Table of Contents

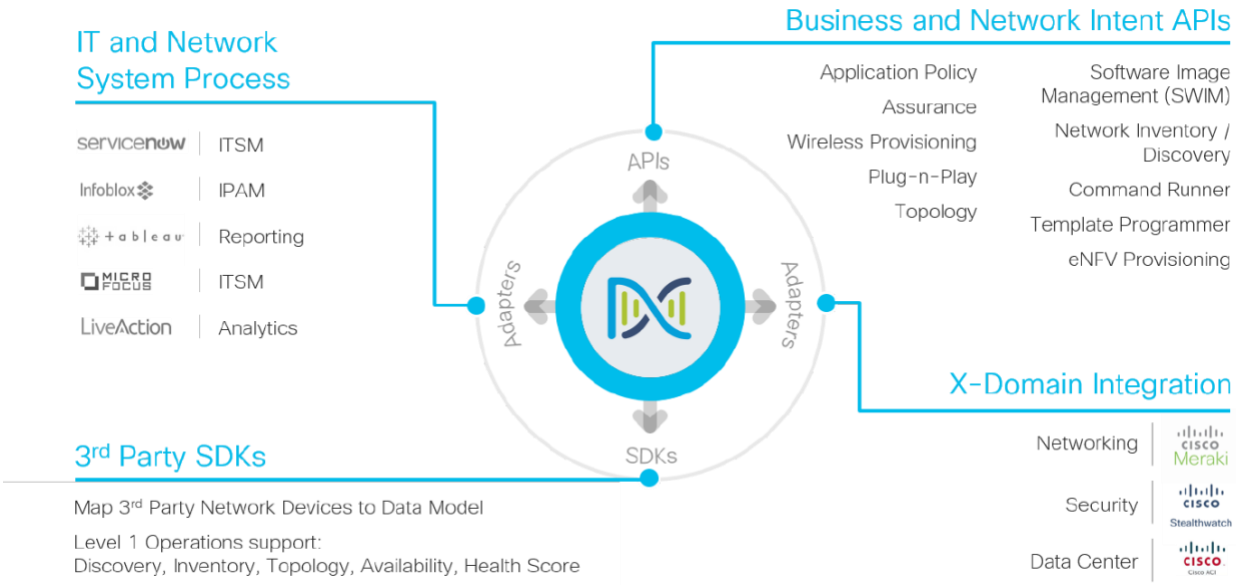
Lab 01: Cisco DNA Center Platform – Authentication

https://developer.cisco.com/learning/tracks/programming-dna/dnac-rest-apis/dnac-101-auth/step/1

Cisco DNA Center Platform - Authentication

Cisco DNA Center is at the heart of Cisco's new era of networking, the intent-based network.

Cisco DNA Center supports the expression of business intent for network use cases, including base automation capabilities in an enterprise network. The Analytics and Assurance features of Cisco DNA Center provide full context, end-to-end visibility into the network through data and insights.



Objectives

When you have completed this lab, you will be able to:

- Authenticate and retrieve a token from Cisco DNA Center.
- Build an authentication python function.

Prerequisites

- Install Postman
- Setup Python

DevNet Sandbox Details

- **Url**: https://sandboxdnac.cisco.com/
- **Username**: devnetuser
- **Password**: Cisco123!

Authentication API by Authorization string

Cisco DNA Center APIs use token-based authentication and HTTPS Basic Authentication to generate an authentication cookie and security token that is used to authorize subsequent requests.

HTTPS Basic uses Transport Layer Security (TLS) to encrypt the connection and data in an HTTP Basic Authentication transaction.

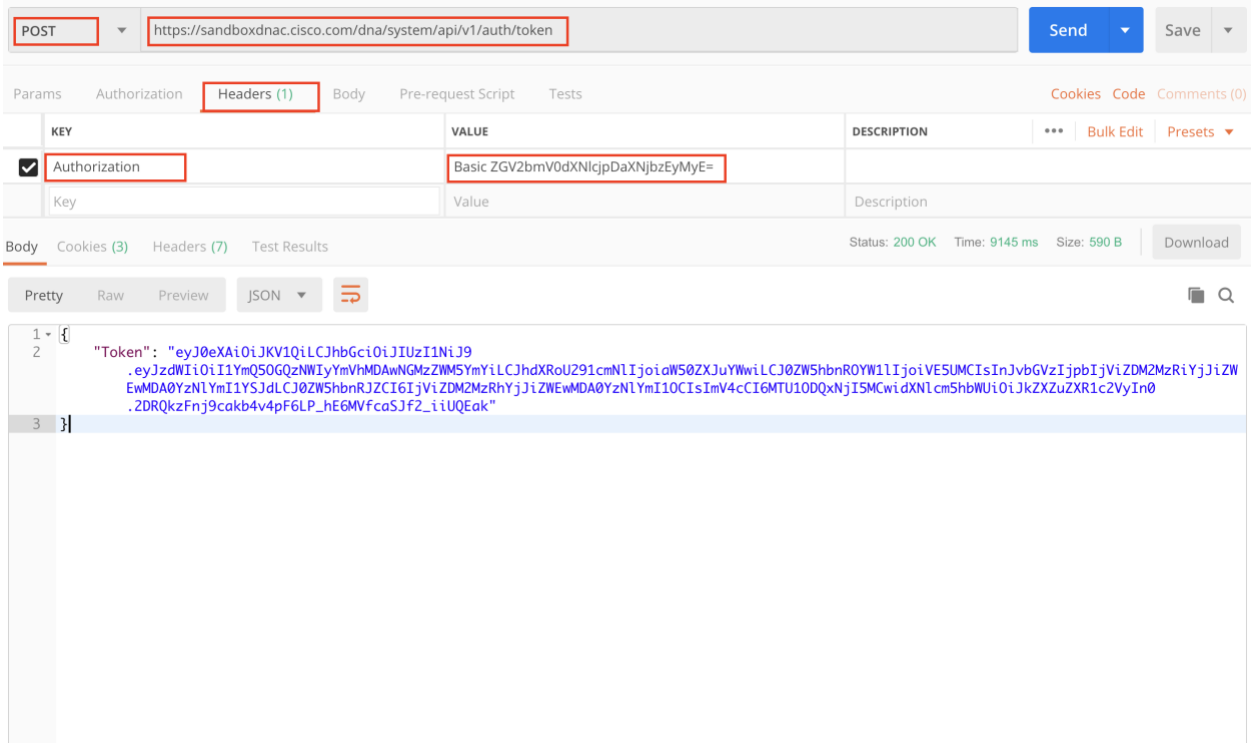
Follow this procedure to set up the request in Postman:

1. Launch Postman and locate the center pane. Here you have the option to select a method for your request. Select **POST** from the drop-down list.
2. In the Request URL section, enter:

```
https://sandboxdnac.cisco.com/dna/system/api/v1/auth/token
```

3. On the Headers tab, create an Authorization key with the value Basic ZGV2bmV0dXNlcjpDaXNjbzEyMyE=

   Note: the authorization value is a Basic Auth Base64 encoding of the username and password provided in the DevNet Sandbox Details section.
4. Click **Send**.



You have successfully made your first Cisco DNAC API Call!

You've just requested a new Token. To authorize subsequent requests, pass the token as the value of the **X-Auth-Token** header of the request. Keep this handy! Later in this learning lab, you'll use this token to send several requests.

Authentication API by Username, password

Cisco DNA Center APIs use token-based authentication and HTTPS Basic Authentication to generate an authentication cookie and security token that is used to authorize subsequent requests.

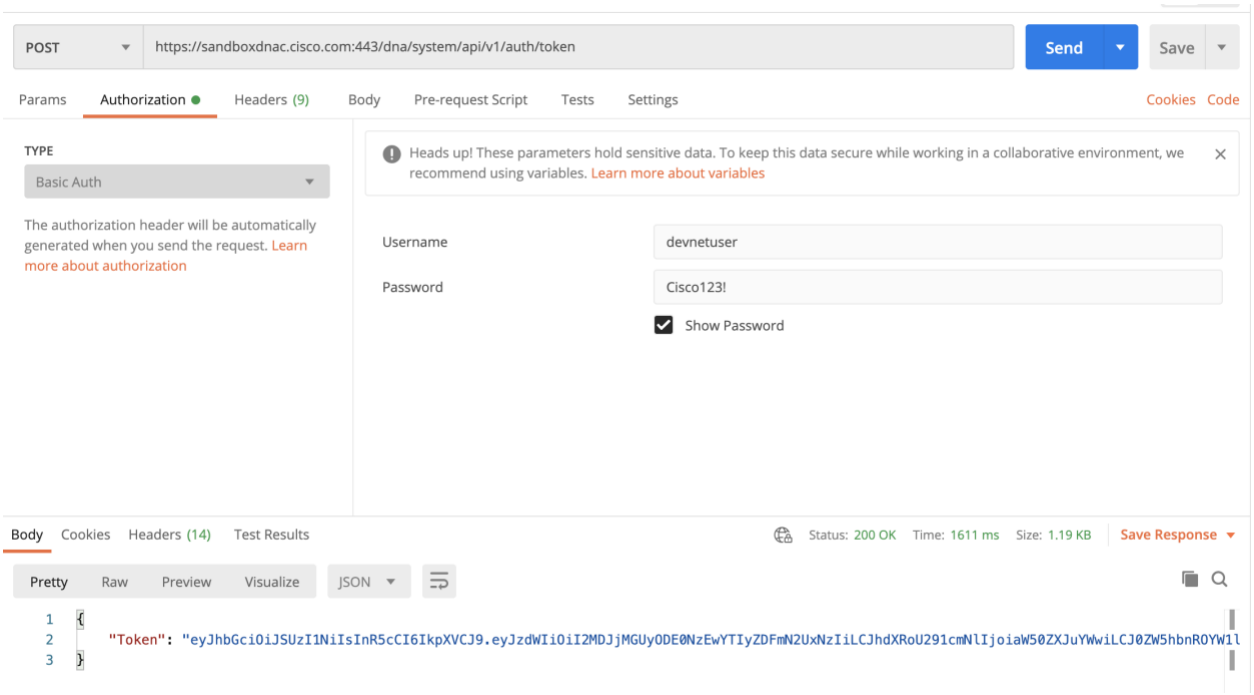
HTTPS Basic uses Transport Layer Security (TLS) to encrypt the connection and data in an HTTP Basic Authentication transaction.

Follow this procedure to set up the request in Postman:

1. Launch Postman and locate the center pane. Here you have the option to select a method for your request. Select **POST** from the drop-down list.
2. In the Request URL section, enter:

https://sandboxdnac.cisco.com/dna/**system**/api/v1/auth/**token**

3. On the Headers Authorization tab, choose Basic Auth and fill in username, password



You've just requested a new Token. To authorize subsequent requests, pass the token as the value of the **X-Auth-Token** header of the request. Keep this handy! Later in this learning lab, you'll use this token to send several requests.

Lab 02: Cisco DNA Center Platform - Network Devices

https://developer.cisco.com/learning/tracks/programming-dna/dnac-rest-apis/dnac-101-network_device/step/1

Cisco DNA Center - Network Devices

Know Your Network REST request paths can retrieve details about clients, sites, topology and devices, add devices to the network and export device data.

Objectives

When you have completed this Learning Lab, you will be able to:

- Pull network device list from your network.
- Build a Python function and parse the data.

Prerequisites

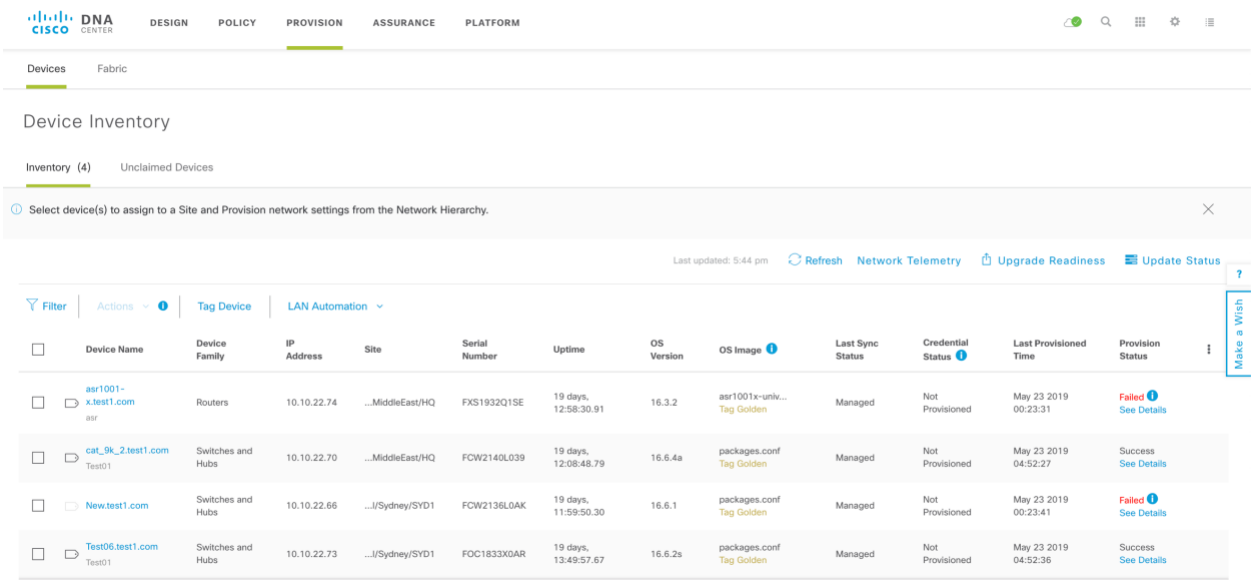
- Install Postman
- Setup Python

Retrieving a list of network devices with Postman

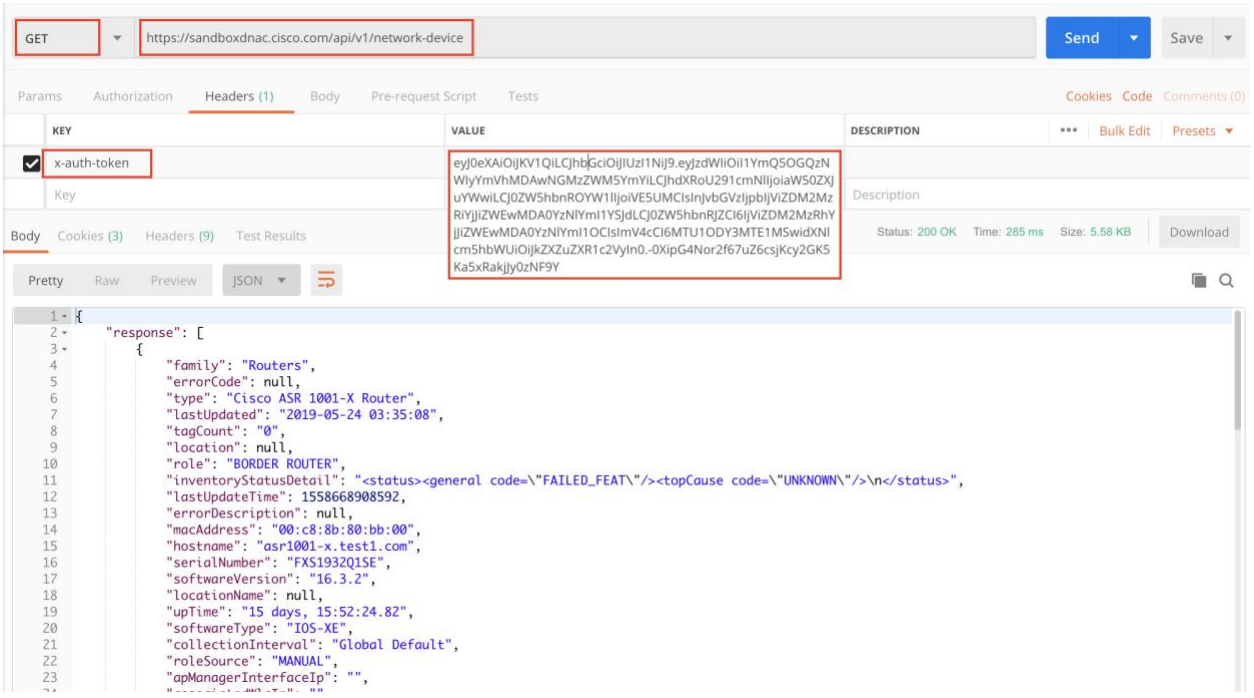
The Network Device Detail Intent API retrieves detailed information about devices by timestamp, MAC address, UUID, name, or nwDeviceName.

Additional REST request paths allow you to retrieve additional information, such as functional capabilities, interfaces, device config, certificate validation status, values of specified fields, modules, and VLAN data associated with specified interfaces. You can also add, delete, update or sync specified devices.

Let's look at how we can pull a network device list that we can tie to an asset management system.



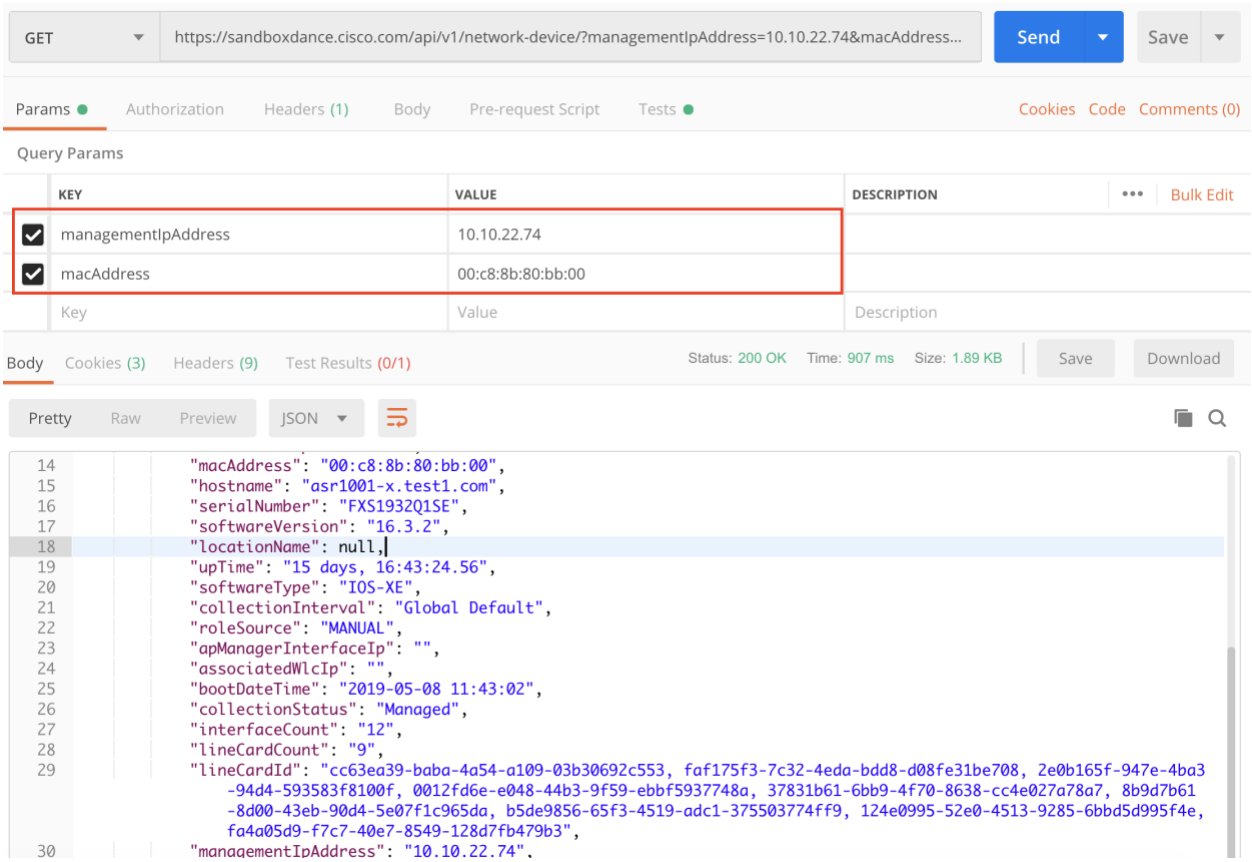
1. Launch Postman and locate the center pane. Here you have the option to select a method for your request. Select **GET** from the drop-down list.
2. In the Request URL section, enter https://sandboxdnac.cisco.com/api/v1/network-device.
3. On the Headers tab, create an x-auth-token key. For the value, use the token that you generated in the Learning Lab **Cisco DNA Center Platform - Authentication**.
4. Click **Send**.




```
{
    "response": [
        {
            "family": "Routers",
            "errorCode": null,
            "type": "Cisco ASR 1001-X Router",
            "lastUpdated": "2019-05-24 03:35:08",
            "tagCount": "0",
            "location": null,
            "role": "BORDER ROUTER",
            "inventoryStatusDetail": "<status><general code=\"FAILED_FEAT\"/><topCause code=\"UNKNOWN\"/>\n</status>",
            "lastUpdateTime": 1558668908592,
            "errorDescription": null,
            "macAddress": "00:c8:8b:80:bb:00",
            "hostname": "asr1001-x.test1.com",
            "serialNumber": "FXS1932Q1SE",
            "softwareVersion": "16.3.2",
            "locationName": null,
            "upTime": "15 days, 15:52:24.82",
            "softwareType": "IOS-XE",
            "collectionInterval": "Global Default",
            "roleSource": "MANUAL",
            "apManagerInterfaceIp": "",
```


Checking specific devices

If you know the IP address or MAC address of a device and only need to check that specific device, you can use the query parameters managementIpAddress or macAddress, with the appropriate value for the device.



Next steps

You have successfully pulled the device list. but this is static, and we need to programmatically pull the list of network devices the controller manages!

Creating a network device list with a Python function

Let's use the Python *requests* library to create a function that when called upon, will return and display the list of devices managed by the controller.

1. The first part of the function will import the required libraries.
    - **requests** is the library of choice to make the api request.
    - **HTTPBasicAuth** is part of the requests library and is used to encode the credentials to Cisco DNA-C
    - **dnac_config** is a Python file that contains Cisco DNA Center configuration info. In this case we are using our DevNet Sandbox

```
import requests
from requests.auth import HTTPBasicAuth
from dnac_config import DNAC, DNAC_PORT, DNAC_USER, DNAC_PASSWORD
```

5. You will now define the function and write the GET request.

```
token = get_auth_token() # Get a Token
url = "https://sandboxdnac.cisco.com/api/v1/network-device" #Network Device endpoint
hdr = {'x-auth-token': token, 'content-type' : 'application/json'} #Build header Info
resp = requests.get(url, headers=hdr)  # Make the Get Request
device_list = resp.json() #capture the data from the controller
print_device_list(device_list) #pretty print the data we want
```

Note print_device_list() and get_auth_token() functions are included part of the complete code.

12. Apply a filter to the data and look for a specific device by creating a query string variable queryString and passing the variable part of the params parameter in your requests.get call.

```
token = get_auth_token() # Get a Token
url = "https://sandboxdnac.cisco.com/api/v1/network-device" #Network Device endpoint
hdr = {'x-auth-token': token, 'content-type' : 'application/json'} #Build header Info
querystring = {"macAddress":"00:c8:8b:80:bb:00","managementIpAddress":"10.10.22.74"}
resp = requests.get(url, headers=hdr, params=querystring)  # Make the Get Request
device_list = resp.json() # Capture data from the controller
print_device_list(device_list) # Pretty print the data
```

20. Execute the code.

```
if __name__ == "__main__":
  get_device_list()
```

If the code is correct, it will generate output similar to the following.

```
hostname                    mgmt IP       serial      platformId      SW Version  role          U
ptime
asr1001-x.test1.com         10.10.22.74   FXS1932Q1SE ASR1001-X       16.3.2      B
ORDER ROUTER   16 days, 16:57:20.36
cat_9k_2.test1.com          10.10.22.70   FCW2140L039 C9300-24UX      16.6.4a     A
CCESS          16 days, 16:10:33.49
New.test1.com               10.10.22.66   FCW2136L0AK C9300-24UX      16.6.1      A
CCESS          16 days, 16:17:26.27
Test06.test1.com            10.10.22.73   FOC1833X0AR WS-C3850-48U-E  16.6.2s
DISTRIBUTION   16 days, 18:08:47.29
```

Congratulations! You have successfully pulled a list of network devices (or a specific subset of devices) and you are ready to integrate the information into the asset management tool of your choice.

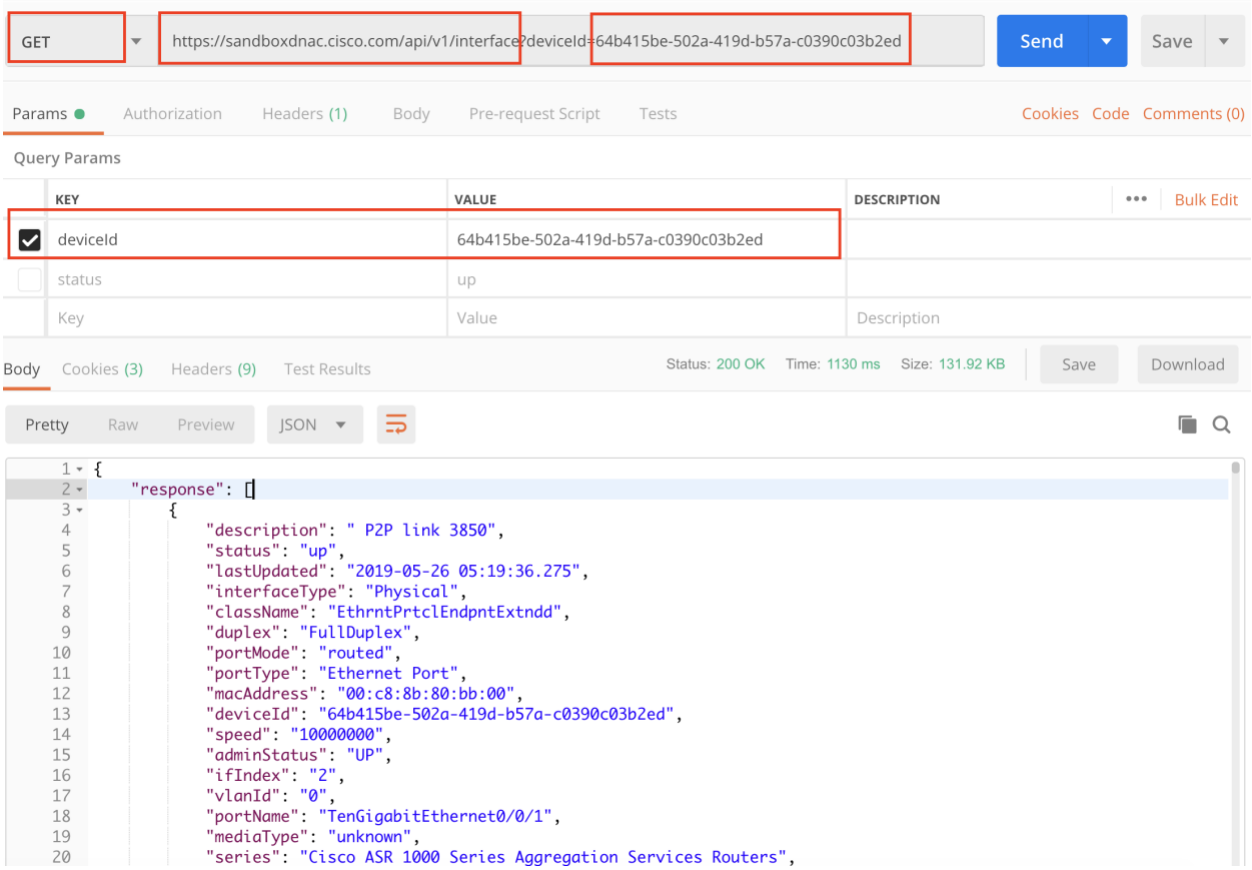
Complete Code

Click below to view the complete script

Retrieving device interface information with Postman

Now that we know the list of devices the controller is managing, let's pull the device interface information. This can be used to audit your network, automate a task to apply port security and create policies based on usage.

1. Launch Postman and locate the center pane. Here you have the option to select a method for your request. Select **GET** from the drop-down list.
2. In the Request URL section, enter https://sandboxdnac.cisco.com/api/v1/interface.
3. On the Headers tab, create an x-auth-token key. For the value, use the token that you generated in the Learning Lab **Cisco DNA Center Platform - Authentication**.
4. On the Params tab, create a deviceId key. For the value, use a device ID that you pulled in the previous section of this Learning Lab.



Next: Creating a list of network device interfaces with a Python function

Creating a list of network device interfaces with a Python function

Let's use the Python *requests* library to create a function that when called upon returns a list of all interfaces to a specific device.

1. The first part of the function will import the required libraries.
    - **requests** is the library of choice to make the API request.
    - **dnac_config** is a Python file that contains Cisco DNA Center configuration info. In this case we are using our DevNet Sandbox

```
2. import requests
3. from dnac_config import DNAC, DNAC_PORT, DNAC_USER, DNAC_PASSWORD
```

2. You will now define the function and write the GET request. This builds on the function that you wrote earlier in this Learning Lab for retrieving a list of devices.

```
3. def get_device_int(device_id):
```

```
    """
    Building out function to retrieve device interface. Using requests.get
    to make a call to the network device Endpoint
    """
    url = "https://sandboxdnac.cisco.com/api/v1/interface"
    hdr = {'x-auth-token': token, 'content-type' : 'application/json'}
    querystring = {"macAddress": device_id} # Dynamically build the query params to get device-specific Interface info
    resp = requests.get(url, headers=hdr, params=querystring) # Make the Get Request
    interface_info_json = resp.json()
    print_interface_info(interface_info_json)
```

Note print_interface_info() function is part of the complete code sample found here. If the code is correct, it will generate output similar to the following.

```
portName                vlanId  portMode  portType        duplex      status  lastUpdated
TenGigabitEthernet0/0/1 0       routed    Ethernet Port   FullDuplex  up      2019-05-26 06:36:07.25
GigabitEthernet0/0/4    0       routed    Ethernet Port   FullDuplex  down    2019-05-26 06:36:07.25
VoIP-Null0              0       routed    OTHER           None        up      2019-05-26 06:36:07.25
GigabitEthernet0/0/2    0       routed    Ethernet Port   FullDuplex  down    2019-05-26 06:36:07.25
GigabitEthernet0/0/3    0       routed    Ethernet Port   FullDuplex  down    2019-05-26 06:36:07.25
```

Congratulations! You have successfully pulled a list of network devices and interfaces.

Complete Code

Click below to view the complete script

Congratulations! You have completed Cisco DNA Center Platform - Network Devices.