.py) 6](#_Toc339527686)

[c. Supporting command files 6](#_Toc339527687)

[d. Output files 6](#_Toc339527688)

[2. Main file 6](#_Toc339527689)

[a. Initilization 6](#_Toc339527690)

[b. Sub-routines 7](#_Toc339527691)

[c. Menu - Based on the Python “Curses” package 9](#_Toc339527692)

[d. Main Loop 10](#_Toc339527693)

[3. Menu Structure 11](#_Toc339527694)

[IV. CLI Interface 12](#_Toc339527695)

[a. Script command (s) 12](#_Toc339527696)

[b. Queue command (q) 12](#_Toc339527697)

[c. Log file command (l) 13](#_Toc339527698)

[d. Interface command (i) 13](#_Toc339527699)

[e. Buffer (b) 14](#_Toc339527700)

[f. Routing (r) 14](#_Toc339527701)

[V. Multi-Chassis Features 15](#_Toc339527702)

[1. Menu 15](#_Toc339527703)

[2. Script 15](#_Toc339527704)

[3. 0, 1, 2 16](#_Toc339527705)

[4. M 16](#_Toc339527706)

[VI. Include files 18](#_Toc339527707)

[a. Nexus Object file: 18](#_Toc339527708)

[VII. Script Files 20](#_Toc339527709)

[b. script.txt 20](#_Toc339527710)

[c. mscript.txt 20](#_Toc339527711)

[d. mrun.txt (mscript.txt output) 20](#_Toc339527712)

# Overview

The Nexus Access project goal is to provide a frame work to demonstrate the Nexus Python Interface with a Client🡪 Server connection.

* 1. The Client is on a Linux machine.
  2. The Server is on the Nexus 3K chassis.

Communication is over the myPort=50007 port and to the myHost = 172.25.187.155. Pending connects is set at 5 maximum.

Client

NexusAccess\_m.py

Nexus\_Object.py

script.txt

mscript.txt

N3K Host [0]

SocketServer\_x.py

N3K Host [0]

SocketServer\_x.py

N3K Host [0]

SocketServer\_x.py

**Client Server Socket Connections**

# Socket Server

### Modules:

* + 1. Nexus 3K Python library CLI Object in addition to the Standard Python libraries for date and time, socket.

from cisco import \*

from datetime import datetime

import time

from socket import \* # get socket and constants

from sys import \*

from string import join

### Initialization

* 1. Set variables myPort and default myHost

myHost = '172.25.187.155' # '' = all available interfaces on host

myPort = 50007 # listen on a non-reserved port number

* 1. Get host IP and save the IP to myHost:

# get ouput of CLI command

oCli = CLI("show int mgmt0 brief | grep mgmt0", False)

buffer = repr(oCli.get\_output())

buffer = buffer.replace ('[','')

buffer = buffer.replace ("']",'')

buffer = buffer.replace ("'"," ")

buffer = buffer.replace ('\\',' ')

buffer = buffer.replace (' ,','\n')

buffer = buffer.replace ("mgmt0","")

buffer = buffer.replace ("--","")

buffer = buffer.replace ("up","")

buffer = buffer.lstrip()

buffer = buffer.split(" ")[0]

myHost = buffer

* 1. Set up Socket connection

# make a TCP socket object

# bind it to server port number

# listen, allow 5 pending connects

sockobj = socket(AF\_INET, SOCK\_STREAM)

sockobj.bind((myHost, myPort))

sockobj.listen(5)

* 1. Function to get the buffer output for the CLI command

def get\_buffer\_monitor\_output(BMcli, bufferValues):

curDate=datetime.now().strftime("%Y.%m.%d.%H.%M.%S")

BMcli.rerun()

f, m, bUsage = BMcli.get\_raw\_output().split("\n")[4].rpartition(' ')

bufferValues.append(curDate + "," + bUsage)

time.sleep(1)

return(bufferValues)

### Main Code loop

* 1. Contains the initialization of the continuous monitor command and counter

bufferValues = []

BMcli = CLI('show hardware internal buffer info pkt-stats brief', False)

counter = 0

* 1. Loop for the monitor data collection with an interior loop for checking the socket state. If the socket has received a command, the command is processed here.

# listen until process killed

while True:

curDate=datetime.now().strftime("%Y.%m.%d.%H.%M.%S")

f, m, bUsage = BMcli.get\_raw\_output().split("\n")[4].rpartition(' ')

bufferValues.append(curDate + ": " + bUsage+"\n")

time.sleep(1)

bvCount=len(bufferValues)

connectionFailed = False

curDate=datetime.now().strftime("%Y.%m.%d.%H.%M.%S")

# In case of no connection requests -

try:

connection, address = sockobj.accept()

except timeout, msg:

connectionFailed = True

pass

# Check for connection and process if data received

# read next line on client socket

if ( connectionFailed != True ):

data = connection.recv(1024)

# Send out the values

if (data == "BMdata"):

BMdata = 20

bvCount=len(bufferValues)

if (bvCount < BMdata): BMdata = bvCount

if (bvCount > 20): BMdata=20

if ( BMdata > bvCount ):

cerror = " Invalid index"

connection.send("bvCount="+repr(bvCount)+cerror)

else:

n = bvCount - BMdata

connection.send(join(bufferValues[n:bvCount]))

# get output

if (data != "BMdata"):

oCli = CLI(data, False)

buffer=repr(oCli.get\_output())

# until eof or socket closed

try: connection.send(buffer)

except: connection.close()

connection.close()

# Socket Client

## Program Structure

### Main file (NexusAccess\_m.py)

### Nexus Object file (Nexus\_Object.py)

### Supporting command files

* + - 1. script.txt
      2. mscript.txt

### Output files

* + - 1. srun.txt
      2. mrun.txt

## Main file

### Initilization

import socket

from Nexus\_Object import \*

import os

from datetime import datetime

import time

import shutil

from shutil import \*

import curses, curses.panel

HOSTS=("172.25.187.155","172.25.187.50","172.25.187.155")

HOST = '172.25.187.155' # The remote host

PORT = 50007 # The same port as used by the server

Nexus1 = Nexus\_switch(HOST,PORT)

### Sub-routines

#### stringNexusCLI

def stringNexusCLI (sbuffer='',host=HOSTS[0]):

try: buffer = repr(Nexus1.s\_socket(sbuffer,host,PORT))

except: return("Socket off-line\n ....\n")

return (buffer)

#### dsp\_output\_str

def dsp\_output\_str(bufferText):

""" Displays output \n"""

length = len (bufferText)

lenx = length - 1

# lines, columns

pad = curses.newpad(100,100)

ymax, xmax = pad.getmaxyx()

if (ymax >= 30): ymax = 30

if (xmax >= 86): xmax = 86

y=4; x=0

# 'Write a loop with lenx and

# 2500 as the increment until n\*2500 < lenx

n = 0

while ( n\*2500 < length):

mu = 2500\*(n+1)

ml = 2500\*n

try: pad.addstr(y,x, bufferText[ml:mu])

except curses.error: pass

# pad.refresh (y,x,ymin,xmin,ymax,xmax)

try: pad.refresh (0,0,1,1,ymax-2, xmax-2)

except curses.error: pass

n = n + 1

if ( n\*2500 < length ):

xinput = screen.getch()

# .... End of loop ........

return()

#### get\_mgmt0\_ip

def get\_mgmt0\_ip(mline=24,host=HOSTS[0]):

try:

inf1 = stringNexusCLI("show int mgmt0 brief | grep mgmt0", host)

except:

return ('xx.xx.xx.xx')

o = inf1

o = Nexus1.stringNexusFormat (o)

o = o.replace ("mgmt0","")

o = o.replace ("--","")

o = o.replace ("up","")

inf = o.lstrip()

inf1= inf.split(" ")[0]

buffer\_mgmt0 = "["+inf1+"] "

return(buffer\_mgmt0)

#### get\_cli\_data

def get\_cli\_data(cli\_string="", host ='0', skip=0):

data = ""

data = stringNexusCLI(cli\_string,host)

data = Nexus1.stringNexusFormat(data,skip)

dsp\_output\_str(data)

return(data)

#### get\_mcli\_data

def get\_mcli\_data( cli\_string="", host ='0'):

data = ""

data = stringNexusCLI(cli\_string,host)

data = Nexus1.stringNexusFormat(data,0)

data = data.replace ('"','\n')

data = data.replace ("\n ","")

# dsp\_output\_str(data)

return(data)

#### get\_script

def get\_script(nexusLogFile = "/bootflash/logs/buffer-nexus.logx"):

""" Runs the script file on the Nexus Chassis \n"""

bufferText="show host"

try: buffer1 = open(nexusLogFile,'r')

except: return("")

bufferText=buffer1.read()

buffer1.close()

return(bufferText

### Menu - Based on the Python “Curses” package

def get\_menu(mline=3):

COL2 = 17

hcolor = curses.color\_pair(1)

hmenu1 = curses.A\_BOLD

screen.addstr(mline, 1 , " Script ",hcolor)

screen.addstr(mline, 2 , "S", hmenu1)

screen.addstr(mline, 9 , "0", hmenu1)

screen.addstr(mline, 11 , "1", hmenu1)

screen.addstr(mline, 13 , "2", hmenu1)

screen.addstr(mline, 15 , "m", hmenu1)

screen.addstr(mline, COL2, " Queues ",hcolor)

screen.addstr(mline, COL2+1, "Q", hmenu1)

COL2 = COL2+10

screen.addstr(mline, COL2, " Log file ",hcolor)

screen.addstr(mline, COL2+1, "L", hmenu1)

COL2 = COL2+10

screen.addstr(mline, COL2, " Inter. ",hcolor)

screen.addstr(mline, COL2+1, "I" ,hmenu1)

COL2 = COL2+10

screen.addstr(mline, COL2, " Buffer ",hcolor)

screen.addstr(mline, COL2+1, "B", hmenu1)

COL2 = COL2+10

screen.addstr(mline, COL2, " Routing ",hcolor)

screen.addstr(mline, COL2+1, "R", hmenu1)

COL2 = COL2+10

screen.addstr(mline, COL2, " EXIT ",hcolor)

screen.addstr(mline, COL2+2, "X", hmenu1)

screen.addstr(mline, COL2+6, " ",hcolor)

smenu1 = "qlibrx"

screen.refresh()

return (smenu1)

def get\_menu\_data (xinput=' ', host = HOSTS[0]):

screen.move(2,16)

bufferText = ""

if xinput == ord(smenu1[0]): get\_cli\_data ('show platform software qd info global\n', host)

if xinput == ord(smenu1[1]): # Get Monitor Status

bufferText = get\_script("/bootflash/logs/buffer-nexus.logx")

bufferText = Nexus1.stringNexusFormat (bufferText)

dsp\_output\_str(bufferText)

if xinput == ord(smenu1[2]): get\_cli\_data ('show int brief\n',host)

if xinput == ord(smenu1[3]): get\_cli\_data ('BMdata',host,1)

if xinput == ord(smenu1[4]): get\_cli\_data ('sh ip route vrf management\n',host,0)

if xinput == ord('s'): # Run Script

bufferText = get\_script('/root/scripts/script.txt')

get\_cli\_data (bufferText,host,0)

if xinput == ord('0'): host = HOSTS[0]

if xinput == ord('1'): host = HOSTS[1]

if xinput == ord('2'): host = HOSTS[2]

if xinput == ord('m'): # Run multi-script

bufferText = get\_script('/root/scripts/mscript.txt')

buffer1 = HOSTS[0] + '\n '+ get\_mcli\_data (bufferText,HOSTS[0])

buffer2 = HOSTS[1] + '\n '+ get\_mcli\_data (bufferText,HOSTS[1])

buffer3 = HOSTS[2] + '\n '+ get\_mcli\_data (bufferText,HOSTS[2])

bufferm= buffer1 + buffer2 + buffer3

dsp\_output\_str(bufferm)

Nexus1.s\_write ("/root/scripts/mrun.txt", bufferm)

return(host)

### Main Loop

# Main

# .. Create Object

# .. call getNexusData

#

os.environ['TERM']='xterm-color'

screen = curses.initscr()

if (curses.has\_colors()):

curses.start\_color()

curses.init\_pair(1, curses.COLOR\_WHITE, curses.COLOR\_BLUE)

hcolor = curses.color\_pair(1)

else:

hcolor = curses.A\_REVERSE

xinput = 0

screen.clear()

host = HOSTS[0]

#buffer\_mgmt0 = get\_mgmt0\_ip(2,host) # Management IP address

while xinput != ord('x'):

# screen.clear()

# screen.border(0)

COL1 = 8

COL2 = 40

mline = 2

#buffer\_mgmt0 = get\_mgmt0\_ip(2, host) # Management IP address

title\_string = " NexusAccess-0.12m "

screen.addstr(1, 1, title\_string,curses.A\_REVERSE)

screen.addstr (1, 59 ,host, curses.A\_REVERSE)

smenu1 = get\_menu(mline) # Menu

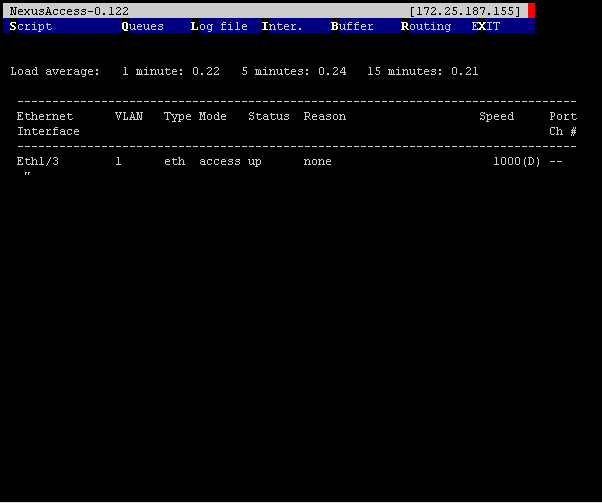
xinput = screen.getch()

host = get\_menu\_data (xinput, host)

curses.endwin()

## Menu Structure

* + 1. The menu and interface is based on the Curses library in the Python program.
    2. There are 7 options on the menu (shown below)
       1. Script - runs script.txt file
       2. Log file - reads local log file
       3. Interface - shows a summary of the N3K interfaces
       4. Buffer - reads and displays the buffer monitor data
       5. Routing - Show a summary of the current routing information
       6. Exit - Shuts down the socket interface and exits to the Linux prompt



# CLI Interface

Consists of sending the commands as strings via the socket interface and receiving the output string and stores this in a buffer.

### Script command (s)

* 1. Loads the file script.txt in to a buffer
  2. Sends the buffer to the server
  3. Reads back the data and displays the result
  4. Sample Script file:

config t

int eth1/3

no shut

show int eth 1/3 brief

### Queue command (q)

* 1. Sends the show queue command to the N3K
  2. Formats the buffer and displays the result

System info:

cos2class-id[7-0 : 0 0 0 0 0 0 0 0

cls|flag|mtuS|cosM|grpM|wred|ecn

0|0x05|1500|0xff|0x01| 1| 1

1|0x00|1500|0x00|0x00| 0| 0

2|0x00|1500|0x00|0x00| 0| 0

3|0x00|1500|0x00|0x00| 0| 0

4|0x00|1500|0x00|0x00| 0| 0

5|0x00|1500|0x00|0x00| 0| 0

6|0x00|1500|0x00|0x00| 0| 0

7|0x00|1500|0x00|0x00| 0| 0

Scheduling info

actual configured

cls|priOut|bwOut qgrp|priOut|bwOut|WREDmin|WREDmax|DropProb|Gain|

0| 0 | 100 0 | 0 | 100 | 99 | 394 | 75 | 7 |

1| 0 | 0 1 | 0 | 0 | 0 | 0 | 0 | 0 |

2| 0 | 0 2 | 0 | 0 | 0 | 0 | 0 | 0 |

3| 0 | 0 3 | 0 | 0 | 0 | 0 | 0 | 0 |

4| 0 | 0 4 | 0 | 0 | 0 | 0 | 0 | 0 |

5| 0 | 0 5 | 0 | 0 | 0 | 0 | 0 | 0 |

6| 0 | 0 6 | 0 | 0 | 0 | 0 | 0 | 0 |

7| 0 | 0 7 | 0 | 0 | 0 | 0 | 0 | 0 |

### Log file command (l)

* 1. Reads the contents of the buffer logx file
  2. Formats and displays the results.

Load average: 1 minute: 0.26 5 minutes: 0.24 15 minutes: 0.19

Processes : 352 total, 2 running

CPU states : 9.9% user, 1.5% kernel, 88.6% idle

Memory usage: 4007288K total, 1466368K used, 2540920K free

Load average: 1 minute: 0.26 5 minutes: 0.24 15 minutes: 0.19

Processes : 352 total, 2 running

CPU states : 9.9% user, 1.5% kernel, 88.6% idle

Memory usage: 4007288K total, 1466368K used, 2540920K free

### Interface command (i)

* 1. Sends the show interface command to the N3K
  2. Formats the buffer and displays the result

--------------------------------------------------------------------------------

Ethernet VLAN Type Mode Status Reason Speed Port

Interface Ch #

--------------------------------------------------------------------------------

Eth1/1 -- eth routed down Link not connected auto(D) --

Eth1/2 1 eth access down Link not connected auto(D) --

Eth1/3 1 eth access up none 1000(D) --

Eth1/4 1 eth access down Link not connected auto(D) --

Eth1/5 1 eth access up none 1000(D) --

Eth1/6 1 eth access down Link not connected auto(D) --

Eth1/7 1 eth access up none 1000(D) --

Eth1/8 1 eth access down Link not connected auto(D) --

Eth1/9 1 eth access down Link not connected auto(D) --

Eth1/10 -- eth routed down Link not connected auto(D) --

Eth1/11 1 eth access down Link not connected auto(D) --

Eth1/12 1 eth access up none 1000(D) --

Eth1/13 1 eth access down Link not connected auto(D) --

Eth1/14 1 eth access up none 1000(D) --

Eth1/15 1 eth access down Link not connected auto(D) --

Eth1/16 1 eth access up none 1000(D) --

Eth1/17 1 eth access down Link not connected auto(D) --

Eth1/18 1 eth access down Link not connected auto(D) --

Eth1/19 1 eth access down Link not connected auto(D) --

### Buffer (b)

* 1. Sends the request for the N3K buffer monitor results
  2. Formats and displays the received the output

### Routing (r)

* 1. Sends the commands for a routing summary
  2. Formats and displays the result

[ IP Route Table for VRF "management"

" \* denotes best ucast next-hop"

" \*\* denotes best mcast next-hop"

" [x/y denotes [preference/metric"

0.0.0.0/0, ubest/mbest: 1/0

\*via 172.25.187.1, mgmt0, [1/0, 1w1d, static

172.25.187.0/24, ubest/mbest: 1/0, attached

\*via 172.25.187.155, mgmt0, [0/0, 1w1d, direct

172.25.187.155/32, ubest/mbest: 1/0, attached

\*via 172.25.187.155, mgmt0, [0/0, 1w1d, local

# Multi-Chassis Features

### Menu

Consists of the following Script, 0, 1, 2 and M.

### Script

Loads /root/scripts/script.txt and then runs the script on the current chassis. Results are displayed on the screen. For example:

echo " --> Running script.txt"

echo " ... show system resources ..."

echo " ... no shut on eth 1/3 "

echo " ... sho inf brief | grep up "

echo " "

show system resources | grep Load

show system resources | grep Load

show system resources | grep Load

show system resources | grep Load

config t

int eth1/3

no shut

echo ""

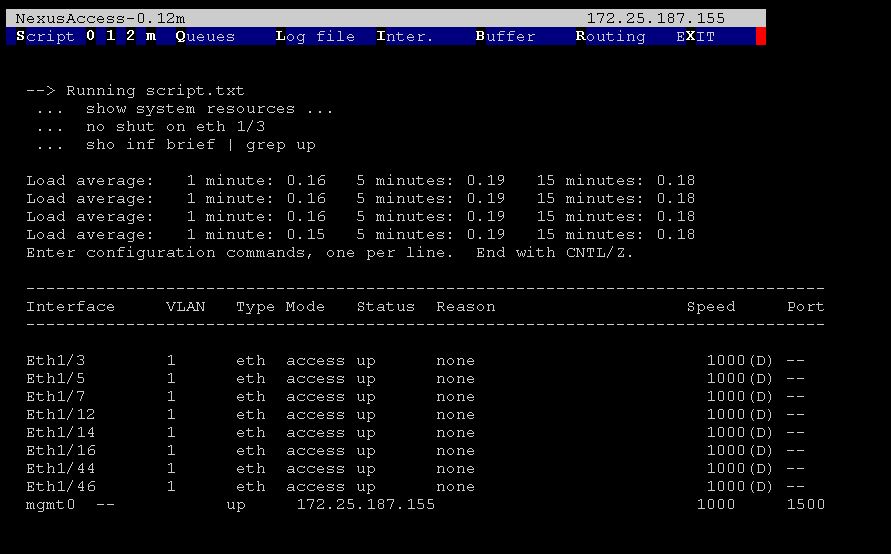
echo "--------------------------------------------------------------------------------"

echo "Interface VLAN Type Mode Status Reason Speed Port"

echo "--------------------------------------------------------------------------------"

echo ""

show int brief | grep up



### 0, 1, 2

Changes the current default chassis to IP addresses 0, 1, or 2. The display line 1 is updated with the new chassis IP address.

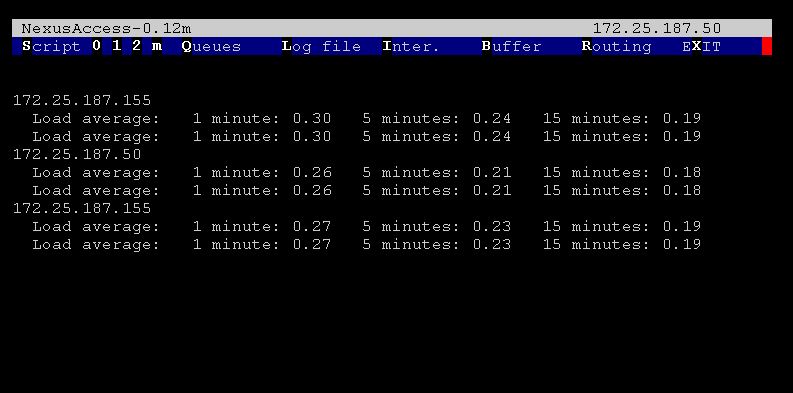
### M

Loads /root/scripts/mscript.txt and then runs the script on the three chassis listed in the code from first to last. Results are displayed on screen. mscript.txt below.

Result:

show system resources | grep Load

show system resources | grep Load



Output: mrun.txt

172.25.187.155

Load average: 1 minute: 0.50 5 minutes: 0.27 15 minutes: 0.20

Load average: 1 minute: 0.50 5 minutes: 0.27 15 minutes: 0.20

172.25.187.50

Load average: 1 minute: 0.16 5 minutes: 0.24 15 minutes: 0.16

Load average: 1 minute: 0.16 5 minutes: 0.24 15 minutes: 0.16

172.25.187.155

Load average: 1 minute: 0.50 5 minutes: 0.27 15 minutes: 0.20

Load average: 1 minute: 0.54 5 minutes: 0.28 15 minutes: 0.21

# Include files

### Nexus Object file:

# Nexus\_Object.py program

# Modified 22 Aug 2012 - Robert J. Stellman

# Initial 01 Oct 2012

#

import socket

import os

from datetime import datetime

import time

import shutil

from shutil import \*

import curses, curses.panel

class Nexus\_switch:

def \_\_init\_\_(self, host, port):

HOST = host # The remote host

PORT = port # The same port as used by the server

def s\_socket(self,sbuffer,HOST,PORT):

# Sends the buffer to the server

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

s.connect((HOST, PORT))

s.sendall(sbuffer)

data = s.recv(6500)

s.close()

return(data)

def s\_read (self,fname = "a.tmp"):

# Loads a file into the buffer string

#

nexusLogFile = fname

bufferText=" ... "

buffer1 = open(nexusLogFile,'r')

bufferText = buffer1.read()

buffer1.close()

return(bufferText)

def s\_write (self,fname= "/bootflash/a.tmp", bufferText=""):

# ... Write to a file

#

bufferFile = open(fname, 'w')

bufferFile.write(bufferText)

bufferFile.close()

return()

def stringNexusFormat (self, bufferText, skip=0):

"""\n Format file data for Humans """

bufferText = bufferText.replace ('"[','')

bufferText = bufferText.replace (']','')

bufferText = bufferText.replace ("'"," ")

bufferText = bufferText.replace ('\\',' ')

bufferText = bufferText.replace ('",','"\n') # Needed for 'show routes'

bufferText = bufferText.replace (' ,','\n')

if (skip == 1): # Buffer Monitor; csv data

bufferText = bufferText.replace ('n','\n')

return (bufferText)

# Script Files

### script.txt

echo " --> Running script.txt"

echo " ... show system resources ..."

echo " ... no shut on eth 1/3 "

echo " ... sho inf brief | grep up "

echo " "

show system resources | grep Load

config t

int eth1/3

no shut

echo ""

echo "--------------------------------------------------------------------------------"

echo "Interface VLAN Type Mode Status Reason Speed Port"

echo "--------------------------------------------------------------------------------"

echo ""

show int brief | grep up

### mscript.txt

show system resources | grep Load

show system resources | grep Load

### mrun.txt (mscript.txt output)

172.25.187.155

Load average: 1 minute: 0.50 5 minutes: 0.27 15 minutes: 0.20

Load average: 1 minute: 0.50 5 minutes: 0.27 15 minutes: 0.20

172.25.187.50

Load average: 1 minute: 0.16 5 minutes: 0.24 15 minutes: 0.16

Load average: 1 minute: 0.16 5 minutes: 0.24 15 minutes: 0.16

172.25.187.155

Load average: 1 minute: 0.50 5 minutes: 0.27 15 minutes: 0.20

Load average: 1 minute: 0.54 5 minutes: 0.28 15 minutes: 0.21