## Lab 21: Access a Cisco IOS XE device using RESTCONF

### Case Study

Before adopting automation, FastCom, a growing e-commerce company, faced significant challenges in its data center operations. Manually configuring servers, deploying applications, and managing network changes across hundreds of devices was a tedious, error-prone, and time-consuming process. This led to frequent configuration drift, extended service outages caused by human error, and a slow pace in rolling out new features or scaling infrastructure to meet fluctuating customer demand. Consequently, customers often experienced service disruptions, slow loading times, and delays in accessing new products, directly impacting customer satisfaction and revenue.

After adopting a comprehensive data center automation strategy, the company underwent a transformative shift. They implemented tools for infrastructure as code, automated provisioning, and continuous configuration management. These advancements enabled the deployment of new servers and applications in minutes rather than hours or days, ensuring consistent and error-free configurations. The immediate benefits included a significant reduction in operational costs, improved system uptime, and the ability to scale infrastructure rapidly and efficiently.

For customers, this translated into a vastly improved experience: near-zero downtime, consistently fast website and application performance, and quicker access to innovative features and products. Ultimately, this enhanced customer loyalty and strengthened the company’s market competitiveness.

### Business Challenge

FastCom faces significant challenges in managing large, complex networks where manual configuration remains the standard practice. The company struggles with slow provisioning of new services, frequent human errors that lead to network outages, and a lack of real-time visibility into the network’s operational state. Scaling operations to meet increasing demand becomes extremely difficult, as each change requires labor-intensive, device-by-device configuration. This approach results in high operational costs and severely limits the organization’s agility.

Moreover, manual processes negatively affect network performance, leading to increased downtime and inefficient utilization of resources. To overcome these challenges, the company’s IT team onboarded you as a Certified DevNet Associate to design and implement an effective, automation-driven solution.

### Solution

You implemented a solution using RESTCONF that effectively addresses these challenges. By adopting RESTCONF, the company benefits from a standardized, programmatic interface to its network devices. This enables the automation of configuration changes, rapid deployment of new services, and enforcement of consistent configurations across the infrastructure, thereby significantly reducing the likelihood of human error.

The RESTCONF protocol offers a streamlined subset of NETCONF functionalities through a RESTful API interface. It allows RESTful API requests to be conducted to an IOS XE device, with replies being sent in XML or JSON format.

You will construct and send API requests to the RESTCONF service that is running on the Cisco IOS XE router using Postman in the first lab segment. In the second part, you will develop Python scripts to replicate the tasks performed using Postman.

1. Launch the DENVET Cisco IOS XE VM and verify connectivity.
2. Configure an IOS XE device for RESTCONF access.
3. Open and configure Postman.
4. Use Postman to send GET requests.
5. Use Postman to send a PUT request.
6. Use a Python script to send GET requests.
7. Use a Python script to send a PUT request.

|  |
| --- |
| **// Launch the DENVET Cisco IOS XE VM and Verify Connectivity**   1. Go to the following website link:   **https://developer.cisco.com/sandbox.html?ReturnUrl=https://devnetsandbox.cisco.com**.Thenloginto DevNet Sandbox.    2. Click on the **Labels** dropdown. Select the **Always-On** check box.    3. In the **Catalyst 8000 Always-On Sandbox**, click on the **Launch** button.    4. Click on the **Review Summary** button. Here you can manage the time duration of the environment. But for this lab, we are using the default 2 days.    5. Click on the **Launch Environment** button.    6. It will take a few seconds to complete the launch of an environment.    7. Click on the **I/O** tab. Here you will get your credentials to access the Cisco IOS XE device via SSH.    8. Turn on the Ubuntu VM and open the **Terminal**. Execute the following command: **ssh username@ip\_address** to SSH the Cisco IOS XE router.  The first time you SSH to the Cisco IOS XE router, the Ubuntu VM warns you about the authenticity of the Cisco IOS XE router. Because you trust the Cisco IOS XE router, answer **yes** to the prompt.  9. Enter your **password** and press **ENTER**. You should now be in the privileged EXEC mode for the Cisco IOS XE router. |

|  |  |
| --- | --- |
| **// Configure an IOS XE Device for RESTCONF Access**  1. On the Cisco IOX XE SSH session terminal, execute the following command: **show platform software yang-management process** to verify that the NETCONF SSH daemon (ncsshd) is running.    2. If the ncsshd daemon is not running, then execute the **configure terminal** command to enter global configuration mode. Then execute the **restconf** command to enable the RESTCONF service on the Cisco IOS XE router.  3. Execute the **exit** command to get out of global configuration mode. In the privileged mode, execute the **show platform software yang-management process** to verify that the required RESTCONF daemons are now running. Remember that ncsshd is the NETCONF service that may be active on your device. We do not require it for this lab. However, you will need **nginx**, the HTTPS server. This enables you to perform REST API calls to the RESTCONF service.  4. Execute the below-provided global configuration commands in the Cisco IOS XE router to enable the HTTPS server and specify that server authentication should use the local database.   |  | | --- | | configure terminal  ip http secure-server  ip http authentication local |   5. Execute the **exit** command to get out of global configuration mode. In the privileged mode, execute the **show platform software yang-management process** to verify that the HTTPS server (nginx) is now running.  **Note:** In the DevNet Sandbox of Cisco IOS XE, all these services are already running. |

|  |
| --- |
| **// Open and Configure Postman**  1. Click on the **Postman** application icon to open it. If this is your first time opening Postman, it may prompt you to establish an account or sign in. You may simply skip signing in by clicking on the **Switch to Lightweight API Client** message at the bottom of the window. Signing in is not necessary to use this application.    2. Click on the **X** icon to close the **Unlock Postman with an account** dialog box.    3. Postman defaults to SSL certification verification. You will not be utilizing SSL certificates with the Cisco IOS XE router; therefore, set this option off. Click on the **File.** Then click on the **Settings**.    4. Click on the **General** tab, set the **SSL certificate verification** to **OFF**. After that, close the **Settings** dialog box. |

|  |
| --- |
| **// Use Postman to Send GET Requests**  1. In this part, you will use Postman to submit a GET request to the Cisco IOS XE router to make sure you can connect to the RESTCONF service.  2. Click on the **plus +** icon next to the **Launchpad** tab to open a **GET Untitled Request**. This interface is where you will complete all of your work in this lab.    3. The request type has already been set to GET. Leave the request type as **GET**. In the **Enter request URL** field, type the following URL: **https://devnetsandboxiosxe.cisco.com/restconf/** that will be used to access the RESTCONF service that is running on the Cisco IOS XE router.    4. Under the URL field, there are tabs listed for **Params**, **Authorization**, **Headers**, **Body**, **Pre-request Script**, **Test**, and **Settings**. In this lab, you will use **Authorization**, **Headers**, and **Body**.  5. Click on the **Authorization**    6. In the **Type**, click the down arrow next to **Inherit auth from parent** and select **Basic Auth**.    7. Enter the local authentication credentials for the Cisco IOS XE router. Enter your **Username** and **Password**.    8. Click on the **Headers**. Then click the **7 hidden**. You can verify that the Authorization key has a Basic value that will be used to authenticate the request when it is sent to the Cisco IOS XE router.    9. The Cisco IOS XE router supports sending and receiving data in XML and JSON formats. For this lab, we will be using JSON. In the **Headers** section, click in the first blank **Key** field and enter **Content-Type** for the type of key. In the **Value** field, enter **application/yang-data+json**. This instructs Postman to send JSON data to the Cisco IOS XE router.    10. Below your **Content-Type** key, add another key-value pair. The **Key** field is **Accept**, and the **Value** field is **application/yang-data+json**.    11. Postman now has all the information required to send the GET request. Click on the **Send** button. The JSON response from the Cisco ISO XE router should be shown underneath the **Temporary Headers**. This JSON response confirms that Postman can now submit more REST API queries to the Cisco IOS XE router.    12. Now that you have completed a successful GET request, you may use it as a template for future requests. At the top of Postman, next to the **Launchpad** tab, right-click on the **GET** tab that you just used and select **Duplicate Tab**.    13. Add the following URL: **https://devnetsandboxiosxe.cisco.com/restconf/data/ietf-interfaces:interfaces** to gather interface information. Then click on the **Send** button.  14. You should receive a JSON response from the Cisco IOS XE router that is similar to the screenshot below. Your output may vary depending on your router.    15. In this lab, we use the GigabitEthernet1 interface. To specify only one interface, modify the URL to only request information about this interface. Enter the following URL: **https://** **devnetsandboxiosxe.cisco.com/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet1** to get the information about the interface IP address. Then click on the **Send** button.  16. The Cisco IOS XE router should provide a JSON response similar to the result seen below. Your output may vary depending on your router. |

|  |  |
| --- | --- |
| **// Use Postman to Send a PUT Request**  1. In this section, we will configure Postman to send a PUT request to the Cisco IOS XE router to create a new loopback interface. For the **Type** of request, click the down arrow next to **GET** and select **PUT**.    2. Enter the following URL: **https://devnetsandboxiosxe.cisco.com/restconf/data/ietf-interfaces:interfaces/interface=Loopback2** to get the information about the interface IP address.    3. To send a PUT request, you need to provide the information for the body of the request. Next to the **Headers**tab, click **Body**. Then click the **Raw** radio button. The field is currently empty. If you click **Send** now, you will get error code **400 Bad Request** because Loopback2 does not exist yet, and you did not provide enough information to create the interface.    4. Fill in the **Body** section with the required JSON data to create a new Loopback2 interface. Copy and paste the below-provided code.   |  | | --- | | {  "ietf-interfaces:interface": {  "name": "Loopback2",  "description": "My first RESTCONF loopback",  "type": "iana-if-type:softwareLoopback",  "enabled": true,  "ietf-ip:ipv4": {  "address": [  {  "ip": "10.2.2.2",  "netmask": "255.255.255.0"  }  ]  },  "ietf-ip:ipv6": {}  }  } |     5. Click on the **Send** button to send the PUT request to the Cisco IOS XE router. Under the Body area, you should notice the HTTP response code **Status: 201 Created**. This shows that the resource was generated successfully.    6. Switch to the Cisco IOS XE router SSH terminal session window. Execute the following command: **show ip interface brief** to verify that the loopback interface was created successfully. |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **// Use a Python script to send GET requests**  1. Open Visual Studio Code. Then click on **File**. After that, click on **Open Folder…**    2. Select the **devnet-src** directory. Then click on the **OK** button to open it.    3. Drag up the **Terminal** window in Visual Studio Code. Execute the **mkdir restconf** command to create a directory.    4. In the **EXPLORER** pane under **DEVNET-SRC**, right-click the **restconf** directory and select **New File**.    5. Name the file **restconf-get.py**.    6. Copy and paste the below-provided script in the **restconf-get.py** to import the modules that are required and disable SSL certificate warnings.   |  | | --- | | import json  import requests  requests.packages.urllib3.disable\_warnings() |   The **json** module contains methods for converting JSON data into Python objects and vice versa. The **requests** module contains methods for sending REST requests to a URL.    7. Copy and paste the below-provided code in the **restconf-get.py** to access the interface information on the Cisco IOS XE router.   |  | | --- | | api\_url = "https://devnetsandboxiosxe.cisco.com/restconf/data/ietf-interfaces:interfaces" |     8. Copy and paste the below-provided 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"  } |     9. Copy and paste the below-provided Python tuple variable named **basicauth** that has two keys needed for authentication, **username** and **password**. Here, enter your username and password.   |  | | --- | | basicauth = ("Username", "Your\_Password") |     10. Use the variables defined in the previous step to define the request parameters. The **requests.get()** method. This method initiates an HTTP GET request to the RESTCONF API on the Cisco IOS XE router. Set the outcome of the request to a variable called **resp**. That variable will contain the API's JSON answer. If the request is successful, the JSON will include the returned YANG data model. Copy and paste the below-provided script in the **restconf-get.py.**   |  | | --- | | resp = requests.get(api\_url, auth=basicauth, headers=headers, verify=False) |     11. To see the HTTP response code. Copy and paste the below-provided print statement in the **restconf-get.py**. Press **Ctrl+S** to save it.   |  | | --- | | print(resp) |     12. Execute **cd restconf** to go inside the directory. Then execute the following command: **python3 restconf-get.py.** You should see the output shown below in the screenshot.    13. You may now retrieve YANG model response values from the response JSON. Copy and paste the below-provided code to convert the response into Python format.   |  | | --- | | response\_json = resp.json() |     14. Copy and paste the below-provided print statement in the **restconf-get.py** to display the JSON data. Press **Ctrl+S** to save it.   |  | | --- | | print(response\_json) |     15. Execute the following command: **python3 restconf-get.py.** You should get output similar to shown in the screenshot below.    16. To prettify the output, edit your print statement to use the **json.dumps()** function with the **indent** parameter. Copy and paste the below-provided code in the **restconf-get.py**. Press **Ctrl+S** to save it.   |  | | --- | | print(json.dumps(response\_json, indent=4)) |     17. Execute the following command: **python3 restconf-get.py.** You should get the output shown below in the screenshot. This output is virtually identical to the output of your first Postman GET request. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **// Use a Python script to send a PUT request**  1. In the **EXPLORER** pane under **DEVNET-SRC**, right-click the **restconf** directory and select **New File**.    2. Name the file **restconf-put.py**.    3. Copy and paste the below-provided script in the **restconf-put.py** to import the modules that are required and disable SSL certificate warnings.   |  | | --- | | import json  import requests  requests.packages.urllib3.disable\_warnings() |     4. Copy and paste the below-provided code in the **restconf-get.py** to access the interface information on the Cisco IOS XE router.   |  | | --- | | api\_url = "https://devnetsandboxiosxe.cisco.com/restconf/data/ietf-interfaces:interfaces/interface=Loopback3" |     5. Copy and paste the below-provided 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"  } |     6. Copy and paste the below-provided Python tuple variable named **basicauth** that has two keys needed for authentication, **username** and **password**. Here, enter your username and password.   |  | | --- | | basicauth = ("Username", "Your\_Password") |     7. Create a Python dictionary variable **yangConfig** that will hold the YANG data that is required to create the new interface Loopback3. You can use the same dictionary that you used previously in Postman. However, change the interface number and address. Also, be aware that Boolean values must be capitalized in Python. Therefore, make sure that the **T** is capitalized in the key-value pair for **“enabled”: True**. Copy and paste the below-provided code in the **restconf-get.py**.   |  | | --- | | yangConfig = {  "ietf-interfaces:interface": {  "name": "Loopback3",  "description": "My second RESTCONF loopback",  "type": "iana-if-type:softwareLoopback",  "enabled": True,  "ietf-ip:ipv4": {  "address": [  {  "ip": "10.3.3.3",  "netmask": "255.255.255.0"  }  ]  },  "ietf-ip:ipv6": {}  }  } |     8. Use the variables defined in the previous step to define the request parameters. The **requests.get()** method. This method initiates an HTTP GET request to the RESTCONF API on the Cisco IOS XE router. Set the outcome of the request to a variable called **resp**. That variable will contain the API's JSON answer. If the request is successful, the JSON will include the returned YANG data model. Copy and paste the below-provided code in the **restconf-get.py.**   |  | | --- | | resp = requests.put(api\_url, data=json.dumps(yangConfig), auth=basicauth, headers=headers, verify=False) |     9. Copy and paste the below-provided script in the **restconf-get.py** to handle the response. If the response is one of the HTTP success messages, the first message will be printed. Any other code value is considered an error. The response code and error message will be printed if an error has been detected. Press **Ctrl+S** to save it.   |  | | --- | | if(resp.status\_code >= 200 and resp.status\_code <= 299):  print("STATUS OK: {}".format(resp.status\_code))  else:  print('Error. Status Code: {} \nError message: {}'.format(resp.status\_code,resp.json())) |     10. Execute the following command: **python3 restconf-get.py** to send the PUT request to the Cisco IOS XE router. You should get a **201 Status Created**.    11. Switch to the Cisco IOS router SSH session terminal window and execute the following command: **show ip interface brief** to verify that the interface was created successfully. |