2022

2.5 Lab – RESTCONF with Python

Programación de Redes

UNIVERSIDAD TECNLOGICA DEL NORTE DE GUANAJUATO

NOMBRE

JUAN PABLO PALMA APODERADO

CARRERA

TSU EN INFRAESTRUCTURA DE REDES DIGITALES

DOCENTE

BARRON RODRIGUEZ GABRIEL

NO. CONTROL

1221100259

FECHA

14 DE OCTUBRE DEL 2022

GRUPO

GIR0441

Lab – RESTCONF with Python

Objectives

Part 1: RESTCONF basics in Python

Part 2: Modify interface configuration with RESTCONF in Python

Background / Scenario

Following up the previous lab activity, in this lab you will learn how to execute the RESTCONF API calls using Python scripts.

Required Resources

- Python 3.x environment
- Access to a router with the IOS XE operating system version 16.6 or higher.

Instructions

Part 1: RESTCONF in Python

In this part, you will use Python to request a RESTCONF API.

Step 1: Import modules and disable SSL warnings.

a. In IDLE, click **File > New File** to open IDLE Editor.

b. Save the file as **lab 2.5.py**.

c. Enter the following commands to import the modules and disable SSL certificate warnings:

```
import json
import requests
requests.packages.urllib3.disable_warnings()
```

The **json** module includes methods convert JSON data to Python objects and vice versa. The **requests** module has methods that will let us send REST requests to a URI.

Step 2: Build the request components.

Create a string variable to hold the API endpoint URI and two dictionaries, one for the request header and one for the body JSON. These are the same tasks you completed in the Postman application.

a. Create a variable named **api_url** and assign the URL (adjust the IP address to match the router's current address).

```
api_url = "https://10.10.20.48/restconf/data/ietf-interfaces:interfaces"
```

Create a dictionary variable named **headers** that has keys for **Accept** and **Content-type** and assign the keys the value **application/yang-data+json**.

```
headers = {
            "Accept": "application/yang-data+json",
            "Content-type":"application/yang-data+json"
}
```

a. Create a Python tuple variable named **basicauth** that has two keys needed for authentication, **username** and **password**.

```
      basicauth = ("developer", "C1sco12345")
```

```
Send the request.
```

You will now use the variables created in the previous step as parameters for the **requests.get()** method. This method sends an HTTP GET request to the RESTCONF API. You will assign the result of the request to a variable name **resp**. That variable will hold the JSON response from the API. If the request is successful, the JSON will contain the returned YANG data model.

b. Enter the following statement:

```
resp = requests.get(api_url, auth=basicauth, headers=headers, verify=False)
```

The various elements of this statement are:

Element	Explanation
`resp`	the variable to hold the response from the API.
`requests.get()`	the method that actually makes the GET request.
`api_url`	the variable that holds the URL address string
`auth`	the tuple variable created to hold the authentication information
`headers=headers`	a parameter that is assigned to the headers variable
`verify=False`	disables verification of the SSL certificate when the request is made

a. Save your script and run it. There will not be any output yet but the script should run without errors. If not, review the steps and make sure your code does not contain any errors.

Step 3: Evaluate the response.

Now the YANG model response values can be extracted from the response JSON.

b. The response JSON is not compatible with Python dictionary and list objects so it is converted to Python format. Create a new variable called **response_json** and assign the variable **resp** to it adding the **json()** method to convert the JSON. The statement is as follows:

```
response_json = resp.json()
```

```
 You can verify that your code returns the JSON in the IDLE Shell
by temporarily adding a print statement to your script, as follows:
```

```
print(response_json)
```

Save and run your script. You should get output similar to the following although your service ticket number will be different:

a. To prettify the output, use the json.dumps() function with the “indent” parameter:

```
print(json.dumps(response_json, indent=4))
```

b. Save and run your script. If you experience errors, check the code again.

Part 2: Modify interface configuration with RESTCONF in Python

Step 4: Create the Python HTTP PUT request

In this part, you will use Python to request a RESTCONF API with a PUT method to create or modify existing configuration.

Step 5: Import modules and disable SSL warnings.

c. In IDLE, click **File > New File** to open IDLE Editor.

d. Save the file as **lab 2.5 part2.py**.

e. Enter the following commands to import the modules and disable SSL certificate warnings:

```
import json
import requests
requests.packages.urllib3.disable_warnings()
```

The **json** module includes methods convert JSON data to Python objects and vice versa. The **requests** module has methods that will let us send REST requests to a URI.

Step 6: Build the request components.

Create a string variable to hold the API endpoint URI and two dictionaries, one for the request header and one for the body JSON. These are the same tasks you completed in the Postman application.

f. Create a variable named **api_url** and assign the URL that targets the **Loopback99** interface.

```
api_url = "https://10.10.20.48:443/restconf/data/ietf-
interfaces:interfaces/interface=Loopback99"
```

a. Create a dictionary variable named **headers** that has keys for **Accept** and **Content-type** and assign the keys the value **application/yang-data+json**.

```
headers = { "Accept": "application/yang-data+json",
            "Content-type":"application/yang-data+json"
}
```

a. Create a Python tuple variable named **basicauth** that has two keys needed for authentication, **username** and **password**.

```
      basicauth = ("developer", "C1sco12345")
```

b. Create a Python dictionary variable yangConfig holding the YANG data to create new interface Loopback99 (you use here the dictionary data from the Postman lab before, be aware that the JSON's boolean **true** is in Python **True** with capital "T"):

```
yangConfig = {
    "ietf-interfaces:interface": {
        "name": "Loopback99",
        "description": "LAB 2.5 Python",
        "type": "iana-if-type:softwareLoopback",
        "enabled": True,
        "ietf-ip:ipv4": {
            "address": [
                {
                    "ip": "192.168.10.101",
                    "netmask": "255.255.255.0"
                }
            ]
        },
        "ietf-ip:ipv6": {}
    }
}
```

Step 7: Send the PUT request.

You will now use the variables created in the previous step as parameters for the **requests.put()** method. This method sends an HTTP PUT request to the RESTCONF API. You will assign the result of the request to a variable name **resp**. That variable will hold the JSON response from the API. If the request is successful, the JSON will contain the returned YANG data model.

a. Enter the following statement:

```
resp = requests.put(api_url, data=json.dumps(yangConfig), auth=basicauth,
headers=headers, verify=False)
if(resp.status_code >= 200 and resp.status_code <= 299):
    print("STATUS OK: {}".format(resp.status_code))
else:
    print("Error code {}, reply: {}".format(resp.status_code, resp.json()))
```

The various elements of this statement are:

Element	Explanation
`resp`	the variable to hold the response from the API.
`requests.get()`	the method that actually makes the GET request.
`api_url`	the variable that holds the URL address string
`data`	the data to the sent to the API endpoint
`auth`	the tuple variable created to hold the authentication information
`headers=headers`	a parameter that is assigned to the headers variable
`verify=False`	disables verification of the SSL certificate when the request is made
`resp.status_code`	The HTTP status code in the API Request reply

a. Save your script and run it.

```
PS C:\Users\pablo\Documents\UTNG Cuatrimestre 4\Programación de Redes\Unidad III\Ejercicios> & C:/Users/pablo/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/pablo/Documents/UTNG Cuatrimestre 4/Programación de Redes/Unidad III/Ejercicios/2.5_lab.py"
STATUS OK: 201
```

Lo ejecute una vez y después lo ejecute de nueve en el IDE de Python muestra el siguiente status code

```
========== RESTART: C:\Users\pablo\Documents\UTNG Cuatrimestre 4\Programación de Redes\Unidad III\Ejercicios\2.5_lab.py ==========
STATUS OK: 204
```

Si bien 200 OK es una respuesta válida y más común, devolver un 204 Sin contenido podría tener sentido ya que no hay absolutamente nada que devolver.

b. Verify using the IOS CLI that the new Loopback99 interface has been created (sh ip int brief).

```
========== RESTART: C:\Users\pablo\Documents\UTNG Cuatrimestre 4\Programación de Redes\Unidad III\Ejercicios\ssh2.2.py ==========
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       10.10.20.48     YES NVRAM  up                    up
GigabitEthernet2       unassigned      YES NVRAM  administratively down down
GigabitEthernet3       unassigned      YES NVRAM  administratively down down
Loopback1              192.168.20.106  YES manual up                    up
Loopback99             192.168.10.101  YES other  up                    up
```

conclusión

Dentro de esto código estamos haciendo diferente forma en la creación de una interfaz lookback con un tipo de código en Python que empieza con la URl después con el HEADERS que se usa para guardar las variables donde se guardan los metadatos que se mandan a al API para poder obtener la información necesaria. La creación de la interfaz es mediante una variable que después se pondrá en el REQUEST.PUT para introducir el código en la router con la interfaz creada. El funcionamiento del scrip va por partes , la primera entra a la interfaz mediante el URL que contiene la lookback 99, la tercera el header que es una variable para que pueda entrar el router, la variable yangConfig crea la interfaz lookback 99 con una dirección ipv4 y una descripción mediante código XML , regresara un status code que compararamos y mostrarlo en la consola.

¿Qué es RESCONF?

Esta guía no pretende indagar en detalle los conceptos de RESTCONF ni de los modelos de datos de YANG.

¿Que usa RESCONF dentro de una URL?

- RESTCONF usa HTTPS como transporte. Esto significa que implementa los métodos GET, POST, PUT, DELETE.
- Los modelos de datos están escritos en YANG. Los datos que recibimos en las peticiones están en formato JSON.
- Existen dos tipos de modelos de datos: abiertos (IEEE, IETF) y nativos (de Cisco, o cualquier otro fabricante).

Para usar RESTCONF, una vez está habilitado en el equipo, debemos en primer lugar armar la URI en la cual se encuentra el recurso al que queremos acceder.

¿Para qué sirve BASICAUTH?

La autenticación básica o BasicAuth es la forma más básica de autenticación disponible para las aplicaciones Web. Es el tipo de autenticación más simple disponible, pero presenta problemas de seguridad que no la hacen recomendable en muchas situaciones.

Tipos de Estatus Code de respuesta

- 200 OK. La petición de solicitud fue correcta.
- 201 Created. La solicitud ha tenido éxito y se ha creado un nuevo recurso como resultado.
- 202 Accepted. La petición de solicitud ha sido aceptada pero el proceso no ha sido completado.
- 203 Non-Authoritative Information.La petición se ha completado correctamente pero la respuesta fue solicitada a un tercero o a una copia local.
- 204 No content. La petición se ha procesado correctamente pero el resultado no tiene contenidos, está vacío.
- 400 Bad Request. El servidor no pudo interpretar la solicitud debido a que el cliente empleó una sintaxis no válida.

- 401 Unauthorized. La petición de respuesta requiere de la autentificación de usuario.
- 403 Forbidden. El servidor entendió la petición, pero se niega a proporcionarla.
- 404 Not Found. El servidor no ha encontrado nada que concuerde con la petición solicitada.

Dentro de esta lista no están todos pero son los más comunes dentro de una petición hacia una API.