**Netmiko 4: Getting Devices from a Dictionary and from a JSON file**

What if you had devices with different device\_types, usernames, or passwords? We can use a dictionary to store individual device information.

**Objectives - Part 1: Using a Dictionary of Devices**

* Use a list of devices, where each device is a dictionary
* Use send\_command() method to show current time of each device

**Objectives - Part 1: Using a JSON file of devices**

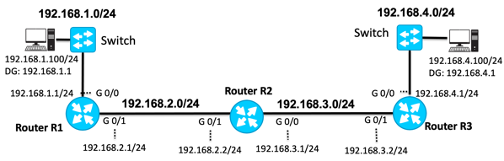
* Use a JSON file of devices
* Use send\_command() method to show current time of each device

**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 Dictionary of Devices**

**Objectives**

* Use a list of devices, where each device is a dictionary
* Use send\_command() method to show current time of each device

**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.

The ConnectHandler() is included but incomplete.

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()

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))

Note: Be sure the **except** statements refer to the key **ip** to get just the IPv4 address of the device we connected to.

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

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

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

**Step 3: Create a dictionary item for each device**

Following the **import netmiko** command, create a variable for each device. Each device is a dictionary with the appropriate credentials for SSH and the ConnectHandler.

For example, **r1** is a variable that holds a dictionary object that can be accessed using its variable name. You can perform dictionary operations on **r1**, such as accessing specific values using keys, adding or modifying key-value pairs, or iterating over its elements.

**r1 = {**

**'ip': '192.168.1.1',**

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

**'username': 'admin',**

**'password': 'cisco'**

**}**

**r2 = {**

**'ip': '192.168.2.2',**

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

**'username': 'admin',**

**'password': 'cisco'**

**}**

**r3 = {**

**'ip': '192.168.3.2',**

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

**'username': 'admin',**

**'password': 'cisco'**

**}**

**s1 = {**

**'ip': '192.168.1.5',**

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

**'username': 'admin',**

**'password': 'cisco'**

**}**

**s3 = {**

**'ip': '192.168.4.5',**

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

**'username': 'admin',**

**'password': 'cisco'**

**}**

Note: Switches S1 and S2 were not configured. Our exception handling code will catch this issue.

**Step 4: Create a dictionary of IP addresses**

Following the dictionaries, create a list of variables you used above. Each element of the list is a variable referencing a dictionary item. The list of devices contains multiple variables, and each variable represents a dictionary.

**devices = [r1, r2, r3, s1 ,s3]**

Note: Later, we will see how to send different sets of commands for various devices.

**Step 5: Create a for loop referring using our list of devices and include the exception handling**

As we did in the previous program, create a for loop that accesses the list of devices. Add the try and except commands for exception handling.

**for device in devices:**

**try:**

**Step 6: Establish an SSH connection - Connect.Handler()**

Within the **for** loop, edit the the **Connect.Handler()**, to establish the SSH connections to each device. Notice we have removed all the arguments in the method. Now, add a single argument which is a single variable prepended with two asterisks, **\*\* device**.

Remember, **device** is a variable of type dictionary. The **\*\*** is used for unpacking the **device** dictionary as a keyword argument. The **\*\*** notation allows you to pass the key-value pairs from the **device** dictionary as separate keyword arguments to the function.

**connection = netmiko.ConnectHandler(\*\*device)**

**Step 7: Display the device IP and current time**

As you have done previously, display the device’s IPv4 address and save the config. You must change **device** to **device[ip']** to print out the value associated with the key **ip**. Otherwise, **device** by itself displays all the information in the dictionary for each device.

Use the **send\_command** method to display the current time on the device.

**print(device['ip'])**

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

**output = connection.send\_command('show clock')**

**print(output))**

**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**

import netmiko

r1 = {

'ip': '192.168.1.1',

'device\_type': 'cisco\_ios',

'username': 'admin',

'password': 'cisco'

}

r2 = {

'ip': '192.168.2.2',

'device\_type': 'cisco\_ios',

'username': 'admin',

'password': 'cisco'

}

r3 = {

'ip': '192.168.3.2',

'device\_type': 'cisco\_ios',

'username': 'admin',

'password': 'cisco'

}

s1 = {

'ip': '192.168.1.5',

'device\_type': 'cisco\_ios',

'username': 'admin',

'password': 'cisco'

}

s3 = {

'ip': '192.168.4.5',

'device\_type': 'cisco\_ios',

'username': 'admin',

'password': 'cisco'

}

devices = [r1, r2, r3, s1 ,s3]

print('\n')

for device in devices:

try:

connection = netmiko.ConnectHandler(\*\*device)

print(device['ip'])

print('-'\*11)

output = connection.send\_command('show clock')

print(output)

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))

**Step 10: Run the program**

192.168.1.1

-----------

\*14:18:38.883 UTC Fri Jun 23 2023

192.168.2.2

-----------

\*11:40:00.947 UTC Fri Jun 23 2023

192.168.3.2

-----------

\*19:55:51.035 UTC Fri Jun 23 2023

Timeout occurred to 192.168.1.5

Common causes of this problem are:

1. Incorrect hostname or IP address.

2. Wrong TCP port.

3. Intermediate firewall blocking access.

Timeout occurred to 192.168.4.5

Common causes of this problem are:

1. Incorrect hostname or IP address.

2. Wrong TCP port.

3. Intermediate firewall blocking access.

Notice the two error conditions that occurred that were caught by our exceptions. This is because we have not configured the switches for remote access.

**Part 2: Using a JSON file of Devices (Optional)**

**Objectives**

* Use a JSON file of devices
* Use send\_command() method to show current time of each device

If you are familiar with JSON, you can try this one.

**Step 1: Create a JSON text file**

JavaScript Object Notation (JSON) is a standardized format commonly used to transfer data as text that can be sent over a network. JSON represents objects as key/value pairs, just like a Python dictionary.

Create a text file with the following information. Save the file as **my\_devices.json.**

You will notice that our JSON file looks almost identical to our Python dictionary, without the variables. You will also notice that the JSON file is a list of dictionaries, the same dictionaries we used in our previous Python program. So yes, you can copy most of the information from the previous Python program and just make some edits!

[

{

"ip": "192.168.1.1",

"device\_type": "cisco\_ios",

"username": "admin",

"password": "cisco"

},

{

"ip": "192.168.2.2",

"device\_type": "cisco\_ios",

"username": "admin",

"password": "cisco"

},

{

"ip": "192.168.3.2",

"device\_type": "cisco\_ios",

"username": "admin",

"password": "cisco"

},

{

"ip": "192.168.1.5",

"device\_type": "cisco\_ios",

"username": "admin",

"password": "cisco"

},

{

"ip": "192.168.4.5",

"device\_type": "cisco\_ios",

"username": "admin",

"password": "cisco"

}

]

**Step 2: Continue with the previous file and make the following deletions**

Remove the dictionary items:

r1 = {

'ip': '192.168.1.1',

'device\_type': 'cisco\_ios',

'username': 'admin',

'password': 'cisco'

}

r2 = {

'ip': '192.168.2.2',

'device\_type': 'cisco\_ios',

'username': 'admin',

'password': 'cisco'

}

etc.

Remove the list of devices:

devices = [r1, r2, r3, s1 ,s3]

**Step 3: Import JSON**

Following **import netmiko**, add the JSON library.

**import json**

Following the **import json,** add the statements below. These statements open the JSON file, which is a list of dictionaries, and assign the values of the dictionary to variable **devices**.

**Step 4: Open the JSON file**

Add the following statement:

**with open('my\_devices.json') as device\_file:**

**devices = json.load(device\_file)**

Here's a breakdown of what the code does:

* The **open()** function is used to open the "my\_devices.json" file in read mode. This function returns a file object that represents the opened file.
* The **with** statement is used to ensure proper handling of the file. It automatically takes care of closing the file after its block of code is executed.
* Inside the with block, the **json.load()** function is used to load the contents of the file into a Python data structure. This function takes the file object returned by **open()** as an argument and parses the JSON data.
* The parsed data is then assigned to the variable **devices**.
* After executing this code, the **devices** variable will hold the contents of the "my\_devices.json" file, which is expected to be a list of dictionaries, with each dictionary representing a different device.

Note: No changes to the **for** loop.

**Bonus coverage on JSON**

A JSON file is not inherently considered a string. Rather, a JSON file contains text data that conforms to the JSON syntax rules and is typically stored as plain text in a file. When you read the contents of a JSON file using file I/O operations in Python, you receive the content as a string.

Here's a breakdown of the process:

1. JSON Format: JSON (JavaScript Object Notation) is a data interchange format that uses a text-based structure to represent data objects. It consists of key-value pairs, arrays (lists), objects (dictionaries), strings, numbers, booleans, and null values. JSON files follow the syntax and rules defined by the JSON format.
2. File Storage: JSON files are stored as plain text files, typically with a .json file extension. They contain the JSON-formatted data as a text string, following the JSON syntax. The JSON content is stored as characters, including whitespace and newline characters, within the file.
3. Reading JSON File: When you read the contents of a JSON file using file I/O operations in Python (such as open() and read()), the file's text content is read and returned as a string. The string represents the exact characters stored in the JSON file, including the JSON syntax and any whitespace.
4. Parsing JSON: To work with the JSON data in Python, you can use the json module. This module provides functions like json.load() or json.loads() to parse the JSON string and convert it into appropriate Python data structures (e.g., dictionaries, lists, strings, numbers) that you can work with programmatically.

So, while a JSON file itself is not considered a string, the contents of the JSON file are read as a string when you retrieve them using file I/O operations. The string representation allows you to manipulate and process the JSON data within your Python code before converting it into Python data structures for further analysis or usage.

**Step 4: Display the program**

import netmiko

import json

with open('my\_devices.json') as device\_file:

devices = json.load(device\_file)

print('\n')

for device in devices:

try:

connection = netmiko.ConnectHandler(\*\*device)

print(device['ip'])

print('-'\*11)

output = connection.send\_command('show clock')

print(output)

print('\n')

connection.disconnect()

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')

# Catch-all exception handler for any other exception

except Exception as e:

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

**Step 5: Run the program**

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

Connecting to device 192.168.1.1

\*15:06:58.091 UTC Fri Jun 23 2023

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

Connecting to device 192.168.2.2

\*12:28:20.211 UTC Fri Jun 23 2023

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

Connecting to device 192.168.3.2

\*20:44:10.287 UTC Fri Jun 23 2023

Timeout occurred to 192.168.1.5

Common causes of this problem are:

1. Incorrect hostname or IP address.

2. Wrong TCP port.

3. Intermediate firewall blocking access.

Timeout occurred to 192.168.4.5

Common causes of this problem are:

1. Incorrect hostname or IP address.

2. Wrong TCP port.

3. Intermediate firewall blocking access.