**Netmiko 3: Sending Configuration Commands with send\_config\_set() and Saving your Changes**

**Objectives - Part 1: Using a List of Configuration Commands**

* Use IOS configuration commands from a Python list to configure a router

**Objectives - Part 2: Saving your configs**

* Use the **save\_config** ConnectionHandler method to copy running-config to startup-config

**Objectives - Part 3: Using a File with Configuration Commands**

* Use IOS configuration commands from a file to configure a router

**Extra**

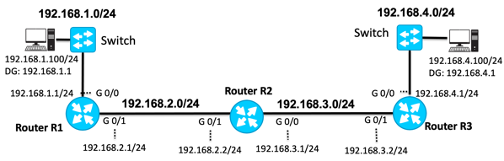
* **Part 4: Sending Commands from multiple files**
* **Part 5: Sending Commands from multiple EEM file**

**Requirements**

* Python: You should be familiar with the basics of Python.
* Using the topology below or your own topology, have access to one or more Cisco IOS devices using SSH.

**Topology**

This lab uses the following topology. You can use a different topology, just be sure to make any necessary modifications to interfaces and addresses. See Netmiko Lab 0: Using a Script for Initial Setup



**Part 1: Using a List of Configuration Commands**

**Objectives - Part 1**

* Use IOS configuration commands from a Python list to configure a router

**Step 1: Open a new Python file**

Using your Python IDE or equivalent, create a new Python file using the code below.

Note: You may wish to start with this template from your previous Python program.

Note: You can always disregard the **try-except** commands. Just be sure to decrease the indentation for the rest of the commands in the **for** loop.

import netmiko

print('\n')

for device in devices:

try:

connection = netmiko.ConnectHandler(ip=device,

device\_type='cisco\_ios',

username='admin',

password='cisco')

connection.disconnect()

except netmiko.exceptions.NetmikoTimeoutException:

print('Timeout occurred to', device)

print('''Common causes of this problem are:

1. Incorrect hostname or IP address.

2. Wrong TCP port.

3. Intermediate firewall blocking access.''')

print('\n')

except netmiko.exceptions.NetMikoAuthenticationException:

print('Authentication error', device)

print('''Common causes of this problem are:

1. Invalid username and password

2. Incorrect SSH-key file

3. Connecting to the wrong device''')

print('\n')

except netmiko.exceptions.netmiko.exceptions.ReadTimeout:

print('Read timeout. pattern not detected', device)

print('''Common causes of this problem are:

1. Missing or incorrect secret password in ConnectHandler()

2. Adjust the regex pattern to better identify the terminating

string. Note, in many situations the pattern is

automatically based on the network device's prompt.

3. Increase the read\_timeout to a larger value.''')

print('\n')

except Exception as e:

print("An error occurred:", str(e))

**Step 2: Create a dictionary of IP addresses**

In this case our list will only consist of a single device. This is because our configuration commands will be specific to this one device. Of course, we would not need a list and the **for** loop that uses this list, but for consistency and adaptability, we will use a for loop with a list of one item.

After the import netmiko command, add:

**devices = ['192.168.1.1’]**

**Step 3: Create a list of strings, configuration commands**

Following the devices list, create a list of strings, with each string being a separate IOS command. Later, we will send this list of configuration commands to our router, device, at 192.168.1.1.

**ipv6\_interface\_list = [**

**'interface g0/0',**

**'ipv6 address 2001:db8:cafe:1::1/64',**

**'ipv6 address fe80::1:1 link-local',**

**‘exit’,**

**'interface g0/1',**

**'ipv6 address 2001:db8:cafe:2::1/64',**

**'ipv6 address fe80::1:2 link-local'**

**]**

**Step 4: Establish an SSH connection - Connect.Handler()**

Within the **for** loop, edit the **Connect.Handler()**. We have included the privileged password **class** since we need privilege level access to make configuration changes.

Be sure this is indented within our **for** loop.

**connection = netmiko.ConnectHandler(ip=device,**

**device\_type='cisco\_ios',**

**username='admin',**

**password='cisco',**

**secret = 'class'**

**)**

**Step 5: Use the enable() method for privileged mode access**

Since we will be sending a list of commands to make changes to the device we will need privileged mode access using the **enable()** method. This command is also with our **for** toop, following the ConnectHandler.

**connection.enable()**

It is important to note that not all configuration methods require the **enable()** method for access. As we will see later, some methods have that capability embedded in the command and may give you an error if you include the **enable()** method.

**Step 6: Use the send\_config\_set() method for sending configuration information**

The **send\_config\_set()** method accepts the configuration commands as a string or a list of strings. A list of strings is more common.

Use our **connection** object with the **send\_config\_set()** method, sending our list, ipvt\_interface\_list, as the argument. Again, be sure all of this is within the **for** loop.

**connection.send\_config\_set(ipv6\_interface\_list)**

**Step 7: Send an IOS command - send\_command()**

Within the same for loop, will will do the following:

* Display the value of **device**, the current IPv4 address of the device we used for SSH
* Use the **send\_command** method to verify our IPv6 addressing information. Again, the **print** option is helpful otherwise you get \n instead of line breaks.

**print(device)**

**print('-'\*11)**

**print(connection.send\_command('show ipv6 interface brief'))**

**Step 8: Disconnect from Netmiko - disconnect()**

This command should be the last statement in the **try** section of your **for** loop (before the first **except**). Please add it if it is not already there.

**connection.disconnect()**

**Step 9: Display the program**

Here is the complete program including comments:

import netmiko

devices = ['192.168.1.1']

ipv6\_interface\_list = [

'ipv6 unicast-routing',

'interface g0/0',

'ipv6 address 2001:db8:cafe:1::1/64',

'ipv6 address fe80::1:1 link-local',

'exit',

'interface g0/1',

'ipv6 address 2001:db8:cafe:2::1/64',

'ipv6 address fe80::1:2 link-local',

'exit'

]

for device in devices:

try:

connection = netmiko.ConnectHandler(ip=device,

device\_type='cisco\_ios',

username='admin',

password='cisco',

secret = 'class'

)

connection.enable()

connection.send\_config\_set(ipv6\_interface\_list)

print(device)

print('-'\*11)

print(connection.send\_command('show ipv6 interface brief'))

print('\n')

connection.disconnect()

except netmiko.exceptions.NetmikoTimeoutException:

print('Timeout occurred to', device)

print('''Common causes of this problem are:

1. Incorrect hostname or IP address.

2. Wrong TCP port.

3. Intermediate firewall blocking access.''')

print('\n')

except netmiko.exceptions.NetMikoAuthenticationException:

print('Authentication error', device)

print('''Common causes of this problem are:

1. Invalid username and password

2. Incorrect SSH-key file

3. Connecting to the wrong device''')

print('\n')

except netmiko.exceptions.ReadTimeout:

print('Read timeout. pattern not detected', device)

print('''Common causes of this problem are:

1. Missing or incorrect secret password in ConnectHandler()

2. Adjust the regex pattern to better identify the terminating

string. Note, in many situations the pattern is

automatically based on the network device's prompt.

3. Increase the read\_timeout to a larger value.''')

print('\n')

except Exception as e:

print("An error occurred:", str(e))

*Rick's program: netmiko\_3a.py*

Note: This program will be used as a starting point for the next part.

**Step 10: Run the program**

192.168.1.1

-----------

Em0/0 [administratively down/down]

unassigned

GigabitEthernet0/0 [up/up]

unassigned

GigabitEthernet0/1 [up/up]

FE80::1:2

2001:DB8:CAFE:2::1

Serial0/0/0 [administratively down/down]

unassigned

Serial0/0/1 [administratively down/down]

unassigned

**Part 2: Using save\_config to Save Configurations**

**Objectives - Part 2**

* Use the **save\_config** ConnectionHandler method to copy running-config to startup-config

When working with several devices at the same time it can be helpful to save all the running-configs at the same time (copy running-config startup-config).

**Step 1: Remove the current previous configuration and show commands**

Delete or comment the following statements in your program. If you wish to do the next part, just comment out these statements.

connection.send\_config\_set(ipv6\_interface\_list)

print(connection.send\_command('show ipv6 interface brief'))

Delete the list: ipv6\_interface\_list

ipv6\_interface\_list = [

'interface g0/0',

'ipv6 address 2001:db8:cafe:1::1/64',

'ipv6 address fe80::1:1 link-local',

'exit',

'interface g0/1',

'ipv6 address 2001:db8:cafe:2::1/64',

'ipv6 address fe80::1:2 link-local'

]

**Step 2: Modify our devices list**

Prior to the for loop, modify our devices list to include all of our routers.

With multiple IP addresses in the **devices** list, you can perform 'copy running-config startup-config' to multiple devices at the same time!

**devices = ['192.168.1.1', '192.168.2.2', '192.168.3.2']**

**Step 3: Add the save\_config() method**

At the previous location of our connection.send\_config\_set() command, add the following:

**output = connection.save\_config**

**print(output)**

* The first statement will copy the running-config to the startup-config and save the Netmiko output to the variable **output**.
* The second statement displays the content of **output** so we can see if the **save\_config** was successful or not. This is similar to using **print(connection.save\_config).**

**Step 4: Display the program**

Here is the complete program including comments:

import netmiko

devices = ['192.168.1.1', '192.168.2.2', '192.168.3.2']

print('\n')

for device in devices:

try:

connection = netmiko.ConnectHandler(ip=device,

device\_type='cisco\_ios',

username='admin',

password='cisco')

print(device)

print('-'\*11)

output = connection.save\_config

print(output)

print('\n')

connection.disconnect()

except netmiko.exceptions.NetmikoTimeoutException:

print('Timeout occurred to', device)

print('''Common causes of this problem are:

1. Incorrect hostname or IP address.

2. Wrong TCP port.

3. Intermediate firewall blocking access.''')

print('\n')

except netmiko.exceptions.NetMikoAuthenticationException:

print('Authentication error', device)

print('''Common causes of this problem are:

1. Invalid username and password

2. Incorrect SSH-key file

3. Connecting to the wrong device''')

print('\n')

except netmiko.exceptions.ReadTimeout:

print('Read timeout. pattern not detected', device)

print('''Common causes of this problem are:

1. Missing or incorrect secret password in ConnectHandler()

2. Adjust the regex pattern to better identify the terminating

string. Note, in many situations the pattern is

automatically based on the network device's prompt.

3. Increase the read\_timeout to a larger value.''')

print('\n')

except Exception as e:

print("An error occurred:", str(e))

*Rick's program: netmiko\_3b.py*

**Step 4: Run the program**

Run the program:

192.168.1.1

-----------

<bound method CiscoIosBase.save\_config of <netmiko.cisco.cisco\_ios.CiscoIosSSH object at 0x10b1abf40>>

192.168.2.2

-----------

<bound method CiscoIosBase.save\_config of <netmiko.cisco.cisco\_ios.CiscoIosSSH object at 0x10b1abf10>>

192.168.3.2

-----------

<bound method CiscoIosBase.save\_config of <netmiko.cisco.cisco\_ios.CiscoIosSSH object at 0x10caef220>>

**Part 3: Sending Commands from a file**

**Objectives**

* Use IOS configuration commands from a file to configure a router

If you are familiar with how Python reads files, you can try this one.

**Step 1: Create a text file**

Create a text file with the following information. Save the file as **ipv6commands-R2.txt**

interface g0/0

ipv6 address 2001:db8:cafe:3::1/64

ipv6 address fe80::3:1 link-local

exit

interface g0/1

ipv6 address 2001:db8:cafe:2::2/64

ipv6 address fe80::2:2 link-local

exit

**Step 2: Continue with the previous code and make the following changes**

Change our device list to use the ipv4 address of R2, 192.168.2.2

**devices = ['192.168.2.2']**

Delete the list: ipv6\_interface\_list

ipv6\_interface\_list = [

'interface g0/0',

'ipv6 address 2001:db8:cafe:1::1/64',

'ipv6 address fe80::1:1 link-local',

'exit',

'interface g0/1',

'ipv6 address 2001:db8:cafe:2::1/64',

'ipv6 address fe80::1:2 link-local'

]

**Step 3: Read the commands from the file**

Within the **for** loop and right after the ConnectHandler() , **connection = netmiko.ConnectHandler(...)**, add the following:

**with open('ipv6commands-R2.txt', 'r') as file:**

**commands = file.read().splitlines()**

This command will read contents from our file **ipv6commands-R2.txt**

Here's a step-by-step explanation of what the code does:

1. The **open()** function is used to open the file **'ipv6commands-R2.txt'**. The file mode **'r'** specifies that the file should be opened in read-only mode.
2. The **with** statement is used to ensure that the file is properly closed after it's been read. It provides a context manager that automatically takes care of opening and closing the file.
3. Inside the **with** block, the **file.read()** method is called to read the entire contents of the file as a string.
4. The **splitlines()** method is called on the resulting string to split it into a list of lines. Each line of the file becomes an element in the resulting list.
5. The resulting list of lines is assigned to the variable **commands**.

After executing this code, the **commands** variable will contain a list of strings, where each string represents a line from the **'ipv6commands-R2.txt'** file.

Note: Make sure that the file **'ipv6commands-R2.txt'** exists in the same directory as the Python script or notebook where this code is being executed.

Note: You might be required to add a path to the beginning of your filename.

**Step 4: Use the send\_config\_set method using the file name**

After the with open commands, use the **send\_config\_set()** method to send the contents of the file. Notice that we have replaced the name of the list with the variable **commands**.

**connection.send\_config\_set(commands)**

Re-enter the send\_command() to validate the change. th

**print(connection.send\_command('show ipv6 interface brief'))**

This rest of our program remains the same.

**Step 5: Display the program**

import netmiko

devices = ['192.168.2.2']

for device in devices:

try:

connection = netmiko.ConnectHandler(ip=device,

device\_type='cisco\_ios',

username='admin',

password='cisco',

secret = 'class'

)

with open('ipv6commands\_R2.txt', 'r') as file:

commands = file.read().splitlines()

connection.enable()

connection.send\_config\_set(commands)

print(device)

print('-'\*11)

print(connection.send\_command('show ipv6 interface brief'))

print('\n')

connection.disconnect()

except netmiko.exceptions.NetmikoTimeoutException:

print('Timeout occurred to', device)

print('''Common causes of this problem are:

1. Incorrect hostname or IP address.

2. Wrong TCP port.

3. Intermediate firewall blocking access.''')

print('\n')

except netmiko.exceptions.NetMikoAuthenticationException:

print('Authentication error', device)

print('''Common causes of this problem are:

1. Invalid username and password

2. Incorrect SSH-key file

3. Connecting to the wrong device''')

print('\n')

except netmiko.exceptions.ReadTimeout:

print('Read timeout. pattern not detected', device)

print('''Common causes of this problem are:

1. Missing or incorrect secret password in ConnectHandler()

2. Adjust the regex pattern to better identify the terminating

string. Note, in many situations the pattern is

automatically based on the network device's prompt.

3. Increase the read\_timeout to a larger value.''')

print('\n')

except Exception as e:

print("An error occurred:", str(e))

*Rick's program: netmiko\_3c.py*

**File:** **ipv6commands-R2.txt**

interface g0/0

ipv6 address 2001:db8:cafe:3::1/64

ipv6 address fe80::3:1 link-local

exit

interface g0/1

ipv6 address 2001:db8:cafe:2::2/64

ipv6 address fe80::2:2 link-local

exit

*Rick's file: ipv6commands-R2.txt*

**Step 6: Run the program**

192.168.2.2

-----------

Em0/0 [administratively down/down]

unassigned

GigabitEthernet0/0 [up/up]

FE80::3:1

2001:DB8:CAFE:3::1

GigabitEthernet0/1 [up/up]

FE80::2:2

2001:DB8:CAFE:2::2

Serial0/0/0 [administratively down/down]

unassigned

Serial0/0/1 [administratively down/down]

unassigned

**Part 4: Sending Commands from multiple files**

EXTRA

**ipv6commands\_R1.txt**

ipv6 unicast-routing

ipv6 unicast-routing

interface g0/0

ipv6 address 2001:db8:cafe:1::1/64

ipv6 address fe80::1:1 link-local

exit

interface g0/1

ipv6 address 2001:db8:cafe:2::1/64

ipv6 address fe80::1:2 link-local

exit

ipv6 route ::/0 2001:db8:cafe:2::2

*Rick's file: ipv6commands-R1.txt*

**ipv6commands\_R2.txt**

ipv6 unicast-routing

interface g0/0

ipv6 address 2001:db8:cafe:3::1/64

ipv6 address fe80::3:1 link-local

exit

interface g0/1

ipv6 address 2001:db8:cafe:2::2/64

ipv6 address fe80::2:2 link-local

exit

ipv6 route 2001:db8:cafe:1::/64 2001:db8:cafe:2::1

ipv6 route 2001:db8:cafe:4::/64 2001:db8:cafe:3::2

*Rick's file: ipv6commands-R2.txt*

**ipv6commands\_R3.txt**

ipv6 unicast-routing

interface g0/0

ipv6 address 2001:db8:cafe:4::1/64

ipv6 address fe80::4:1 link-local

exit

interface g0/1

ipv6 address 2001:db8:cafe:3::2/64

ipv6 address fe80::3:2 link-local

exit

ipv6 route ::/0 2001:db8:cafe:3::1

*Rick's file: ipv6commands-R3.txt*

import netmiko

devices = [

{'ip': '192.168.1.1', 'name':'r1'},

{'ip': '192.168.2.2', 'name':'r2'},

{'ip': '192.168.3.1', 'name':'r3'}

]

for device in devices:

try:

connection = netmiko.ConnectHandler(ip=device['ip'],

device\_type='cisco\_ios',

username='admin',

password='cisco',

secret = 'class'

)

**path = '/Users/...'**

**filename = 'ipv6commands\_'**

**filename = path + filename + device['name'] + '.txt'**

with open(filename, 'r') as file:

commands = file.read().splitlines()

connection.enable()

connection.send\_config\_set(commands)

print(device['ip'])

print('-'\*11)

print(connection.send\_command('show ipv6 interface brief'))

print('\n')

connection.disconnect()

except netmiko.exceptions.NetmikoTimeoutException:

print('Timeout occurred to', device['ip'])

print('''Common causes of this problem are:

1. Incorrect hostname or IP address.

2. Wrong TCP port.

3. Intermediate firewall blocking access.''')

print('\n')

except netmiko.exceptions.NetMikoAuthenticationException:

print('Authentication error', device['ip'])

print('''Common causes of this problem are:

1. Invalid username and password

2. Incorrect SSH-key file

3. Connecting to the wrong device''')

print('\n')

except netmiko.exceptions.ReadTimeout:

print('Read timeout. pattern not detected', device['ip'])

print('''Common causes of this problem are:

1. Missing or incorrect secret password in ConnectHandler()

2. Adjust the regex pattern to better identify the terminating

string. Note, in many situations the pattern is

automatically based on the network device's prompt.

3. Increase the read\_timeout to a larger value.''')

print('\n')

except Exception as e:

print("An error occurred:", str(e))

*Rick's program: netmiko\_3e.py*

**Part 5: Sending Commands from multiple EEM file**

EXTRA

EEM - Change static routes with an AD of 150, so static routes stay until ospf routes with AD of 110 are installed in routing table. Then enable OSPF. Little or no packet loss.

**ospf\_eem\_r1.txt**

event manager applet configure-ospf-once

event none sync yes

action 1.0 cli command "enable"

action 2.0 cli command "configure terminal"

action 3.1 cli command "ip route 192.168.3.0 255.255.255.0 192.168.2.2 150"

action 3.2 cli command "ip route 192.168.4.0 255.255.255.0 192.168.2.2 150"

action 4.0 cli command "router ospf 1"

action 5.0 cli command "router-id 1.1.1.1"

action 6.0 cli command "network 192.168.0.0 0.0.255.255 area 0"

action 7.0 cli command "exit"

action 8.0 cli command "no event manager applet configure-ospf-once"

event timer countdown time 120

**ospf\_eem\_r2.txt**

event manager applet configure-ospf-once

event none sync yes

action 1.0 cli command "enable"

action 2.0 cli command "configure terminal"

action 3.1 cli command "ip route 192.168.1.0 255.255.255.0 192.168.2.1 150"

action 4.0 cli command "router ospf 1"

action 5.0 cli command "router-id 2.2.2.2"

action 6.0 cli command "network 192.168.0.0 0.0.255.255 area 0"

action 7.0 cli command "exit"

action 8.0 cli command "no event manager applet configure-ospf-once"

event timer countdown time 120

**ospf\_eem\_r3.txt**

event manager applet configure-ospf-once

event none sync yes

action 1.0 cli command "enable"

action 2.0 cli command "configure terminal"

action 3.1 cli command "ip route 192.168.1.0 255.255.255.0 192.168.2.2 150"

action 3.2 cli command "ip route 192.168.2.0 255.255.255.0 192.168.2.2 150"

action 4.0 cli command "router ospf 1"

action 5.0 cli command "router-id 3.3.3.3"

action 6.0 cli command "network 192.168.0.0 0.0.255.255 area 0"

action 7.0 cli command "exit"

action 8.0 cli command "no event manager applet configure-ospf-once"

event timer countdown time 120

**Code**

'''

EEM:

- Do not need SSH access once it runs.

- Can use the event timer to delay it, and sync with EEMs on other devices.

Option 1: EEM - Change static routes with an AD of 150, so static routes stay until ospf routes with AD of 110 are installed in routing table. Then enable OSPF. Little or no packet loss.

Option 2: include 'no ip route' statements in the EEM files but this causes delay. Some packet loss.

Option 3: Create separate EEM file with delay (or python sleep()) that has 'no ip route' to give time for ospf to converge. Some packet loss.

'''

import netmiko

# List of devices; 'ip' for SSH and 'name' for filename

devices = [

{'ip': '192.168.1.1', 'name':'r1'},

{'ip': '192.168.2.2', 'name':'r2'},

{'ip': '192.168.3.2', 'name':'r3'}

]

for device in devices:

try:

connection = netmiko.ConnectHandler(ip=device['ip'],

device\_type='cisco\_ios',

username='admin',

password='cisco',

secret = 'class'

)

################################################################

# EEM Commands (OSPF)

################################################################

path = '/Users/... '

filename = 'ospf\_eem\_'

eem\_filename = path + filename + device['name'] + '.txt'

print(filename)

# Read commands from a file

with open(eem\_filename, 'r') as file:

eem\_commands = file.read().splitlines()

connection.enable()

connection.send\_config\_set(eem\_commands)

**connection.send\_command('show event manager policy registered ')**

**connection.send\_command('event manager run configure-ospf-once')**

print(device['ip'])

print('-'\*11)

print(connection.send\_command('show running-config | section router ospf'))

print('\n')

connection.disconnect()

except netmiko.exceptions.NetmikoTimeoutException:

print('Timeout occurred to', device['ip'])

print('''Common causes of this problem are:

1. Incorrect hostname or IP address.

2. Wrong TCP port.

3. Intermediate firewall blocking access.''')

print('\n')

except netmiko.exceptions.NetMikoAuthenticationException:

print('Authentication error', device['ip'])

print('''Common causes of this problem are:

1. Invalid username and password

2. Incorrect SSH-key file

3. Connecting to the wrong device''')

print('\n')

except netmiko.exceptions.ReadTimeout:

print('Read timeout. pattern not detected', device['ip'])

print('''Common causes of this problem are:

1. Missing or incorrect secret password in ConnectHandler()

2. Adjust the regex pattern to better identify the terminating

string. Note, in many situations the pattern is

automatically based on the network device's prompt.

3. Increase the read\_timeout to a larger value.''')

print('\n')

except Exception as e:

print("An error occurred:", str(e))

*Rick's program: netmiko\_3f.py*

Congratulations! Now move on to the next lab…