RESTCONF

LAB - 3

Disclaimer

This Lab Guide is designed to assist candidates to facilitate Technology learning. While every effort has been made to ensure that all material is as complete and accurate as possible, the enclosed material is presented on an “as is” basis. Neither the authors nor RSTForum assume any liability or responsibility to any person or entity with respect to loss or damages incurred from the information contained in this Lab guide. This workbook was developed by RSTForum. Any similarities between material presented in this Lab Guide and any other Lab Guide or any other material is completely coincidental.

Hands on with cURL: client for URLs

This lab exercise will get you started with RESTCONF connections, retrieving data, and sending configurations to the network.

Understanding cURL:

The cURL name is a play on words: "Client for URLs" and "cURL URL Request Library." It's a simple command-line tool with lots of options for putting together a REST API request.

When using cURL to make REST API method calls, you typically use the following options and arguments:

- -X followed by a request verb such as GET, PUT, POST, PATCH, or DELETE.
- -H followed by a header such as a token to send with the requests.

For example, a GET request is often used to get a list or to get details about a single resource. So lets walk through an example,

Step 1: Use cURL for RESTCONF configuration

1. The following is a sample RESTCONF request to router to check the allowed methods available for restconf configurations

```
curl -i -k -X "OPTIONS" "https://10.0.0.1:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity" -H 'Accept: application/yang-data+json' -u 'cisco:cisco'
```

2. The router will reply with allowed methods available for RESTCONF.

```
< HTTP/1.1 200 OK
Server: nginx
Date: Wed, 29 Apr 2020 12:40:50 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
Allow: DELETE, GET, HEAD, PATCH, POST, PUT, OPTIONS
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Accept-Patch: application/yang-data+xml, application/yang-data+json
Pragma: no-cache
```

3. The following is a sample RESTCONF request to get primary ip address details of configured GigabitEthernet interfaces.

```
curl -i -k -X "GET" "https://10.0.0.1:443/restconf/data/Cisco-IOS-XE-native:native/interface?fields=GigabitEthernet/ip/address/primary;name" -H 'Accept: application/yang-data+json' -u 'cisco:cisco'
```

4. The router will reply with allowed methods available for RESTCONF.

```
HTTP/1.1 200 OK
Server: nginx
Date: Wed, 29 Apr 2020 14:03:46 GMT
Content-Type: application/yang-data+json
Transfer-Encoding: chunked
Connection: keep-alive
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Pragma: no-cache

{
  "Cisco-IOS-XE-native:interface": {
    "GigabitEthernet": [
      {
        "ip": {
          "address": {
            "primary": {
              "address": "10.0.0.1",
              "mask": "255.255.255.0"
            }
          }
        }
      },
      {
        "ip": {
          "address": {
            "primary": {
              "address": "10.255.255.1",
              "mask": "255.255.255.0"
            }
          }
        }
      }
    ]
  }
}
```

5. The following is a sample RESTCONF request to get ip address details of interface GigabitEthernet 2.

```
curl -i -k -X "GET" "https://10.0.0.1:443/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=2/ip" -H 'Accept: application/yang-data+json' -u 'cisco:cisco'
```

6. The router will reply with allowed methods available for RESTCONF.

```
HTTP/1.1 200 OK
Server: nginx
Date: Wed, 29 Apr 2020 14:17:34 GMT
Content-Type: application/yang-data+json
Transfer-Encoding: chunked
Connection: keep-alive
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Pragma: no-cache

{
  "Cisco-IOS-XE-native:ip": {
    "address": {
      "secondary": [
        {
          "address": "200.0.0.200",
          "mask": "255.0.0.0",
          "secondary": [null]
        }
      ],
      "primary": {
        "address": "10.255.255.1",
        "mask": "255.255.255.0"
      }
    }
  }
}
```

7. The following is a sample RESTCONF request to get primary ip address of interface GigabitEthernet 1.

```
curl -i -k -X "GET" "https://10.0.0.1:443/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=1/ip/address/primary" -H 'Accept: application/yang-data+json' -u 'cisco:cisco'
```

8. The router will reply with allowed methods available for RESTCONF.

```
HTTP/1.1 200 OK
Server: nginx
Date: Wed, 29 Apr 2020 14:19:28 GMT
Content-Type: application/yang-data+json
Transfer-Encoding: chunked
Connection: keep-alive
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Pragma: no-cache

{
  "Cisco-IOS-XE-native:primary": {
    "address": "10.0.0.1",
    "mask": "255.255.255.0"
  }
}
```

9. The following is a sample RESTCONF request to change hostname.

```
curl -i -k -X  PUT https://10.0.0.1/restconf/data/Cisco-IOS-XE-native:native/hostname -H 'Accept: application/yang-data+json' -H 'Content-Type: application/yang-data+json' -H 'cache-control: no-cache' -u 'cisco:cisco' -d '{"hostname": "HelloRST"}'
```

10. The following is a sample RESTCONF request to change Loopback ip address

```
curl -i -k -X PATCH https://10.0.0.1/restconf/data/Cisco-IOS-XE-native:native/interface/Loopback=1/ip/address/primary -H 'Accept: application/yang-data+json' -H 'Content-Type: application/yang-data+json' -H 'cache-control: no-cache' -u 'cisco:cisco' -d '{"primary": {"address": "1.1.1.1", "mask": "255.255.255.0"}}'
```

Step 2: Python for RESTCONF configuration

1. The following is a sample RESTCONF request in python to get IP address of loopback 1 interface.

```
import requests
import urllib3

USER = 'cisco'
PASS = 'cisco'

# disable warnings from SSL/TLS certificates
requests.packages.urllib3.disable_warnings()

url = "https://10.0.0.1/restconf/data/Cisco-IOS-XE-native:native/interface/Loopback=1/ip/addre
ss/primary"

headers = {
   'Accept': "application/yang-data+json",
   'Content-Type': "application/yang-data+json",
   'cache-control': "no-cache",
   }

response = requests.get(url, auth=(USER, PASS),
                        headers=headers, verify=False)
print(response.text)
```

2. The router will reply as below.

```
From cffi callback <function _verify_callback at 0x7f4be07c61d0>:
Traceback (most recent call last):
  File "/usr/lib/python2.7/dist-packages/OpenSSL/SSL.py", line 315, in wrapper
    _lib.X509_up_ref(x509)
AttributeError: 'module' object has no attribute 'X509_up_ref'
{
  "Cisco-IOS-XE-native:primary": {
    "address": "100.200.200.200",
    "mask": "255.255.255.0"
  }
}
```

3. The following is a sample RESTCONF request in python to get hostname of router.

```
import requests
import urllib3

USER = 'cisco'
PASS = 'cisco'

# disable warnings from SSL/TLS certificates
requests.packages.urllib3.disable_warnings()

url = "https://10.0.0.1/restconf/data/Cisco-IOS-XE-native:native/hostname"

payload = ""
headers = {
   'Accept': "application/yang-data+json",
   'Content-Type': "application/yang-data+json",
   'cache-control': "no-cache",
   }

print("----------------------")
print("RSTForum- Configuration Parameters:")
print("----------------------")
response = requests.request("GET",url, auth=(USER, PASS), data=payload,
                    headers=headers, verify=False)

print(response.text)
print("----------------------")
print("RSTForum- Thanks you:")
print("----------------------")
```

4. The router will reply as below.

```
From cffi callback <function _verify_callback at 0x7f87a2192050>:
Traceback (most recent call last):
  File "/usr/lib/python2.7/dist-packages/OpenSSL/SSL.py", line 315, in wrapper
    _lib.X509_up_ref(x509)
AttributeError: 'module' object has no attribute 'X509_up_ref'
{
  "Cisco-IOS-XE-native:hostname": "RSTForum"
}
```

5. The following is a sample RESTCONF request in python to post Loopback Ip address.

```
import requests
import urllib3

USER = 'cisco'
PASS = 'cisco'

# disable warnings from SSL/TLS certificates
requests.packages.urllib3.disable_warnings()

url = "https://10.0.0.1/restconf/data/Cisco-IOS-XE-native:native/interface/Loopback=1/ip/addre
ss/primary"

payload = "{\"primary\": {\"address\": \"88.88.88.88\", \"mask\": \"255.255.255.0\"}}"
headers = {
   'Accept': "application/yang-data+json",
   'Content-Type': "application/yang-data+json",
   'cache-control': "no-cache",
   }

print("----------------------")
print("RSTForum- Configuration Parameters:")
print("----------------------")
response = requests.request("PATCH",url, auth=(USER, PASS), data=payload,
                    headers=headers, verify=False)

print(response.text)
print("----------------------")
print("RSTForum- Thanks you:")
print("----------------------")
```

6. The router will reply as below.

```
From cffi callback <function _verify_callback at 0x7fa0034951d0>:
Traceback (most recent call last):
 File "/usr/lib/python2.7/dist-packages/OpenSSL/SSL.py", line 315, in wrapper
   _lib.X509_up_ref(x509)
AttributeError: 'module' object has no attribute 'X509_up_ref'
```

7. The following is a sample RESTCONF request in python to put hostname.

```
import requests
import urllib3

USER = 'cisco'
PASS = 'cisco'

# disable warnings from SSL/TLS certificates
requests.packages.urllib3.disable_warnings()

url = "https://10.0.0.1/restconf/data/Cisco-IOS-XE-native:native/hostname"

payload = "{\"hostname\": \"RSTForum\"}"
headers = {
   'Accept': "application/yang-data+json",
   'Content-Type': "application/yang-data+json",
   'cache-control': "no-cache",
   }
```

8. The router will reply as below.

```
From cffi callback <function _verify_callback at 0x7fe88bf8d050>:
Traceback (most recent call last):
  File "/usr/lib/python2.7/dist-packages/OpenSSL/SSL.py", line 315, in wrapper
    _lib.X509_up_ref(x509)
AttributeError: 'module' object has no attribute 'X509_up_ref'
```