A. 8.3.5-lab-explore-yang-models

Networking Academy

Lab - Explore YANG Models

Objectives

Part 1: Launch the DEVASC VM

Part 2: Explore a YANG Model on GitHub

Part 3: Explore a YANG Model Using pyang

Background / Scenario

YANG models define the exact structure, data types, syntax and validation rules for the content of messages exchanged between a managed device and another system communicating with the device. Working with files using the YANG language can be a bit overwhelming for the level of details in these files.

In this lab, you will learn how to use the open source **pyang** tool to transform YANG data models from files using the YANG language, into a much easier to read format. Using the "tree" view transformation, you will identify what the key elements of the ietf-interfaces YANG model are.

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 DEVASC-LAB**, do so now. If you have already completed that lab, launch the DEVASC VM now.

Part 2: Explore a YANG Model on GitHub

In this Part, you will install pyang module into your DEVASC VM and explore how it transforms YANG files. Pyang simplifies working with YANG files. The module comes with a pyang command line executable that transforms YANG files into a more human-readable format.

Step 1: Explore Cisco IOS XE YANG models in the GitHub repository.

a. Open Chromium and navigate to https://github.com/YangModels/yang.

b. Under the **master** branch, navigate to the YANG models for the Cisco IOS XE version 16.9.3 by clicking the following directories: **vendor > cisco > xe > 1693**.

c. Scroll down below all the Cisco YANG models and find where the IETF models begin. Look for **ietf-interfaces.yang**.

d. Click **ietf-interfaces.yang** and scroll through all the container nodes, leaf nodes, and list nodes. If you are familiar with output from the IOS command show interfaces, then you should recognize some or all of the nodes. For example, around line 221 you will see the leaf enabled.

```
leaf enabled {
  type boolean;
```

```
        default "true";
        description
          "This leaf contains the configured, desired state of the
           interface.
           Systems that implement the IF-MIB use the value of this
           leaf in the 'running' datastore to set
           IF-MIB.ifAdminStatus to 'up' or 'down' after an ifEntry
           has been initialized, as described in RFC 2863.
           Changes in this leaf in the 'running' datastore are
           reflected in ifAdminStatus, but if ifAdminStatus is
           changed over SNMP, this leaf is not affected.";
        reference
          "RFC 2863: The Interfaces Group MIB - ifAdminStatus";
      }
```

Step 2: Copy the ietf-interfaces.yang model to a folder on your VM.

a. Open VS code.

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

c. Click **OK**.

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

e. Create a subdirectory called **pyang** in the /**devnet-src** directory.

```
devasc@labvm:~/labs/devnet-src$ mkdir pyang
devasc@labvm:~/labs/devnet-src$
```

f. Return to your Chromium tab where the **ietf-interfaces.yang** model is still open. Scroll back to the top, if necessary, and click Raw to display just the YANG model data.

g. Select and copy the URL.

h. In the terminal, go to the pyang folder.

i. Use **wget** to save the raw ietf-interfaces.yang file.

```
devasc@labvm:~/labs/devnet-src/pyang$ wget
https://raw.githubusercontent.com/YangModels/yang/master/vendor/cisco/xe/1693
/ietf-interfaces.yang
--2020-06-22 20:42:20--
https://raw.githubusercontent.com/YangModels/yang/master/vendor/cisco/xe/1693/ietf-
interfaces.yang
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.0.133,
151.101.192.133, 151.101.128.133, ...
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|151.101.0.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 24248 (24K) [text/plain]
Saving to: 'ietf-interfaces.yang'

ietf-interfac 100%  23.68K  --.-KB/s    in 0.05s

2020-06-22 20:42:21 (439 KB/s) - 'ietf-interfaces.yang' saved [24248/24248]

devasc@labvm:~/labs/devnet-src/pyang$
```

You now have a local version of the **ietf-interfaces.yang** model that you can manipulate with **pyang**.

Part 3: Explore a YANG Model Using pyang

In this Part, you will install the **pyang** module into your DEVASC VM and explore how it transforms the YANG model you copied from GitHub. Pyang simplifies working with YANG files. The module comes with a **pyang** command line executable that transforms YANG files into a more human readable format.

Step 1: Verify pyang is installed and up to date.

a. In VS Code, open a terminal window.

b. Verify that pyang is already installed with the **pyang -v** command. Your version number may be different than the one shown here. You can also

```
devasc@labvm:~/labs/devnet-src$ pyang -v
pyang 2.2.1
devasc@labvm:~/labs/devnet-src$
```

c. (Optional) You can verify that you have the latest pyang updates using the following **pip3** command. Any updates after this lab was written will be downloaded and installed.

```
devasc@labvm:~/labs/devnet-src$ pip3 install pyang --upgrade
Requirement already up-to-date: pyang in ./.local/lib/python3.8/site-packages (2.2.1)
Requirement already satisfied, skipping upgrade: lxml in ./.local/lib/python3.8/site-
packages (from pyang) (4.5.0)
devasc@labvm:~/labs/devnet-src$
```

Step 2: Transform the ietf-interfaces.yang model.

a. Navigate to the **pyang** directory.

```
devasc@labvm:~/labs/devnet-src$ cd pyang
devasc@labvm:~/labs/devnet-src/pyang$
```

b. Enter pyang -h | more to explore the options for transforming the YANG model. Look for the **-f** option as shown below. You will use the **tree** formatting option.

```
devasc@labvm:~/labs/devnet-src/pyang$ pyang -h | more
Usage: pyang [options] [<filename>...]

Validates the YANG module in <filename> (or stdin), and all its dependencies.

Options:
  -h, --help            Show this help message and exit
  -v, --version         Show version number and exit
<output omitted>
  -f FORMAT, --format=FORMAT
                        Convert to FORMAT.  Supported formats are: yang, yin,
                        dsdl, jstree, jsonxsl, capability, identifiers, jtox,
                        uml, name, omni, tree, depend, sample-xml-skeleton
<output omitted>
devasc@labvm:~/labs/devnet-src/pyang$
```

c. Transform the **ietf-interfaces.yang** model into a tree format with the following command. Notice that the **leaf enabled** is much easier to find and read in this format.

```
devasc@labvm:~/labs/devnet-src/pyang$ pyang -f tree ietf-interfaces.yang
```

```
ietf-interfaces.yang:6: error: module "ietf-yang-types" not found in search path
module: ietf-interfaces
  +--rw interfaces
  |  +--rw interface* [name]
  |     +--rw name                        string
  |     +--rw description?                string
  |     +--rw type                        identityref
  |     +--rw enabled?                    boolean
  |     +--rw link-up-down-trap-enable?   enumeration {if-mib}?
  +--ro interfaces-state
     +--ro interface* [name]
        +--ro name               string
        +--ro type               identityref
        +--ro admin-status       enumeration {if-mib}?
        +--ro oper-status        enumeration
        +--ro last-change?       yang:date-and-time
        +--ro if-index           int32 {if-mib}?
        +--ro phys-address?      yang:phys-address
        +--ro higher-layer-if*   interface-state-ref
        +--ro lower-layer-if*    interface-state-ref
        +--ro speed?             yang:gauge64
        +--ro statistics
           +--ro discontinuity-time    yang:date-and-time
           +--ro in-octets?            yang:counter64
           +--ro in-unicast-pkts?      yang:counter64
           +--ro in-broadcast-pkts?    yang:counter64
           +--ro in-multicast-pkts?    yang:counter64
           +--ro in-discards?          yang:counter32
           +--ro in-errors?            yang:counter32
           +--ro in-unknown-protos?    yang:counter32
           +--ro out-octets?           yang:counter64
           +--ro out-unicast-pkts?     yang:counter64
           +--ro out-broadcast-pkts?   yang:counter64
           +--ro out-multicast-pkts?   yang:counter64
           +--ro out-discards?         yang:counter32
           +--ro out-errors?           yang:counter32
devasc@labvm:~/labs/devnet-src/pyang$
```

B. 8.6.7-lab---construct-a-python-script-to-manage-webex-teams

Networking Academy

Lab - Construct a Python Script to Manage Webex Teams

Objectives

Part 1: Launch the DEVASC VM

Part 2: Get Your Webex Teams Access Token

Part 3: Test Your Access Token

Part 4: Manage People in Webex Teams

Part 5: Manage Rooms in Webex Teams

Part 6: Manage Memberships in Webex Teams

Part 7: Manage Messages in Webex Teams

Background / Scenario

In this lab, you will use Webex Teams APIs to authenticate, manage people, manage rooms, manage memberships to rooms, and send a message.

Required Resources

- 1 PC with operating system of your choice with Webex Teams Installed
- 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.

Note: Part of the **Lab - Install the Virtual Machine Lab Environment** is to install the Webex Teams application and add at least one other person to your contacts list. This must be completed before proceeding with this lab.

Part 2: Get Your Webex Teams Access Token

In this Part, you sign up for or log in to your Webex account, explore the API documentation, get your access token, and then test your access token that you will use in your API calls.

Step 1: Login to Webex

a. Open the Chromium web browser.

b. Go to the Webex developer website: https://developer.webex.com/

c. Log in if you already have an account. If not, go through the signup process.

Step 2: Explore the API documentation.

a. Click **Documentation**.

b. Webex has many APIs that you can use in your applications. Click the **API Reference** to expand its submenu

c. Explore all the varieties of API calls. In this lab, you will use API documentation for **People**, **Rooms**, **Membership**, and **Message**.

Step 3: Get your access token.

a. Still in **Documentation**, scroll to the back to the top, if necessary, and click **Getting Started** under **REST API**.

b. Under **Accounts and Authentication**, notice that Webex supports a personal access token. An authentication token is required for all REST API calls. Click the Copy icon under Your Personal Access Token.

Note: A Personal Access Token provides access to your account to anyone who knows it. Make sure to keep it secret.

Note: You will get a message stating that the token is valid for a set amount of time, which was 12 hours at the time this lab was written. You will need to get a new token and update your Python scripts if you return to this lab after your token expires.

c. Copy your personal access token into a text file for use later in this lab.

Part 3: Test Your Access Token

You can test your access token within the OpenAPI documentation on the developer site. However, you will use your token in Python scripts. Therefore, you should test that it works in a script, as well.

Step 1: Test your access token on the developer site.

You can test your access token inside the OpenAPI documentation at https://developer.webex.com.

a. Return to your browser and click **Documentation**, if necessary.

b. Under **API Reference**, click **People**, and then click **Get My Own Details**.

c. In the **Try it** panel on the far right, notice that your token is already populated.

d. You can click **Try it** or **Run** to test your access. You will see the **Response** with your personal information.

e. Click **Request** to see the full URL used to send the GET request. You will use this URL in the next step in your Python script.

f. In the middle section, you can review all the documentation for the **Response Properties**.

Step 2: Use a Python script to test your access token.

a. Open VS code. Then click **File > Open Folder...** and navigate to the **devnet-src/webex-teams** directory. Click **OK**.

b. In the **EXPLORER** panel, you should now see all the placeholder .py files you will use in this lab. Click the **authentciation.py** file.

c. Place the following code into the file. Be sure to replace **your_token_here** with your personal access token you copied in the previous step.

```
import requests
import json

access_token = 'your_token_here'
url = 'https://webexapis.com/v1/people/me'
```

```
headers = {
    'Authorization': 'Bearer {}'.format(access_token)
}
res = requests.get(url, headers=headers)
print(json.dumps(res.json(), indent=4))
```

d. Save and run the file. You should get the same output your saw in the OpenAPI documentation.

Note: The values for some of the keys have been truncated in the output below.

```
devasc@labvm:~/labs/devnet-src/webex-teams$ python3 authentication.py
{
    "id": "Y2lz...UyM2U",
    "emails": [
        "your-email@example.com"
    ],
    "phoneNumbers": [],
    "displayName": "Your-First-Name Your-Last-Name",
    "nickName": "Your-Nick-Name",
    "firstName": " Your-First-Name",
    "lastName": "Your-Last-Name",
    "avatar": "https://9643-417f-9974...6baa4-1600",
    "orgId": "Y2lzY2...UxMGY",
    "created": "2012-06-15T20:23:12.529Z",
    "lastActivity": "2020-06-02T20:16:52.111Z",
    "status": "active",
    "type": "person"
}
devasc@labvm:~/labs/devnet-src/webex-teams$
```

Part 4: Manage People in Webex Teams

In Webex Teams, People are registered users. Through the API, you can retrieve a list of people, you can create a person, retrieve an individual person's details, update a person, and delete a person.

Step 1: Locate the API documentation for listing details of a registered Webex Teams user.

a. Return to the developer.webex.com website. Under **API Reference** > **People**, click the **Method** for **List People**.

b. Under **Query Parameters**, find the **email** parameter. This is the parameter you will use to find a specific user in your organization. Alternatively, you could use the **displayName** parameter, if you know the exact name. You can use the Try it feature.

Step 2: Use a Python script to list details of a registered Webex Teams user.

a. In VS Code, click the **list-people.py** file.

b. Place the following code into the file. Be sure to replace **your_token_here** with your personal access token and **user@example**.com with an actual registered Webex Team user in your organization.

```
import requests
import json

access_token = 'your_token_here'
```

```
url = 'https://webexapis.com/v1/people'
headers = {
    'Authorization': 'Bearer {}'.format(access_token),
    'Content-Type': 'application/json'
}
params = {
    'email': 'user@example.com'
}
res = requests.get(url, headers=headers, params=params)
print(json.dumps(res.json(), indent=4))
```

c. Save and run the script. You should get the same output similar to the following. If you get a message like, **{'message': 'Invalid email address.'...**, it means that you did not replace the empty email parameter with a legitimate email address for a registered Webex Teams user. The value for the **id** key will be used in the next API call.

Note: The values for some of the keys have been truncated in the output below.

```
devasc@labvm:~/labs/devnet-src/webex-teams$ python3 list-people.py
{
    "notFoundIds": null,
    "items": [
        {
            "id": "Y2l...2I", # You will use this value in the next step
            "emails": [
                "user@example.com"
            ],
            "phoneNumbers": [
                {
                    "type": "mobile",
                    "value": "+1234567690"
                }
            ],
            "displayName": "displayName",
            "nickName": "nickName",
            "firstName": "firstName",
            "lastName": "lastName",
            "avatar": "https://9643-417f-9974...6baa4-1600",
            "orgId": "Y2lzY...UxMGY",
            "created": "2012-06-15T20:39:19.726Z",
            "lastActivity": "2020-06-04T13:57:01.688Z",
            "status": "active",
            "type": "person"
        }
    ]
}
devasc@labvm:~/labs/devnet-src/webex-teams$
```

Step 3: List additional administrative details for a person.

a. If you are a Webex Teams administrator, you can get additional details for a person by using the value of the person **id** key in your API call. Add the following code to your **list-people.py** script. Replace **previous_id_here** with value for **id** from the previous API call.

```
person_id = 'previous_id_here'
url = 'https://webexapis.com/v1/people/{}'.format(person_id)
headers = {
    'Authorization': 'Bearer {}'.format(access_token),
    'Content-Type': 'application/json'
}
res = requests.get(url, headers=headers)
print(json.dumps(res.json(), indent=4))
```

b. Save the file and run it. As a non-administrator, you will get information that is very similar to the previous step.

```
devasc@labvm:~/labs/devnet-src/webex-teams$ python3 list-people.py
{
    <first API call output omitted>
}
{
    "id": "Y2l...2I",
    "emails": [
        "user@example.com"
    ],
    "phoneNumbers": [
        {
            "type": "mobile",
            "value": "+1234567890"
        }
    ],
    "displayName": "displayName",
    "nickName": "nickName",
    "firstName": "firstName",
    "lastName": "lastName",
    "avatar": "https://9643-417f-9974...6baa4-1600",
    "orgId": "Y2l...MGY",
    "created": "2012-06-15T20:39:19.726Z",
    "lastActivity": "2020-06-04T14:39:36.535Z",
    "status": "active",
    "type": "person"
}
devasc@labvm:~/labs/devnet-src/webex-teams$
```

Part 5: Manage Rooms in Webex Teams

Rooms, also called spaces in the user interface, let people send messages and files to collaborate virtually in collective meeting places. In this Part, you will list rooms, create a room, and get a room's details.

Step 1: Locate and investigate the API documentation for rooms.

a. Return to the developer.webex.com website. Under **API Reference**, click **Rooms**.

b. Explore the various API calls you can make with the **Rooms** API.

c. Click the GET request for **List Rooms** and explore the **Query Parameters**.

Step 2: Use a Python script to list all the rooms for an authenticated user.

a. For this step, you will need to be a member of at least one room. A conversation with one other person is considered a room in Webex Teams.

b. In VS Code, click the **list-rooms.py** file.

c. Place the following code into the file. Be sure to replace **your_token_here** with your personal access token.

```
import requests

access_token = 'your_token_here'
url = 'https://webexapis.com/v1/rooms'
headers = {
    'Authorization': 'Bearer {}'.format(access_token),
    'Content-Type': 'application/json'
}
params={'max': '100'}
res = requests.get(url, headers=headers, params=params)
print(res.json())
```

d. Save and run the file. Your output will be different than the following. Only one room is listed here. ID values have been truncated.

```
devasc@labvm:~/labs/devnet-src/webex-teams$ python3 list-rooms.py
{'items': [{'id': 'Y2l...ZTE0', 'title': 'User Name', 'type': 'direct', 'isLocked':
False, 'lastActivity': '2020-06-01T16:34:56.536Z', 'creatorId': 'Y2lz...yM2U',
'created': '2020-06-01T16:30:21.816Z', 'ownerId': 'Y2lz...xMGY'}
# additional rooms displayed up to 'max' value.
]}
devasc@labvm:~/labs/devnet-src/webex-teams$
```

Step 3: Locate and investigate the API documentation for posting to the Rooms API.

a. Return to the developer.webex.com website. Under **API Reference**, click **Rooms**, if necessary.

b. The Rooms API has one POST method for **Create a Room**. Click the link see **Query Parameters** that are available. In your script, you will use the required **title** parameter.

Step 4: Use a Python script to create a room.

a. In VS Code, click the **create-rooms.py** file.

b. Place the following code into the file. Be sure to replace **your_token_here** with your personal access token. Notice that this is a POST request and uses the **title** parameter.

```
import requests

access_token = 'your_token_here'
url = 'https://webexapis.com/v1/rooms'
headers = {
    'Authorization': 'Bearer {}'.format(access_token),
    'Content-Type': 'application/json'
}
```

```
params={'title': 'DevNet Associate Training!'}
res = requests.post(url, headers=headers, json=params)
print(res.json())
```

c. Save and run the file. You should get a response similar to the following. ID values have been truncated. The room **id** and **title** are highlighted. Copy the room ID and save it in a text file for use in the rest of this lab.

```
devasc@labvm:~/labs/devnet-src/webex-teams$ python3 create-rooms.py
{'id': 'Y2l...GNm', 'title': 'DevNet Associate Training!', 'type': 'group',
'isLocked': False, 'lastActivity': '2020-06-04T16:50:19.371Z', 'creatorId':
'Y2l...M2U', 'created': '2020-06-04T16:50:19.371Z', 'ownerId': 'Y2l...MGY'}
devasc@labvm:~/labs/devnet-src/webex-teams$
```

d. In your Webex Teams application, verify you now see the **DevNet Associate Training!** room. You are currently the only member.

Step 5: Use a Python script to get room details.

What if you want to start a meeting in your new room? You can do another GET call to retrieve the Session Initiation Protocol (SIP) address, the meeting URL, and the dial-in phone numbers.

a. In VS Code, click the **get-room-details.py** file.

b. Place the following code into the file. Replace **your_token_here** with your personal access token. Replace **your_room_id** with the value you got in the previous step.

```
import requests

access_token = 'your_token_here'
room_id = 'your_room_id'
url = 'https://webexapis.com/v1/rooms/{}/meetingInfo'.format(room_id)
headers = {
    'Authorization': 'Bearer {}'.format(access_token),
    'Content-Type': 'application/json'
}
res = requests.get(url, headers=headers)
print(res.json())
```

c. Save and run the file. You should get a response similar to the following. Values have been truncated.

```
devasc@labvm:~/labs/devnet-src/webex-teams$ python3 get-room-details.py
{'roomId': 'Y2l...GNm', 'meetingLink': 'https://cisco.webex.com/m/3272...a837',
'sipAddress': '162...468@cisco.webex.com', 'meetingNumber': '162...0468',
'callInTollFreeNumber': '+1-866-...-9903', 'callInTollNumber': '+1-408-...-6800'}
devasc@labvm:~/labs/devnet-src/webex-teams$
```

Part 6: Manage Memberships in Webex Teams

In this Part, you will use the Membership API to add someone to your room.

Step 1: Locate and investigate the API documentation for memberships.

a. Return to the developer.webex.com website. Under **API Reference**, click **Memberships**.

b. Explore the various API calls you can make with the **Memberships** API.

c. Click the GET request for **List Memberships** and explore the **Query Parameters**.

Step 2: Use a Python script to list the members of the room.

a. In VS Code, click the **list-memberships.py** file.

b. Place the following code into the file. Replace **your_token_here** with your personal access token. Replace **your_room_id** with the value from you got in the previous part.

```
import requests

access_token = 'your_token_here'
room_id = 'your_room_id'
url = 'https://webexapis.com/v1/memberships'
headers = {
    'Authorization': 'Bearer {}'.format(access_token),
    'Content-Type': 'application/json'
}
params = {'roomId': room_id}
res = requests.get(url, headers=headers, params=params)
print(res.json())
```

c. Save and run the file. You should get a response similar to the following. You should be the only member unless you have added someone in the Webex Teams application. ID values have been truncated.

```
devasc@labvm:~/labs/devnet-src/webex-teams$ python3 list-memberships.py
{'items': [{'id': 'Y2l...RjZg', 'roomId': 'Y2l...GNm', 'personId': 'Y2l...M2U',
'personEmail': 'user@example.com', 'personDisplayName': 'personDisplayName',
'personOrgId': 'Y2l...MGY', 'isModerator': False, 'isMonitor': False, 'created':
'2020-06-04T16:50:19.819Z'}]}
devasc@labvm:~/labs/devnet-src/webex-teams$
```

Step 3: Locate and investigate the API documentation for posting to the Memberships API.

a. Return to the developer.webex.com website. Under **API Reference**, click **Memberships**, if necessary.

b. The Memberships API has one POST method for **Create a Membership**. Click the link see **Query Parameters** that are available. In your script, you will use the required **roomID** and **personEmail** parameters.

Step 4: Use a Python script to create a membership, adding someone to the room.

For this step, you will need the email of someone else who is a registered Webex Teams user in your organization. You can use the same email you used previously to list details about a person.

a. In VS Code, click the **create-membership.py** file.

b. Place the following code into the file. Replace **your_token_here** with your personal access token. Replace **your_room_id** with the value from you got in the previous part. Replace **new-user@example.com**w with the email of the person you want to add to the room.

```
import requests

access_token = 'your_token_here'
room_id = 'your_room_id'
person_email = 'new-user@example.com'
url = 'https://webexapis.com/v1/memberships'
headers = {
    'Authorization': 'Bearer {}'.format(access_token),
    'Content-Type': 'application/json'
```

```
}
params = {'roomId': room_id, 'personEmail': person_email}
res = requests.post(url, headers=headers, json=params)
print(res.json())
```

c. Save and run the file. You should get a response similar to the following. You should be the only member unless you have added someone in the Webex Teams application. ID values have been truncated.

```
devasc@labvm:~/labs/devnet-src/webex-teams$ python3 list-memberships.py
{'items': [{'id': 'Y2l...RjZg', 'roomId': 'Y2l...GNm', 'personId': 'Y2l...M2U',
'personEmail': 'user@example.com', 'personDisplayName': 'personDisplayName',
'personOrgId': 'Y2l...MGY', 'isModerator': False, 'isMonitor': False, 'created':
'2020-06-04T16:50:19.819Z'}]}
devasc@labvm:~/labs/devnet-src/webex-teams$
```

Part 7: Manage Messages in Webex Teams

In Webex Teams, a message can contain plain text, Markdown, or a file attachment. Each message is displayed on its own line along with a timestamp and sender information. You can use this Messages API to list, create, and delete messages. In this Part, you will send a message to the room you created in this lab.

Step 1: Locate and investigate the API documentation for messages.

a. Return to the developer.webex.com website. Under **API Reference**, click **Messages**.

b. Explore the various API calls you can make with the **Messages** API.

c. Click the POST request for **Create a Message** and explore the **Query Parameters**. Notice for a simple text message, you can use the **text** or **markdown** parameter. In this step, you will specify a message with Markdown formatting. Search the internet to learn more about Markdown.

Step 2: Use a Python script to send a message to a Webex room.

a. In VS Code, click the **creat-markdown-message.py** file.

b. Place the following code into the file. Replace **your_token_here** with your personal access token. Replace **your_room_id** with the value from you got in the previous part.

```
import requests

access_token = 'your_token_here'
room_id = 'your_room_id'
message = 'Hello **DevNet Associates**!!'
url = 'https://webexapis.com/v1/messages'
headers = {
    'Authorization': 'Bearer {}'.format(access_token),
    'Content-Type': 'application/json'
}
params = {'roomId': room_id, 'markdown': message}
res = requests.post(url, headers=headers, json=params)
print(res.json())
```

c. Save and run the file. You should get a response similar to the following. Notice that the Markdown was converted to HTML. ID values have been truncated.

```
devasc@labvm:~/labs/devnet-src/webex-teams$ python3 create-markdown-
message.py
{'id': 'Y2l...RjZg', 'roomId': 'Y2l...GNm', 'roomType': 'group', 'text': 'Hello DevNet
Associates!!', 'personId': 'Y2l...M2U', 'personEmail': 'user@example.com', 'markdown':
```

```
'Hello **DevNet Associates**!!', 'html': '<p>Hello <strong>DevNet
Associates</strong>!!</p>', 'created': '2020-06-04T19:08:49.145Z'}
devasc@labvm:~/labs/devnet-src/webex-teams$
```