Table of Contents

[Lab 01: Cisco DNA Center Platform – Authentication 2](#_Toc102307000)

[Cisco DNA Center Platform - Authentication 2](#_Toc102307001)

[Objectives 2](#_Toc102307002)

[Prerequisites 2](#_Toc102307003)

[DevNet Sandbox Details 3](#_Toc102307004)

[Authentication API by Authorization string 3](#_Toc102307005)

[Authentication API by Username, password 4](#_Toc102307006)

[Lab 02: Cisco DNA Center Platform - Network Devices 6](#_Toc102307007)

[Cisco DNA Center - Network Devices 6](#_Toc102307008)

[Objectives 6](#_Toc102307009)

[Prerequisites 6](#_Toc102307010)

[Retrieving a list of network devices with Postman 6](#_Toc102307011)

[Checking specific devices 7](#_Toc102307012)

[Creating a network device list with a Python function 8](#_Toc102307013)

[Complete Code 10](#_Toc102307014)

[Retrieving device interface information with Postman 10](#_Toc102307015)

[Creating a list of network device interfaces with a Python function 11](#_Toc102307016)

[Complete Code 12](#_Toc102307017)

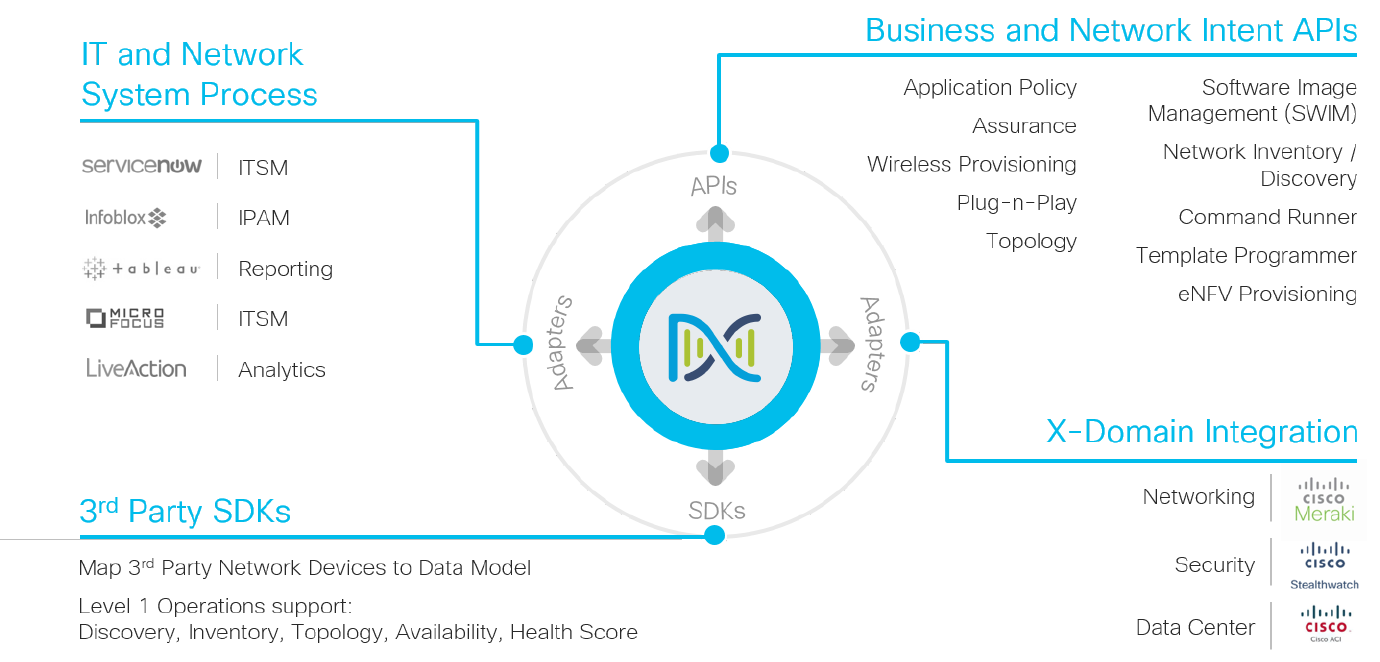
# Lab 01: Cisco DNA Center Platform – Authentication

<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](https://www.getpostman.com/)
* [Setup Python](https://www.python.org/downloads/)

**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:

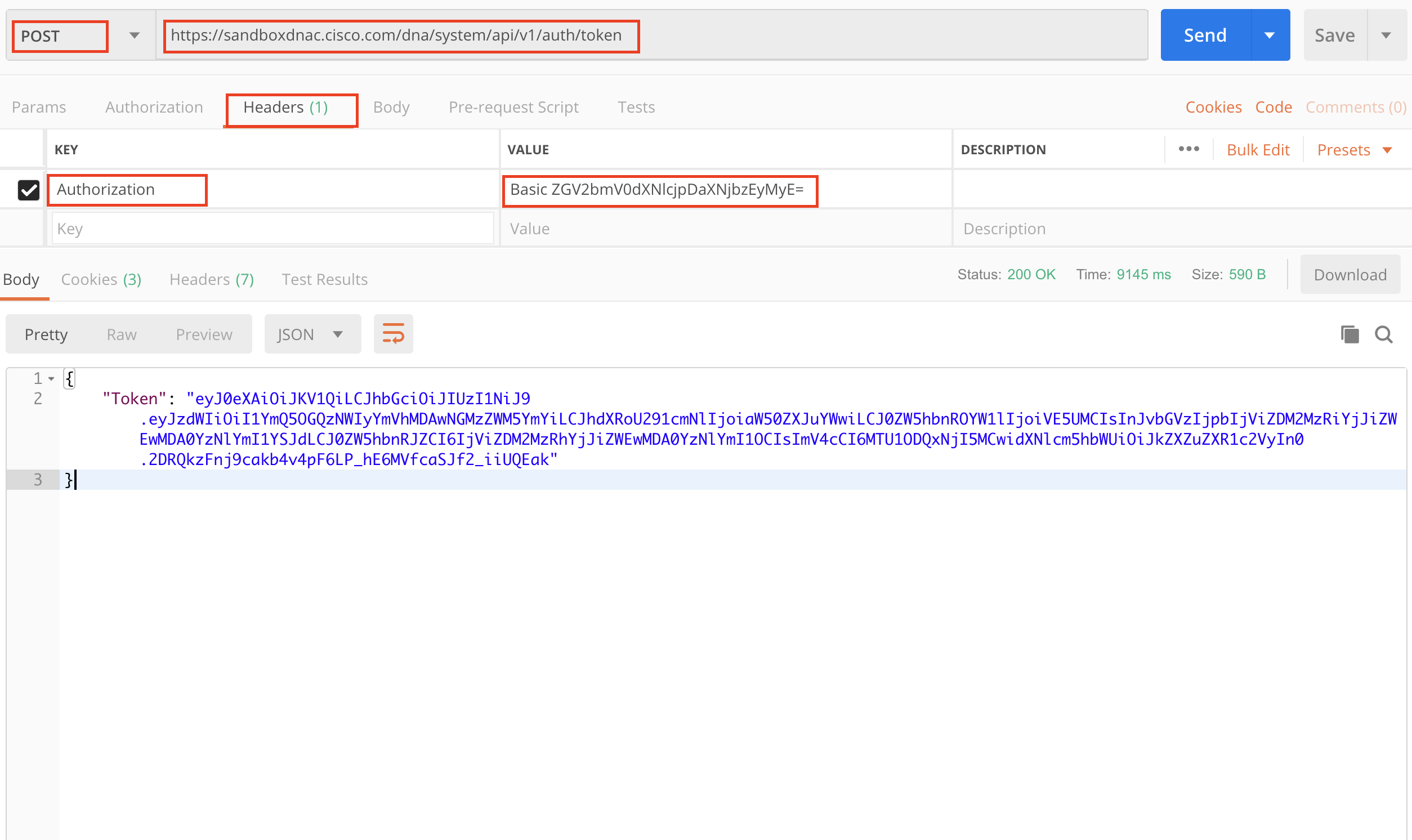
1. Launch Postman and locate the center pane. Here you have the option to select a method for your request. Select **POST** from the drop-down list.
2. In the Request URL section, enter:

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

1. 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.

1. 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.

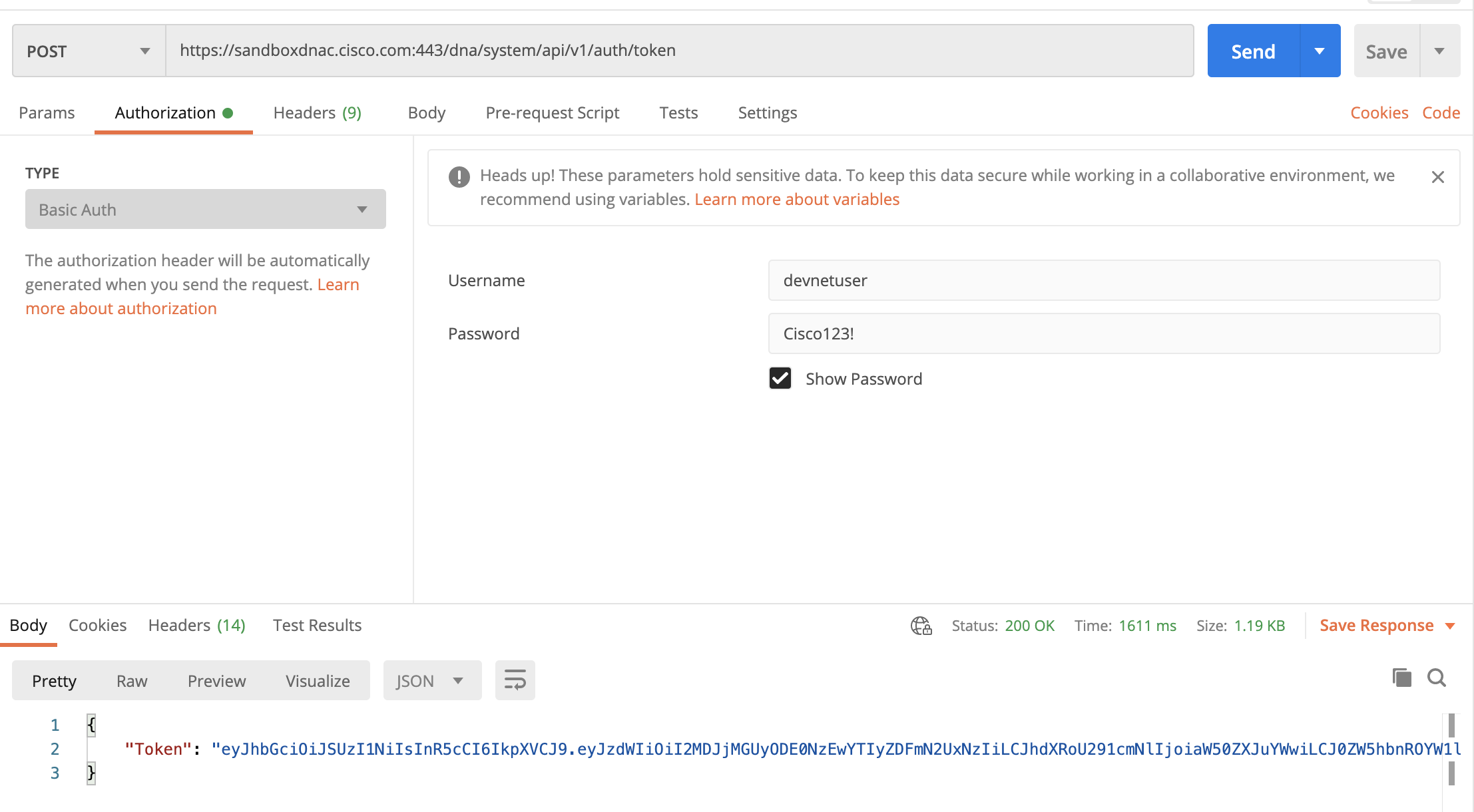
HTTPS Basic uses Transport Layer Security (TLS) to encrypt the connection and data in an HTTP Basic Authentication transaction.

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**

1. 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

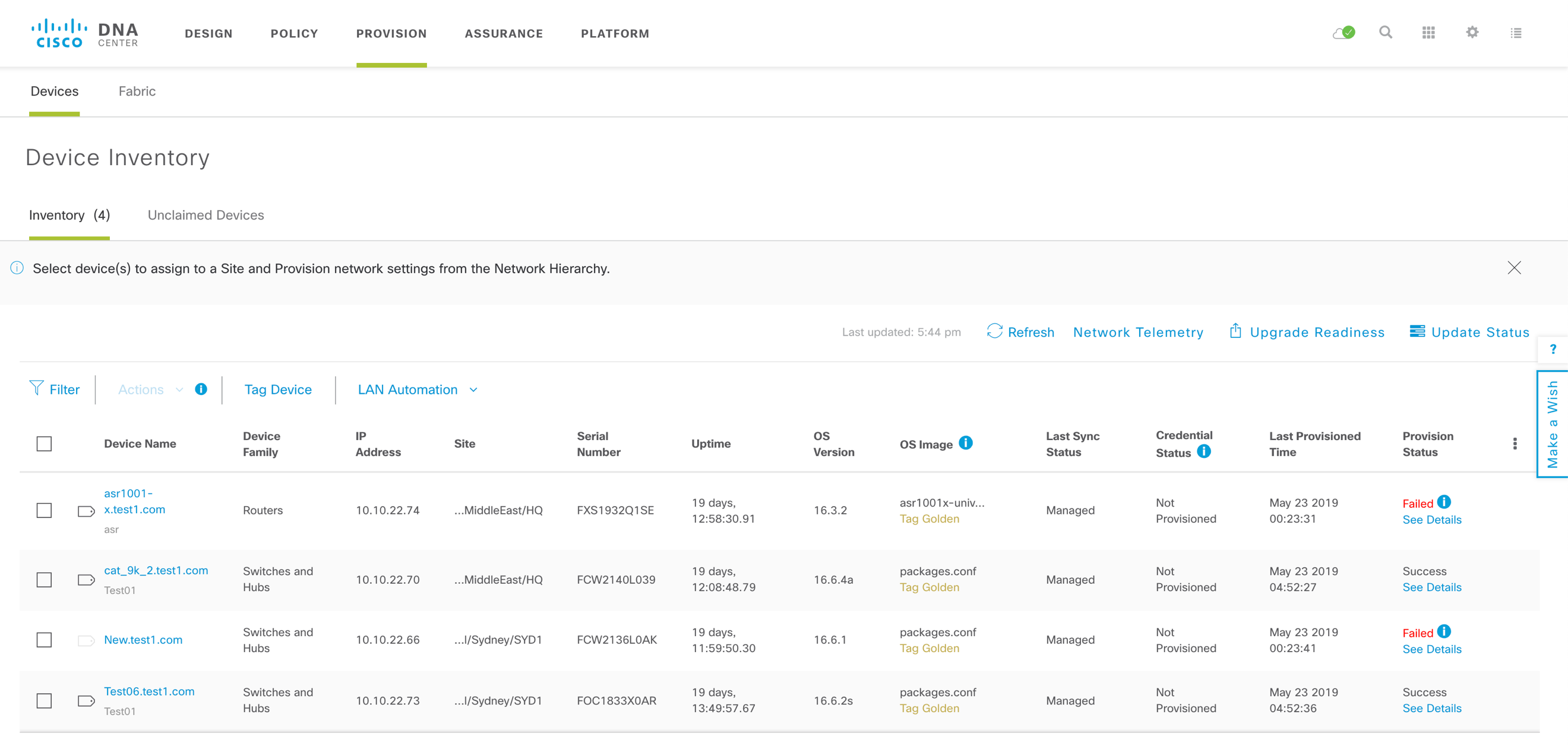
* [Install Postman](https://www.getpostman.com/)
* [Setup Python](https://www.python.org/downloads/)

# Retrieving a list of network devices with Postman

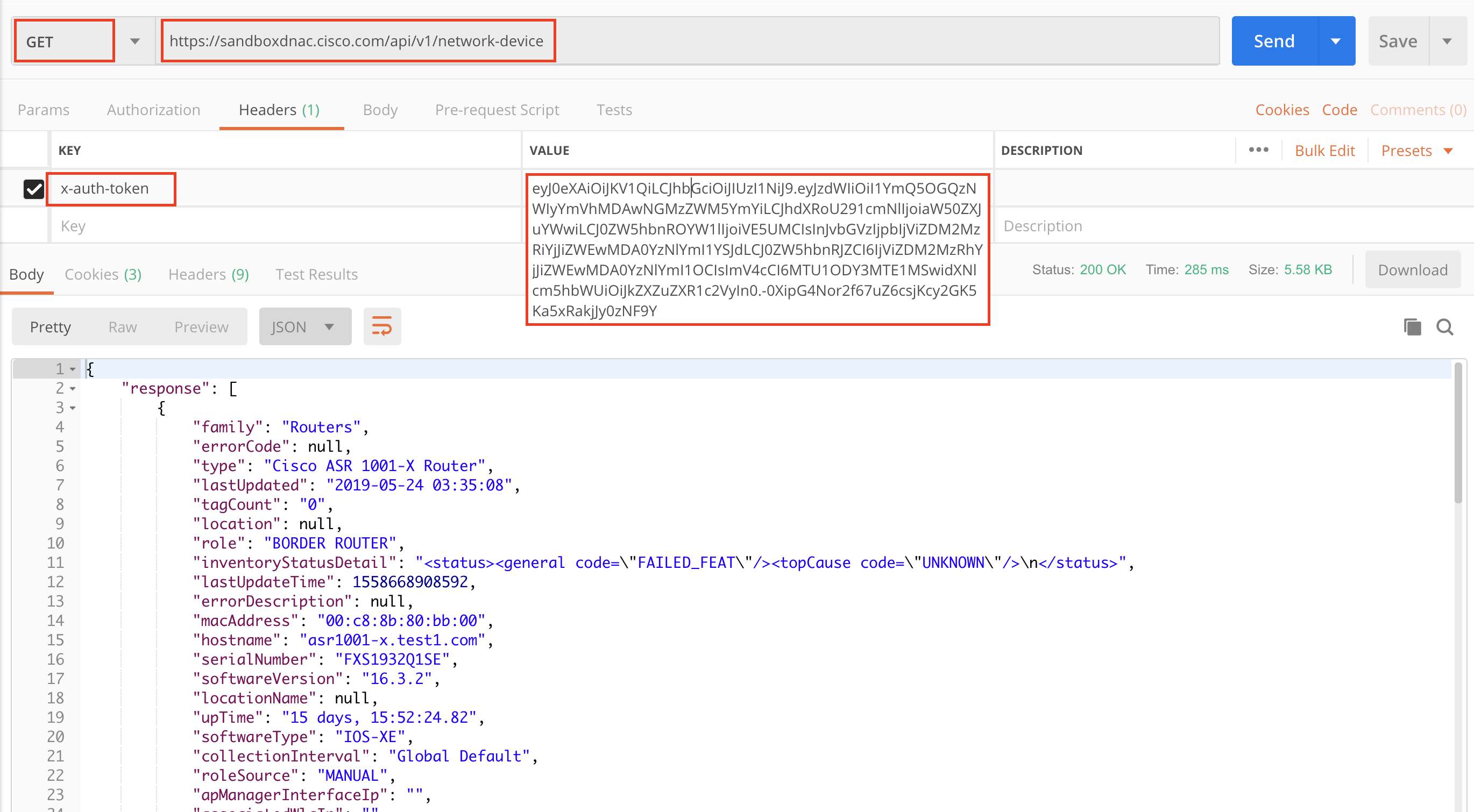
The Network Device Detail Intent API retrieves detailed information about devices by timestamp, MAC address, UUID, name, or nwDeviceName.

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.

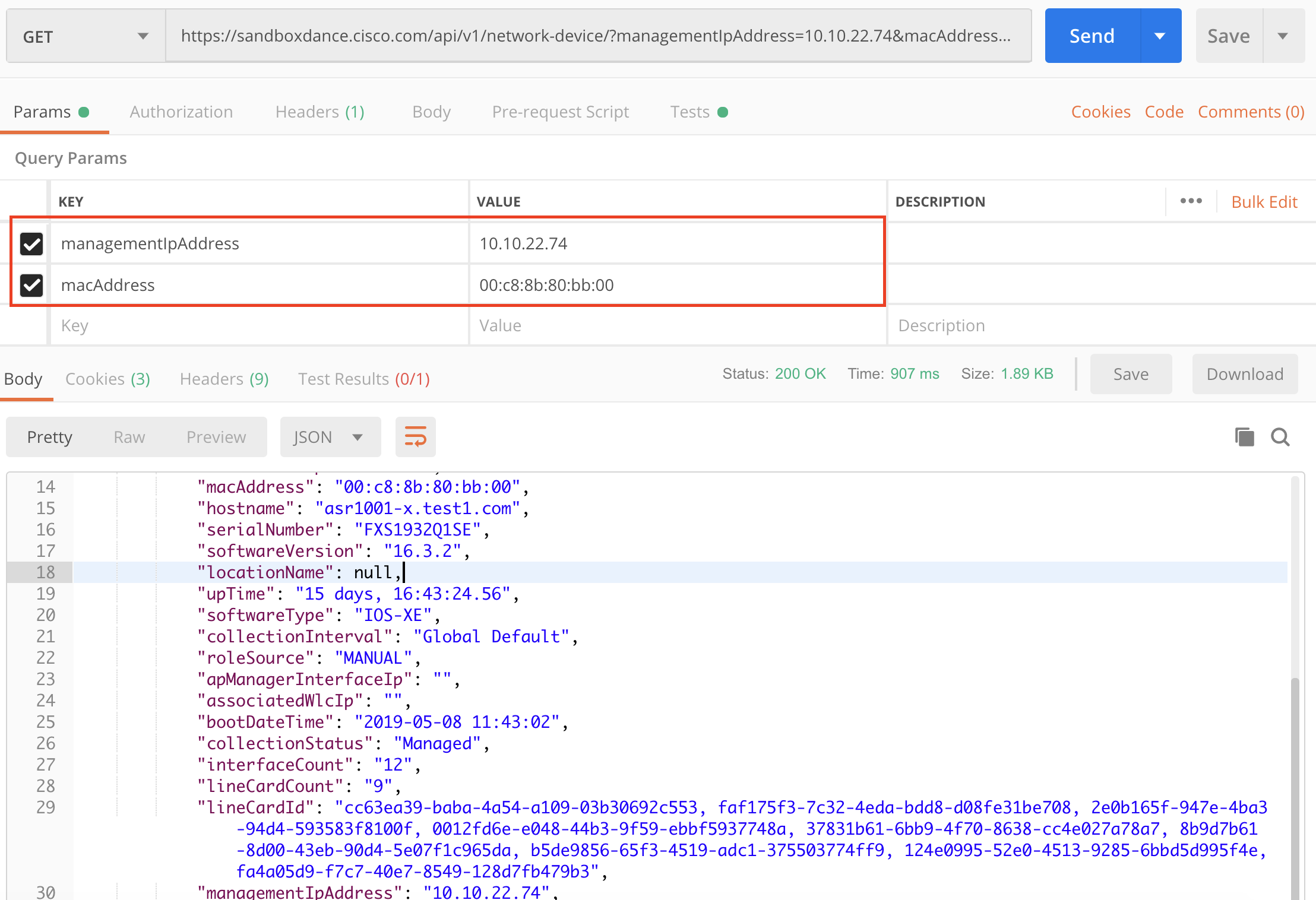


1. Launch Postman and locate the center pane. Here you have the option to select a method for your request. Select **GET** from the drop-down list.
2. In the Request URL section, enter https://sandboxdnac.cisco.com/api/v1/network-device.
3. On the Headers tab, create an x-auth-token key. For the value,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



## 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](https://devnetsandbox.cisco.com/RM/Diagram/Index/c3c949dc-30af-498b-9d77-4f1c07d835f9?diagramType=Topology)
2. **import** requests
3. **from** requests.auth **import** HTTPBasicAuth
4. **from** dnac\_config **import** DNAC, DNAC\_PORT, DNAC\_USER, DNAC\_PASSWORD
5. You will now define the function and write the GET request.
6. token = get\_auth\_token() # Get a Token
7. url = "https://sandboxdnac.cisco.com/api/v1/network-device" #Network Device endpoint
8. hdr = {'x-auth-token': token, 'content-type' : 'application/json'} #Build header Info
9. resp = requests.get(url, headers=hdr) # Make the Get Request
10. device\_list = resp.json() #capture the data from the controller
11. 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.

1. 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.
2. token = get\_auth\_token() # Get a Token
3. url = "https://sandboxdnac.cisco.com/api/v1/network-device" #Network Device endpoint
4. hdr = {'x-auth-token': token, 'content-type' : 'application/json'} #Build header Info
5. querystring = {"macAddress":"00:c8:8b:80:bb:00","managementIpAddress":"10.10.22.74"}
6. resp = requests.get(url, headers=hdr, params=querystring) # Make the Get Request
7. device\_list = resp.json() # Capture data from the controller
8. print\_device\_list(device\_list) # Pretty print the data
9. Execute the code.
10. **if** \_\_name\_\_ == "\_\_main\_\_":
11. get\_device\_list()

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

hostname mgmt IP serial platformId SW Version role Uptime

asr1001-x.test1.com 10.10.22.74 FXS1932Q1SE ASR1001-X 16.3.2 BORDER ROUTER 16 days, 16:57:20.36

cat\_9k\_2.test1.com 10.10.22.70 FCW2140L039 C9300-24UX 16.6.4a ACCESS 16 days, 16:10:33.49

New.test1.com 10.10.22.66 FCW2136L0AK C9300-24UX 16.6.1 ACCESS 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.

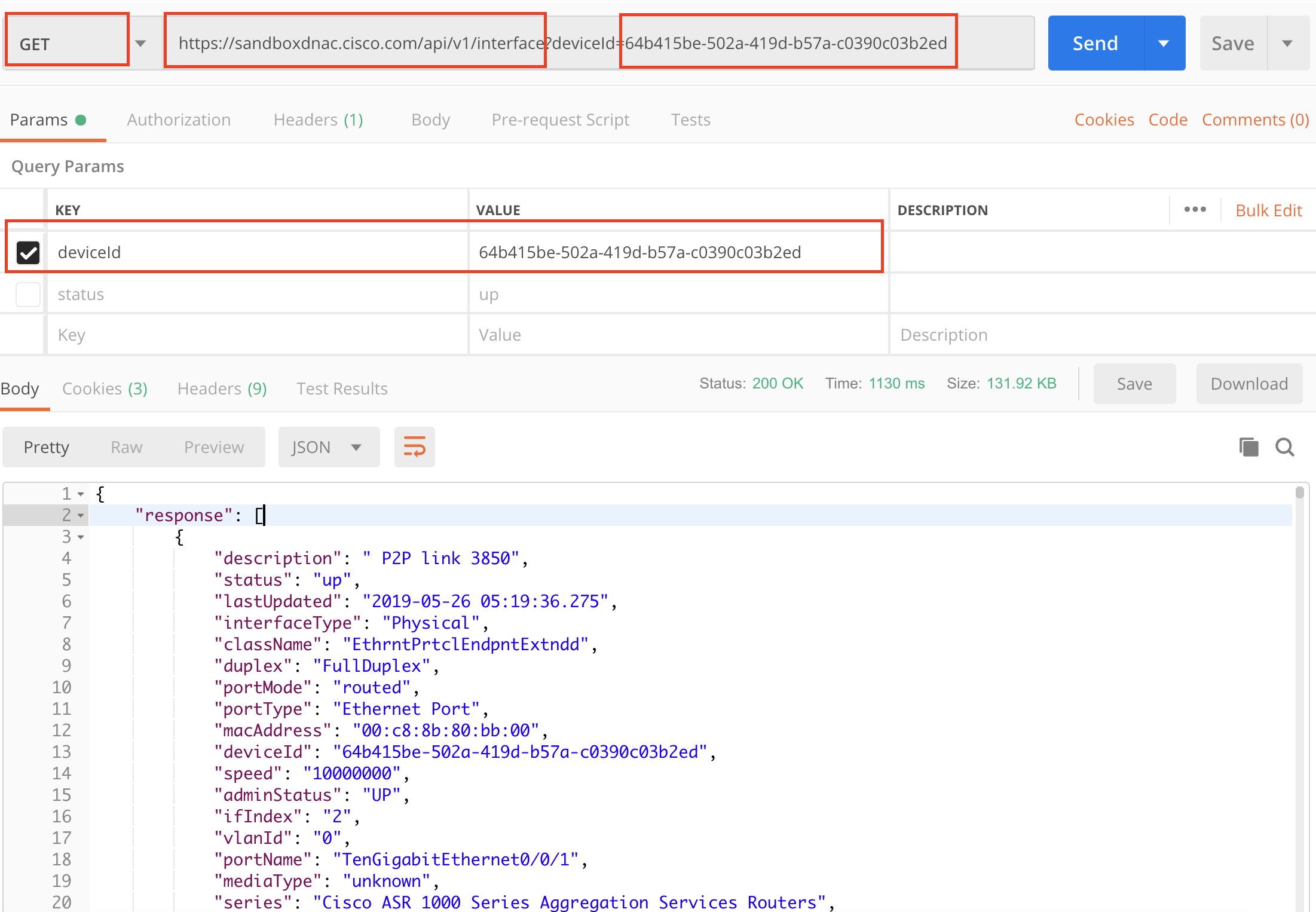
**Complete Code**

Click below to view the [complete script](https://github.com/kiskander/DNAC-101-Code)

# 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](https://devnetsandbox.cisco.com/RM/Diagram/Index/c3c949dc-30af-498b-9d77-4f1c07d835f9?diagramType=Topology)
2. **import** requests
3. **from** dnac\_config **import** DNAC, DNAC\_PORT, DNAC\_USER, DNAC\_PASSWORD
4. 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.
5. **def** **get\_device\_int**(device\_id):
6. """
7. Building out function to retrieve device interface. Using requests.get
8. to make a call to the network device Endpoint
9. """
10. url = "https://sandboxdnac.cisco.com/api/v1/interface"
11. hdr = {'x-auth-token': token, 'content-type' : 'application/json'}
12. querystring = {"macAddress": device\_id} # Dynamically build the query params to get device-specific Interface info
13. resp = requests.get(url, headers=hdr, params=querystring) # Make the Get Request
14. interface\_info\_json = resp.json()
15. 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](https://github.com/kiskander/DNAC-101-Code)

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