Packet Tracer - Implement REST APIs with an SDN Controller

Addressing Table

Note: All subnet masks are /24 (255.255.255.0).

Device	Interface	IP Address
R1	G0/0/0	192.168.101.1/24
	S0/1/0	192.168.1.2
R2	G0/0/0	192.168.102.1
	S0/1/1	192.168.2.2
R3	G0/0/0	10.0.1.1
	G0/0/1	10.0.2.1
	S0/1/0	192.168.1.1
	S0/1/1	192.168.2.1
SWL1	VLAN 1	192.168.101.2
SWL2	VLAN 1	192.168.102.2
SWR1	VLAN 1	10.0.1.2
SWR2	VLAN 1	10.0.1.3
SWR3	VLAN 1	10.0.1.4
SWR4	VLAN 1	10.0.1.5
Admin	NIC	10.0.1.129
PC1	NIC	10.0.1.130
PC2	NIC	10.0.2.129
PC3	NIC	10.0.2.130
PC4	NIC	192.168.102.3
Example Server	NIC	192.168.101.100
PT-Controller	NIC	192.168.101.254

Objectives

Part 1: Launch the DEVASC VM

Part 2: Verify External Connectivity to Packet Tracer

Part 3: Request an Authentication Token with Postman

Part 4: Send REST Requests with Postman

Part 5: Send REST Requests with VS Code

Part 6: Send REST Requests Inside Packet Tracer

Background / Scenario

In this Packet Tracer activity, you will use the Packet Tracer Network Controller and associated API documentation to send REST requests from Postman and from Visual Studio Code (VS Code). Packet Tracer also supports a Python coding environment. Therefore, in the final Part of this activity, you will send REST requests from within Packet Tracer.

Required Resources

- 1 PC with operating system of your choice
- Virtual Box or VMWare
- DEVASC Virtual Machine

Instructions

Part 1: Launch the DEVASC VM

If you have not already completed the **Lab - Install the Virtual Machine Lab Environment**, do so now. If you have already completed that lab, launch the DEVASC VM now.

Part 2: Verify External Connectivity to Packet Tracer

In this Part, you will verify that Packet Tracer can be accessed by other applications on the DEVASC VM. This activity must be completed entirely within the DEVASC virtual machine environment. Support for other setups is not provided.

Step 1: If you have not done so already, open the Packet Tracer activity.

a. Within the DEVASC VM, access your course curriculum in the Chromium browser.

b. Navigate to the page for this activity.

c. Download and launch the file **Packet Tracer - Implement REST APIs with an SDN Controller.pka** associated with these instructions.

Step 2: Verify Packet Tracer's settings for external access.

a. Click **Options > Preferences > Miscellaneous**. Under **External Network Access**, verify that **Enable External Access for Network Controller REST API** is checked.

b. Close the **Preferences** window.

c. Click **PT-Controller0 > Config**.

d. On the left, under **REAL WORLD**, click **Controller**.

e. Check **Access Enabled** and make note of the port number, which is most likely **58000**. This is the port number you will need when externally accessing the Packet Tracer activity from Chromium, VS Code, and Postman later in this activity.

Step 3: Verify you can access Packet Tracer from another program on the DEVASC VM.

Open Chromium and navigate to **http://localhost:58000/api/v1/host**.

You will get the following response. This step verifies that you can externally access Packet Tracer and **PT-Controller0**. Notice that authorization requires a ticket. You will get an authorization token in the next Part.

```
{
```

```
    "response": {
        "detail": "Security Authentication Failure",
        "errorCode": "REST_API_EXTERNAL_ACCESS",
        "message": "Ticket-based authorization: empty ticket."
    },
    "version": "1.0"
} {
```

Part 3: Request an Authentication Token with Postman

In this Part, you will investigate the REST API documentation in Packet Tracer and use Postman to request an authentication token from the **PT-Controller0**. You can also do this in VS Code with a Python script.

Step 1: Investigate the REST API documentation for the Network Controller.

To see the REST API documentation for **PT-Controller0**, complete the following steps:

a. Click **Admin > Desktop > Web Browser**.

b. Enter **192.168.101.254**.

c. Log in to **PT-Controller0** with user **cisco** and password **cisco123!**.

d. Click the menu next to the Cisco logo and choose **API Docs**.

e. You can also access this same documentation from the Help menu. Click **Help > Contents**.

f. In the navigation pane on the left, scroll down about two-thirds of the way and click **Network Controller API**. This provides the same documentation you found on **PT-Controller0**.

g. In the API documentation, click **addTicket**. You will use this documentation in the next step.

Note: Some REST API functionality may not be available in the current version of Packet Tracer.

Step 2: Create a new POST request.

a. After reviewing the **addTicket** REST API Method documentation, open Postman. In the **Launch** area, click the plus sign to create a new **Untitled Request**.

b. Click the down arrow and change the type from **GET** to **POST**.

c. Enter the URL **http://localhost:58000/api/v1/ticket**.

d. Below the URL field, click **Body**. Change the type to **raw**.

e. Click the down arrow next to **Text** and change it to **JSON**. This change will also set the “Content-type” HTTP Header to “application/json” that is required for this API call.

f. Paste the following JSON object into the **Body** field. Make sure your code is properly formatted

```
{
    "username": "cisco",
    "password": "cisco123!"
}
```

Step 3: Send the POST request

a. Click **Send** to send the POST request to the **PT-Controller0**.

You should get a response similar to the following. However, **your_serviceTicket** will be an actual value.

```
{
    "response": {
```

```
            "idleTimeout": 900,
            "serviceTicket": "your_serviceTicket",
            "sessionTimeout": 3600
        },
        "version": "1.0"
    }
```

b. Copy the **serviceTicket** value without the quotes to a text file for later use.

Part 4: Send REST Requests with Postman

In this Part, you will use your service ticket to send three REST requests to the PT-Controller0.

Step 1: Create a new GET request for all network devices in the network.

a. In Postman, click the plus sign to create a new **Untitled Request**.

b. Enter the URL **http://localhost:58000/api/v1/network-device**.

c. Below the URL field, click **Headers**.

d. Under the last **KEY**, click the **Key** field and enter **X-Auth-Token**.

e. In the **Value** field, enter the value for your service ticket.

Step 2: Send the GET request.

Click **Send** to send the GET request to the **PT-Controller0**.

You should get a response listing the details that the controller has for the nine network devices in the network. The response for the first device is shown here.

```
{
    "response": [
        {
            "collectionStatus": "Managed",
            "connectedInterfaceName": [
                "GigabitEthernet0/0/0",
                "GigabitEthernet0",
                "FastEthernet0"
            ],
            "connectedNetworkDeviceIpAddress": [
                "192.168.101.1",
                "192.168.101.254",
                "192.168.101.100"
            ],
            "connectedNetworkDeviceName": [
                "R1",
                "NetworkController",
                "Example Server"
            ],
            "errorDescription": "",
            "globalCredentialId": "53046ecc-88c3-49f6-9626-ca8ab9db6725",
            "hostname": "SWL1",
            "id": "CAT1010BT47-uuid",
            "interfaceCount": "29",
```

```
            "inventoryStatusDetail": "Managed",
            "lastUpdateTime": "6",
            "lastUpdated": "2020-06-11 22:55:51",
            "macAddress": "000C.CF42.2B11",
            "managementIpAddress": "192.168.101.2",
            "platformId": "3650",
            "productId": "3650-24PS",
            "reachabilityFailureReason": "",
            "reachabilityStatus": "Reachable",
            "serialNumber": "CAT1010BT47-",
            "softwareVersion": "16.3.2",
            "type": "MultiLayerSwitch",
            "upTime": "4 hours, 55 minutes, 11 seconds"
        },
<output omitted>
    ],
    "version": "1.0"
}
```


Step 3: Duplicate the GET request and modify it for all hosts on the network.

a. In Postman, right-click the tab for your host **GET** request and choose **Duplicate Tab**.

b. All information in the ticket is the same except for the URL. Simply change **network-device** to **host**: **http://localhost:58000/api/v1/host**.

Step 4: Send the GET request.

Click **Send** to send the GET request to the **PT-Controller0**.

You should get a response listing the details that the controller has for the six host devices in the network. The response for the first device is shown here.


```
{
    "response": [
        {
            "connectedAPMacAddress": "",
            "connectedAPName": "",
            "connectedInterfaceName": "GigabitEthernet1/0/24",
            "connectedNetworkDeviceIpAddress": "192.168.102.2",
            "connectedNetworkDeviceName": "SWL2",
            "hostIp": "192.168.102.3",
            "hostMac": "00E0.F96C.155B",
            "hostName": "PC4",
            "hostType": "Pc",
            "id": "PTT08108MO8-uuid",
            "lastUpdated": "2020-06-11 22:49:32",
            "pingStatus": "SUCCESS"
        },
<output omitted>
    ],
    "version": "1.0"
}
```

Step 5: Close Postman to free up memory in the DEVASC VM.

Part 5: Send REST Requests with VS Code

In this Part, you will use Python script in VS Code to send the same API requests you sent in Postman. However, you will also use Python for loops to parse the JSON and display only specific key value pairs.

Step 1: Use a script to request a service ticket.

a. Open VS code.

b. Click **File > Open Folder...** and navigate to the **devnet-src/ptna** directory.

c. Click **OK**.

Notice in the **EXPLORE** pane on the left that three scripts are shown: **01_get-ticket.py**, **02_get-network-device.py**, and **03_get-host.py**. Review the code for each. Notice that the scripts for network devices and hosts require that you replace the **your_serviceTicket** value with the value Packet Tracer gave you when you requested a ticket. Request a new service ticket to see the function of the **01_get-ticket.py** script.

d. Open a terminal window in VS Code: Terminal > New Terminal.

e. Run the **01_get-ticket.py** to see output similar to the following.

```
devasc@labvm:~/labs/devnet-src/ptna$ python3 01_get-ticket.py
Ticket request status:  201
The service ticket number is:  your_serviceTicket
devasc@labvm:~/labs/devnet-src/ptna$
```

f. Replace the your_serviceTicket value in 02_get-network-device.py and 03_get-host.py with the value Packet Tracer gave you.

Step 2: Use a script to request a list of network devices.

Previously in Postman, the call to the network device's API returned a list of all nine network devices and all the information available for each device. However, the **02_get-network-device.py** script prints only the values of the keys that the programmer is interested in: **hostname**, **platformId**, and **managementIpAddress**.

In the terminal window, run the **02_get-network-device.py** script.

```
devasc@labvm:~/labs/devnet-src/ptna$ python3 02_get-network-device.py
Request status:  200
SWL1      3650     192.168.101.2
R1        ISR4300          192.168.1.2
R3        ISR4300          192.168.2.1
SWR1      3650     10.0.1.2
SWR2      3650     10.0.1.3
R2        ISR4300          192.168.2.2
SWL2      3650     192.168.102.2
SWR4      3650     10.0.1.5
SWR3      3650     10.0.1.4
devasc@labvm:~/labs/devnet-src/ptna$
```

Step 3: Use a script to request a list of host devices.

Similarly, the programmer chose to list specific information for each of the six host devices connected to the network.

In the terminal window, run the 03_get-host.py script.

```
devasc@labvm:~/labs/devnet-src/ptna$ python3 03_get-host.py
Request status:  200
PC4      192.168.102.3   00E0.F96C.155B          GigabitEthernet1/0/24
PC3      10.0.2.129      0004.9A42.C245          GigabitEthernet1/0/24
PC1      10.0.1.129      00E0.A330.3359          GigabitEthernet1/0/22
PC2      10.0.2.130      0060.47C1.A4DB          GigabitEthernet1/0/23
Admin    10.0.1.130      0050.0FCE.B095          GigabitEthernet1/0/21
Example Server   192.168.101.100         000A.413D.D793          GigabitEthernet1/0/3
devasc@labvm:~/labs/devnet-src/ptna$
```

Part 6: Send REST Requests Inside Packet Tracer (Optional)

In this Part, you will use the same scripts with one small edit to send the same API requests inside Packet Tracer that you sent from VS Code.

Step 1: Create a Project in Packet Tracer

a. In Packet Tracer, click the **Admin** PC.

b. Click the **Programming** tab.

c. There is currently no project. Click **New**.

d. Enter **REST APIs** as the **Name** and choose **Empty - Python** as the template.

e. Click **Create**.

The **REST APIs (Python)** project is now created with a blank **main.py** script.

Step 2: Modify the scripts to run inside Packet Tracer.

Access from one application to another on the same host machine requires that the port number be specified in the URL. However, Packet Tracer is simulating a real network. In the real world, you do not normally specify the port number when making API requests. In addition, you would use a domain name or IP address in the URL.

a. In **VS Code**, copy the code for **03_get-host.py**.

b. In the **Admin > Programming** tab, double-click the main.py script to open it.

c. Paste the code in the **main.py** script.

d. Change the **api_url**. Replace **localhost:58000/api/v1/host** with **192.168.101.254/api/v1/host**.

e. Edits are automatically saved. Click **Run**. Packet Tracer output does not exactly simulate what you see in the Linux command line. However, you should see similar output as shown below.

```
Starting REST APIs (Python)...
('Request status: ', 200)
('PC4', '\t', '192.168.102.3', '\t', '00E0.F96C.155B', '\t', 'GigabitEthernet1/0/24')
('PC3', '\t', '10.0.2.129', '\t', '0004.9A42.C245', '\t', 'GigabitEthernet1/0/24')
('PC1', '\t', '10.0.1.129', '\t', '00E0.A330.3359', '\t', 'GigabitEthernet1/0/22')
('PC2', '\t', '10.0.2.130', '\t', '0060.47C1.A4DB', '\t', 'GigabitEthernet1/0/23')
('Admin', '\t', '10.0.1.130', '\t', '0050.0FCE.B095', '\t', 'GigabitEthernet1/0/21')
('Example Server', '\t', '192.168.101.100', '\t', '000A.413D.D793', '\t',
'GigabitEthernet1/0/3')
REST APIs (Python) finished running.
```

f. Copy and paste **02_get-network-device.py** into the **main.py**. Change the URL and run it.

```
REST APIs (Python) finished running.
Starting REST APIs (Python)...
('Request status: ', 200)
('SWL1', '\t', '3650', '\t', '192.168.101.2')
('R1', '\t', 'ISR4300', '\t', '192.168.1.2')
('R3', '\t', 'ISR4300', '\t', '192.168.2.1')
('SWR1', '\t', '3650', '\t', '10.0.1.2')
('SWR2', '\t', '3650', '\t', '10.0.1.3')
('R2', '\t', 'ISR4300', '\t', '192.168.2.2')
('SWL2', '\t', '3650', '\t', '192.168.102.2')
('SWR4', '\t', '3650', '\t', '10.0.1.5')
('SWR3', '\t', '3650', '\t', '10.0.1.4')
REST APIs (Python) finished running
```