Discovery 14: Utilize RESTCONF for Network Configuration

Task 1: Retrieve Device Configuration in Postman

In this procedure, you will configure the RESTCONF interface on a Cisco CSR1000v router, and, using Postman, you will retrieve the router's configuration. You will gradually build up the URL for getting the IP configuration of an interface.

Activity

Step 1: From the desktop, open the Terminal and SSH to CSR1kv1 using the IP address 10.0.0.20. The credentials are in the Job Aids.

```
student@student-workstation:~$ ssh cisco@10.0.0.20
Password:



csr1kv1#
```

Step 2: Enter the global configuration mode and enter the **restconf** command to enable the RESTCONF interface.

```
csr1kv1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
csr1kv1(config)#restconf
csr1kv1(config)#
```

Step 3: To send a request to the RESTCONF interface, the secure HTTP server needs to be enabled. Use the **ip http secure-server** command to enable it. Do not close the session yet.

```
csr1kv1(config)#ip http secure-server
csr1kv1(config)#
```

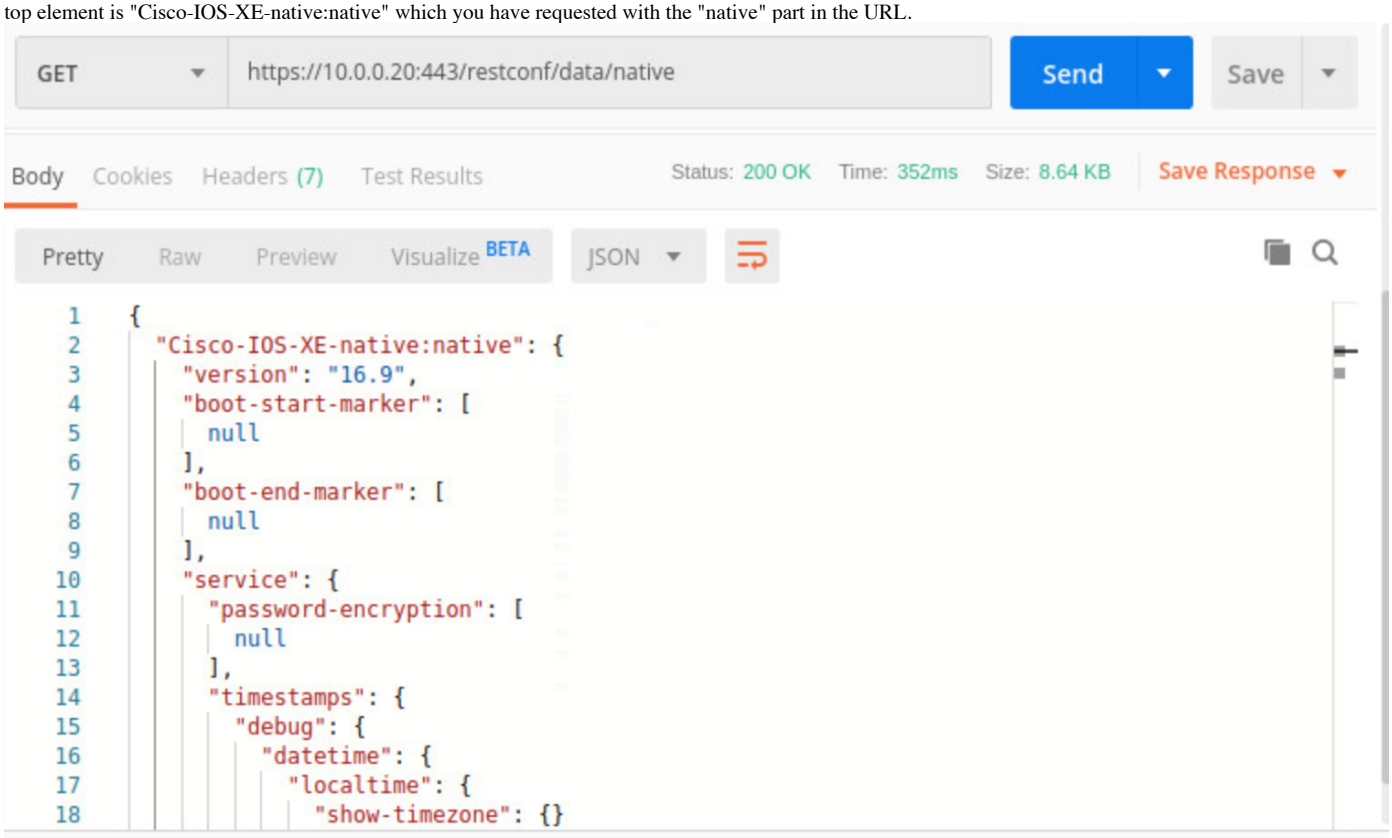
Step 4: The RESTCONF interface is enabled and you can start sending requests to the router. Since the certificate on the router is self-signed, the SSL certificate validation needs to be disabled before sending the request using Postman. From the desktop, open **Postman**, click **File > Settings.** A window with the settings will open. Disable the SSL certificate verification and close the settings.

Step 5: Click the **plus** sign to create a new tab with a request.

Step 6: Create a GET request. For the URL, use https://10.0.0.20:443/restconf/data/native to get the full configuration of the device. Set the authorization to basic, for the username and password use cisco. Set the Header key **Accept** to "application/yang-data+json."

Step 7: Send the request and check the response.

The response is displayed under the request. The response status should be "200 OK" and the Body contains the full configuration of the devices in JSON. The top element is "Cisco-IOS-XE-native:native" which you have requested with the "native" part in the URL.



Step 8: Add "/interface" at the end of the request's URL, so you retrieve the configurations of the interfaces.

The full URL is https://10.0.0.20:443/restconf/data/native/interface and the response should be as follows.

```
{
  "Cisco-IOS-XE-native:interface": {
    "GigabitEthernet": [
      {
        "name": "1",
        "ip": {
          "address": {
            "primary": {
              "address": "192.168.0.30",
              "mask": "255.255.255.0"
            }
          },
          "Cisco-IOS-XE-nat:nat": {
            "inside": [
              null
            ]
          }
        },
        "mop": {
          "enabled": false,
          "sysid": false
        },
        "Cisco-IOS-XE-ethernet:negotiation": {
          "auto": true
        }
      },
<... output omitted ...>
```

Step 9: Specify the third GigabitEthernet interface in the URL to get only its configuration. Use the "=" sign to match the interface with the current number.

The full URL is https://10.0.0.20:443/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=3.

```
{
  "Cisco-IOS-XE-native:GigabitEthernet": {
    "name": "3",
    "shutdown": [
      null
    ],
    "mop": {
      "enabled": false,
      "sysid": false
    },
    "Cisco-IOS-XE-ethernet:negotiation": {
      "auto": true
    }
  }
}
```

Step 10: Change the URL to get the IP address of the first GigabitEthernet interface.

The URL for getting the IP address of the first GigabitEthernet is https://10.0.0.20:443/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=1/ip/address.

```
{
  "Cisco-IOS-XE-native:address": {
    "primary": {
      "address": "192.168.0.30",
      "mask": "255.255.255.0"
    }
  }
}
```

Task 2: Interpret Configuration Using YANG Model

In this procedure, you will open a YANG model for the Cisco IOS XE router and interpret the model with the help of the JSON received from RESTCONF.

Activity

Step 1: From the desktop, open **Visual Studio Code**. Your working directory contains a YANG model, open it.

Step 2: The YANG model describes the structure of the configuration with the values and value formatting. Locate line 1085 where the container address is defined.

```
// interface * / ip address XXX
container address {
  description
    "Set the IP address of an interface";
  choice address-choice {
    case fixed-case {
      container primary {
        leaf address {
          type inet:ipv4-address;
        }
        leaf mask {
          type inet:ipv4-address;
        }
```

```
      }
      list secondary {
        key "address";
        leaf address {
          type inet:ipv4-address;
        }
        leaf mask {
          mandatory true;
          type inet:ipv4-address;
        }
        leaf secondary {
          description
            "Make this IP address a secondary address";
          mandatory true;
          type empty;
        }
      }
    }
```

Step 3: Compare the primary container with the IP address configuration of the GigabitEthernet 1 in JSON.

In the YANG file, the "container primary," which defines the primary IP address as necessary, while the second IP address is not, which means that when configuring the IP address to an interface it is necessary to define the primary before the secondary. The "container primary" also defines the IP address as inet:ipv4-address type, same as for the mask, both of which are required. The *list* keyword indicates that you might configure multiple secondary IP addresses.

```
# JSON
https://10.0.0.20:443/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=1/ip/address
{
  "Cisco-IOS-XE-native:address": {
    "primary": {
      "address": "192.168.0.30",
      "mask": "255.255.255.0"
    }
  }
}

# YANG
// interface * / ip address XXX
container address {
  description
    "Set the IP address of an interface";
  choice address-choice {
    case fixed-case {
      container primary {
        leaf address {
          type inet:ipv4-address;
        }
        leaf mask {
          type inet:ipv4-address;
        }
      }
      list secondary {
        key "address";
        leaf address {
          type inet:ipv4-address;
        }
        leaf mask {
          mandatory true;
          type inet:ipv4-address;
        }
        leaf secondary {
          description
            "Make this IP address a secondary address";
          mandatory true;
          type empty;
        }
      }
    }
```

Task 3: Update Configuration Using Postman

In this procedure, you will create a POST request for changing an interface configuration on a Cisco CSR1000v router.

Activity

Step 1: From the desktop, open **Postman** and in a new tab set the request method to **POST**.

Step 2: Set the authorization to basic and both the username and password to cisco. Set the Header keys **Accept** and **Content-Type** to "application/yang-data+json."

Step 3: Click on **Body** and select **Raw**. The bottom part changes to a text field where you will write the JSON structure.

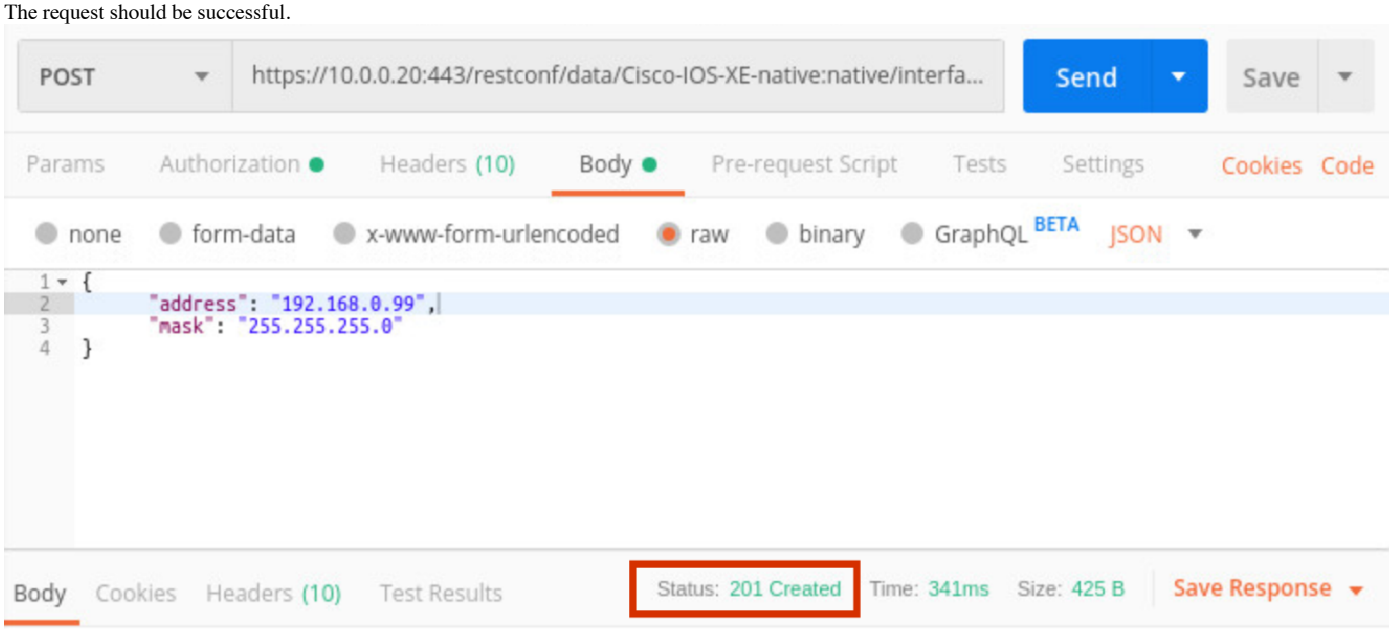
Step 4: Write a JSON structure that configures the IP address for the GigabitEthernet 3 to 192.168.0.99.

```
{
  "address": "192.168.0.99",
```

```
  "mask": "255.255.255.0"
}
```

Step 5: Send the request to the URL https://10.0.0.20:443/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=3/ip/address/primary.

The request should be successful.



Step 6: Switch to the Terminal window connected to the CSR1kv1 and verify that the configuration has been applied using the **show running-config interface GigabitEthernet 3** command.

```
csr1kv1#show running-config interface GigabitEthernet 3
Building configuration...

Current configuration : 131 bytes
!
interface GigabitEthernet3
 ip address 192.168.0.99 255.255.255.0
 shutdown
 negotiation auto
 no mop enabled
 no mop sysid
end
```

Task 4: Retrieve Device Configuration in Python

You will learn how to retrieve the device configuration using Python.

Activity

Step 1: Open Visual Studio Code and open the folder /home/workspace. Open the restconf.py file and check the containing code.

The code is missing a few statements which need to be written.

```
import sys, requests, urllib3, json

#CSR1kv1 Credentials
URL  = ''
USER = ''
PASS = ''

# Headers required for RESTCONF
headers = {'Content-Type': 'application/yang-data+json',
           'Accept': 'application/yang-data+json'}

# Disable warnings
urllib3.disable_warnings()

# Main method
def main():
    ##########################################
    # Retrieve Device Configuration in Python #
    ##########################################
    # URL for API calls
    url_ge1 = URL + ''

    # GET call to CSR1kv1, SSL checking turned off

    # Print the returned JSON
```

```
    ####################################
    # Update Configuration Using Python #
    ####################################
    # URL for API calls
    url_ge3 = URL + ''

    # New IP address

    # PUT call to CSR1kv1, SSL checking turned off


    # Print the response's status code

if __name__ == '__main__':
    main()
```

Step 2: Add the URL and the credentials for the CSR1kv1 device.

```
#CSR1kv1 Credentials
URL  = 'https://10.0.0.20:443/'
USER = 'cisco'
PASS = 'cisco'
```

Step 3: Add the part of the URL used previously for accessing the IP address of the first interface.

```
# URL for API calls
url_ge1 = URL + 'restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=1/ip/address'
```

Step 4: The headers are set up correctly. In the **main** method, write the GET request using module requests and **get** method. Set the pass the arguments *url*, *auth*, *headers*, and set *verify* to False for not validating the router's certificate. Save the response into a variable named *response*.

```
response = requests.get(url_ge1,
                        auth=(USER, PASS),
                        headers=headers,
                        verify=False)
```

Step 5: Print the returned response.

```
# Print the returned JSON
print(response.text)
```

Step 6: Run the code.

The output is the same as when you have used Postman.

```
{
  "Cisco-IOS-XE-native:address": {
    "primary": {
      "address": "192.168.0.30",
      "mask": "255.255.255.0"
    }
  }
}
```

Task 5: Update Configuration Using Python

The interface GigabitEthernet 3 has an IP address configured. You will learn how to change the configuration of the interface using Python.

Activity

Step 1: Add part of the URL to point to the primary address of the interface GigabitEthernet 3.

```
# URL for API calls
url_ge3 = URL + 'restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=3/ip/address/primary/addr
```

Step 2: In the same file, restconf.py, create a dictionary object with the keys address and mask, with their corresponding values of "192.168.0.88" and "255.255.255.0." Save the object in a variable named *ip_address*.

```
    # New IP address
    ip_address = {"address": "192.168.0.88", "mask": "255.255.255.0"}
```

Step 3: Write a PUT request and use the data argument for passing the JSON structure. Use the **dumps** method of JSON object to transform the content of the variable *ip_address*.

```
response_change = requests.put(url_ge3,
                        auth=(USER, PASS),
                        headers=headers,
                        verify=False,
                        data=json.dumps(ip_address))
```

Step 4: Print the response code of the variable *response_change* using the *status_code* parameter.

```
# Print the response's status code
print(response_change.status_code)
```

Step 5: Run the code.

The printed code should be 204.

```
204
```

Step 6: Switch to the Terminal window connected to the CSR1kv1 and verify that the configuration has been applied using the **show running-config interface GigabitEthernet 3** command.

```
csr1kv1#show running-config interface GigabitEthernet 3
Building configuration...

Current configuration : 131 bytes
!
interface GigabitEthernet3
 ip address 192.168.0.88 255.255.255.0
 shutdown
 negotiation auto
 no mop enabled
 no mop sysid
end
```