

使用带Python的Catalyst Center API

目录

[简介](#)

[先决条件](#)

[要求](#)

[使用的组件](#)

[配置](#)

[概述](#)

[模块](#)

[生成令牌](#)

[测试API](#)

[带报头参数的API](#)

[带查询参数的API](#)

简介

本文档介绍如何使用Python使用Cisco Catalyst Center上提供的不同API。

先决条件

要求

基本知识：

- 思科Catalyst中心
- API
- Python

使用的组件

- 思科Catalyst Center 2.3.5.x
- Python 3.x.x

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您的网络处于活动状态，请确保您了解所有命令的潜在影响。



注意：思科技术支持中心(TAC)不为Python提供技术支持。如果您遇到Python问题，请联系Python支持寻求技术支持。

配置

概述

Cisco Catalyst Center提供了许多API。要验证可以使用哪些API，请在Catalyst Center上导航到平台>开发人员工具包> API。

Check out our API capabilities and try them out for yourself

Explore our developer documentation or test different APIs in your network environment to build, connect, and leverage rich capabilities of Cisco DNA Center.



🔍 Search

Authentication

Cisco DNA Center System

Health and Performance

Licenses

Platform

User and Roles

Connectivity

Fabric Wireless

SDA

Wireless

Ecosystem Integrations

ITSM

Event Management

Integrations

🔍 Search API

Authentication

Authentication APIs provide an authorized token for accessing any REST API.

***Prerequisite*:** Add the request header 'x-auth-token' with the generated authorized token to get a successful API response.

Method	Name	Description	URL	Actions
POST	importCertificate	This method is used to upload a certificate	/certificate	⋮
POST	importCertificateP12	This method is used to upload a PKCS#12 file	/certificate-p12	⋮
POST	Authentication API	API to obtain an access token, which remains valid for 1 hour. The token obtained using this API is required to be set as value to the X-Auth-Token HTTP...	/auth/token	⋮

Catalyst Center API页

每个API都有其自己的目的，具体取决于需要在Catalyst Center上执行的信息或操作。要使API正常运行，作为先决条件，必须使用令牌向Catalyst Center正确进行身份验证并获得成功的API响应。该令牌相应地标识REST调用方的权限。

此外，确定构成API的组件也非常重要，如下所示：

- URL：提供对特定资源的访问的终端。
- 方法：所有API都必须包含方法。它定义了客户端要对特定终端执行的操作/操作。示例：POST、GET、PUT、DELETE。
- 报头：以键值对格式提供有关请求的其他信息。例如，授权报头提供使用凭证的身份验证方法。
- 参数：通过使用API向终端提供特定说明的变量。参数可以是终端的URL的一部分。
- 负载：在API调用期间需要发送到终端的数据。

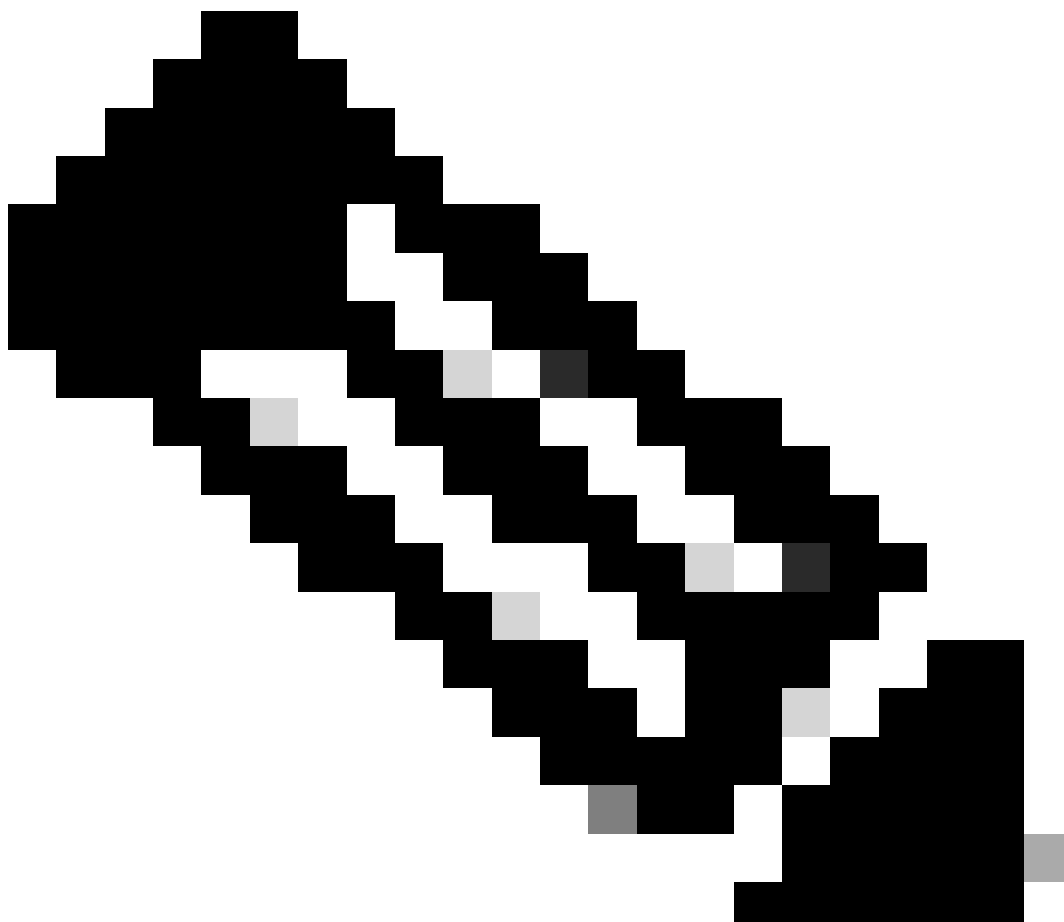


注意：有关Catalyst Center上提供的每个API的详细信息，请参阅[API参考指南](#)。

模块

使用的Python模块：

- 请求：此模块允许向特定URL发送HTTP/1.1请求。有关模块的详细信息，请参阅[请求模块指南](#)。
- base64：提供编码和解码功能。有关模块的详细信息，请参阅[base64模块指南](#)。
- json：此模块允许从API响应获取特定数据。有关模块的详细信息，请参阅[json模块指南](#)。



注意：有关如何安装Python模块的更多信息，请参阅[安装Python模块](#)文档。

生成令牌

名为身份验证API的API必须用于生成新令牌。

身份验证API：

```
POST https://<CatalystCenterIP>/dna/system/api/v1/auth/token
```

必须说明生成的令牌有效期为1小时。1小时后，必须使用上面提到的相同API生成新令牌。

在新的Python文件中，导入模块(请求、base64和json)，然后创建四个变量：

```
import requests
import base64
import json

user = 'user'    # User to login to Catalyst Center
password = 'password'    # Password to login to Catalyst Center
token = ''      # Variable to store the token string
authorizationBase64 = ''    # Variable that stores Base64 encoded string of "username:password"
```

身份验证API支持将基本身份验证作为报头中的授权令牌。基本身份验证是一种可用于对终端进行身份验证的方法，该方法提供用户名和密码，用冒号(username : password)分隔。两个值都采用base64编码，终端会解码登录凭证并检查用户是否能够访问。

要为我们的用户名和口令创建Base64编码的字符串，请使用base64模块。为此，使用了b64encode功能。

```
byte_string = (f'{user}:{password}').encode("ascii")
authorizationBase64 = base64.b64encode(byte_string).decode()
```

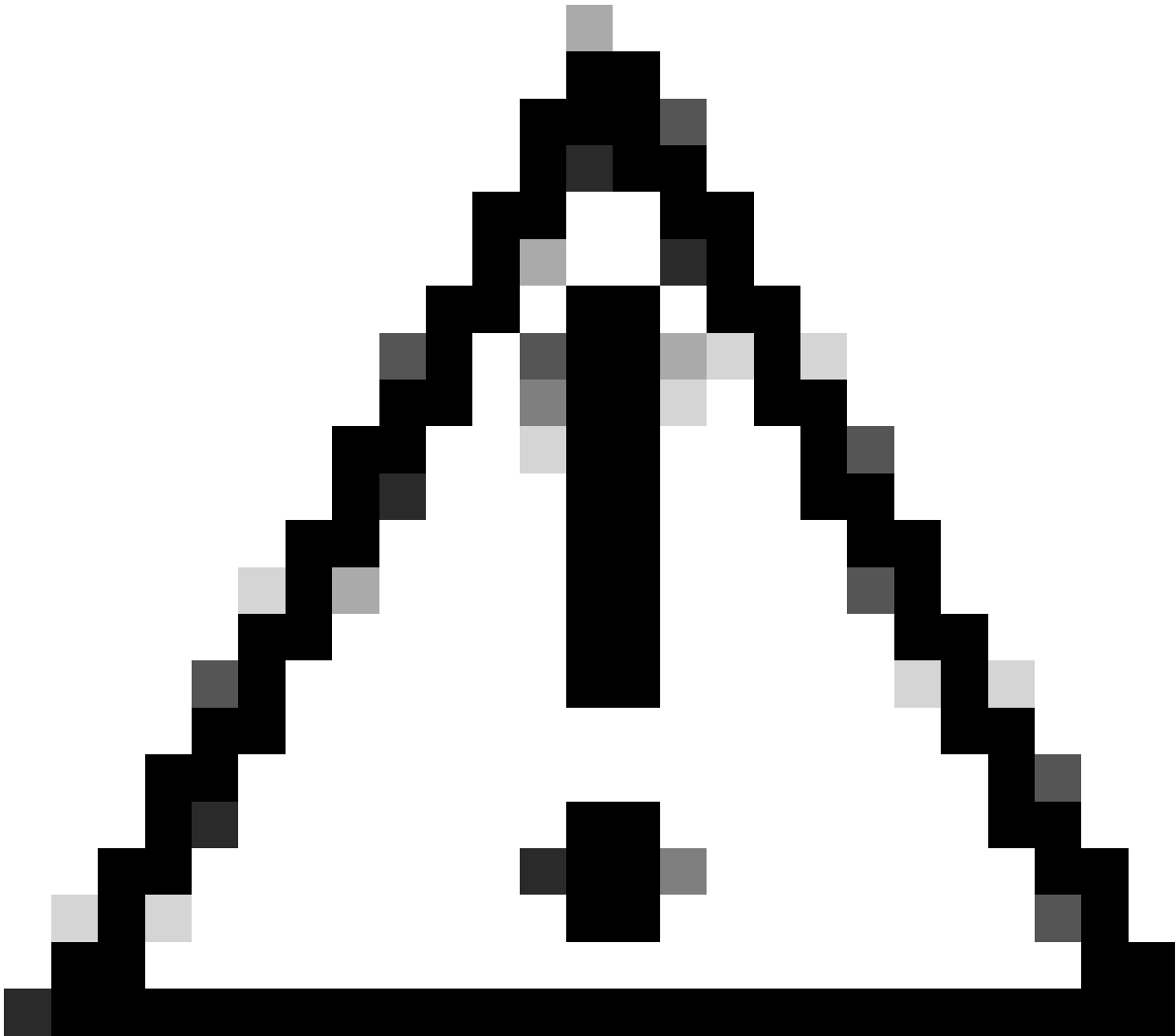
根据上述代码，使用“.encode("ascii")”函数创建了byte_string变量。这是因为base64.b64encode函数需要类似于字节的对象。另请注意，user和password变量用于保留字符串格式user : password。最后，使用用户和密码创建了一个base64编码的字节字符串。使用“decode()”方法，值已转换为str对象。

要进行验证，可以打印authorizationBase64变量的值：

```
print(authorizationBase64)
```

输出示例：

```
am9yZ2QhbDI6Sm9yZ2VhbDXxXxXx
```



注意：base64不是加密算法。不得用于安全目的。身份验证API还支持将AES密钥加密作为报头中的授权令牌，从而提高安全性。

现在已创建一个base64编码的字符串（使用用户和密码向Catalyst Center进行身份验证），是时候继续使用模块请求进行API身份验证API调用了。另外，称为request的函数允许获取包含请求文本的响应对象。

该方法的语法：

```
requests.request("method", "url", **kwargs)
```

**kwargs表示传递到请求中的任何参数，例如，cookie、用户代理、负载、报头等。

身份验证API指定方法为POST，URL为“/dna/system/api/v1/auth/token”，并且需要在Header中指定基本身份验证。

创建这些变量是为了将其用于request()函数。

```
url = https://<CatalystCenterIP>/api/system/v1/auth/token
headers = {
    'content-type': "application/json",
    'Authorization': 'Basic ' + authorizationBase64
}
```

对于headers变量，指定了两个内容。第一个类型是content-type，它指定发送到终端的资源媒体类型（这帮助终端准确分析并处理数据）。第二个参数是Authorization，在这种情况下，会发送变量authorizationBase64（存储我们的base64压缩字符串）作为参数来向Catalyst Center进行身份验证。

现在，继续使用request()函数执行API调用。下一个代码显示函数的语法：

```
response = requests.request("POST", url, headers=headers)
```

创建response变量是为了存储我们发出的API调用的数据。

要打印获取的响应，请在响应变量中将print函数与text()方法一起使用。text()方法使用从Catalyst Center收到的响应生成str对象。

```
print(response.text)
```

输出示例：

```
{"Token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpzZW50b3V5IiwiaWF0IjoiYm50ZXJyYXVwLCW2vMPubU0JN1q"}
!--- Output is suppressed
```

注意：如果Catalyst Center使用的是自签名证书，则API请求可能会失败，并显示下一个错误：

```
requests.exceptions.SSLError: HTTPSConnectionPool(host='X.X.X.X', port=443): Max retries exceeded
```

要解决此问题，需要将verify参数作为False添加到请求功能中。这将忽略验证来自终端(Catalyst Center)的SSL证书。

```
response = requests.request("POST", url, headers=headers, verify=False)
```

请注意，根据从API身份验证调用收到的响应，此结构类似于Python中的词典，但它是一个str对象

。

要验证对象的类型，请使用type()函数。

```
print(type(response.text))
```

返回下一个输出：

```
<class 'str'>
```

实际上，只需要从从API接收的响应中提取令牌值，而不是整个字符串，因为要使用其他Catalyst Center API，只有令牌必须作为参数传递。

由于从API调用收到的响应具有类似于Python中词典的结构，但对象类型为str，因此需要使用json模块将所述的对象转换为词典。这将从从API接收的整个字符串中提取令牌值。

为此，函数json.loads()将字符串转换为词典，以便稍后仅提取令牌值并将其直接分配到令牌变量中。

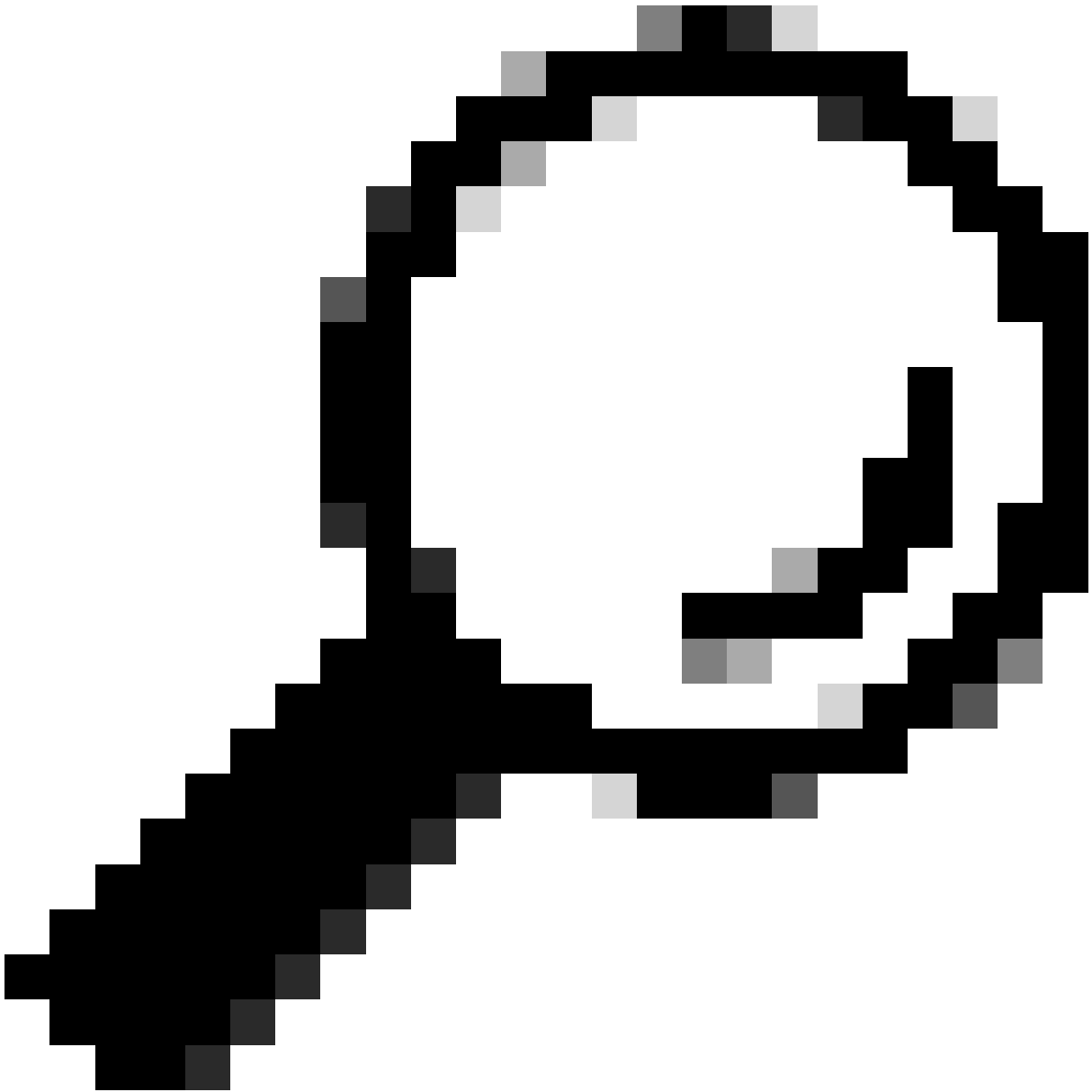
```
token = json.loads(response.text) # Converting the response.text string value into a dictionary (It is  
token = (token["Token"]) # Extracting just the token value by specifying the key as a parameter.
```

要验证token变量是否仅将令牌作为其值，请继续打印。

```
print(token)
```

输出示例：

```
eyJhbGciOiJSUzI1NiIsInR5cGU6IiwiLCJpc09s1zVmNjk0NjhkNTFhNDJlZWELCU291cmN1IjoiaW50ZXJuYmwiLCW2vMPubU0JN1qxOXNe1jMzY  
!--- Output is suppressed
```



提示：由于每个生成的令牌默认在1小时内到期，因此每次令牌到期时，都可以创建并调用包含生成令牌代码的Python方法，而无需通过调用创建的方法运行整个程序。

测试API

现在，令牌已成功分配给令牌变量，可以使用可用的Catalyst Center API。

在这种情况下，将测试Cisco DNA中心节点配置摘要 API。

思科DNA中心节点配置摘要

GET <https://<CatalystCenterIP>/dna/intent/api/v1/nodes-config>

此API提供有关Catalyst Center的当前配置的详细信息，例如配置的NTP服务器、节点名称、集群内链路、LACP模式等。

思科DNA中心节点配置摘要 API指定，在本例中，使用的方法是GET，URL是"/dna/intent/api/v1/nodes-config"，并且，由于已经提取令牌字符串并将其分配给令牌变量，因此这次令牌作为API调用报头中的变量传递为'X-Auth-Token'：后跟令牌。

这将验证对Catalyst Center执行的每个API调用的请求。请记住，每个令牌持续1小时。1小时后，必须生成新令牌才能继续向Catalyst Center进行API调用。

继续创建变量以测试API：

```
nodeInfo_url = "https://<CatalystCenterIP>/dna/intent/api/v1/nodes-config"
nodeInfo_headers = {
    'X-Auth-Token': token
}

nodeInfoResponse = requests.request("GET", nodeInfo_url, headers=nodeInfo_headers)
```

nodeInfo_url变量的创建是为了存储我们的API的URL。nodeInfo_headers变量存储我们的API的标头。在本示例中，“X-Auth-Token：”和Token变量作为参数传递以成功向Catalyst Center验证请求。最后，nodeInfoResponse变量存储API的响应。

要验证收到的响应，您可以使用print()函数。

输出示例：

```
{"response": {"nodes": [{"name": "Catalyst Center", "id": "ea5dbec1-fbb6-4339-9242-7694eb1cXxXx", "netw
!--- Output is suppressed
```



注意：如果Catalyst Center中使用自签名证书，则API请求可能会失败，并出现下一个错误：

```
requests.exceptions.SSLError: HTTPSConnectionPool(host='X.X.X.X', port=443): Max retries exceeded
```

要解决此问题，需要将verify参数添加为False到请求。这会抑制从终端(Catalyst Center)验证SSL证书。

```
nodeInfoResponse = requests.request("GET", nodeInfo_url, headers=nodeInfo_headers, verify=False)
```

从API收到的响应可能难以阅读。使用json()模块，可以用更易读的字符串打印响应。首先，必须使用json.loads()函数后跟json.dumps()函数将API响应加载到JSON对象：

```
jsonFormat = (json.loads(nodeInfoResponse.text)) # Creating a JSON object from the string received from
print(json.dumps(jsonFormat, indent=1)) # Printing the response in a more readable string using the dump
```

json.dumps : 此函数返回JSON对象，该对象作为JSON格式字符串中的参数。

缩进 : 此参数定义JSON格式字符串的缩进级别。

输出示例 :

```
{
  "response": {
    "nodes": [
      {
        "name": "X.X.X.X",
        "id": "ea5dbec1-fbb6-4339-9242-7694eb1xXxX",
        "network": [
          {
            "slave": [
              "enp9s0"
            ],
            "lACP_supported": true,
            "intra_cluster_link": false,
!--- Output is suppressed
```

带报头参数的API

有些API需要在报头中发送一些参数，才能按预期工作。在这种情况下，将测试Get Client Enriculation Details API。

```
GET https://<CatalystCenterIP>/dna/intent/api/v1/client-enrichment-details
```

要验证API需要哪些报头参数才能正常运行，请导航到平台>开发人员工具包> APIs >获取客户端丰富详细信息，然后单击API的名称。此时将打开一个新窗口，并在Parameters 选项下显示API运行所需的报头参数。

Get Client Enrichment Details



GET <https://10.88.244.133/dna/intent/api/v1/client-enrichment-details>

Enriches a given network End User context (a network user-id or end user's device Mac Address) with details about the user, the devices that the user is connected to and the assurance issues that the user is impacted by

[Cisco DevNet API Guide](#)

TAGS

Client Enrichment Network Event

Parameters Responses Policies Code Preview

Request Header Parameters

Name	Description	DataType	Required	Default Value
entity_type	Client enrichment details can be fetched based on either User ID or Client MAC address. This parameter value must either be network_user_id/mac_address	string	Yes	
entity_value	Contains the actual value for the entity type that has been defined	string	Yes	
issueCategory	The category of the DNA event based on which the underlying issues need to be fetched	string	No	

在这种情况下，根据说明，对于entity_type参数，值可以是network_user_id或mac_address，并且entity_value参数必须包含已定义的实体类型的值。

要继续操作，需要定义两个新变量，entity_type和entity_value，以及对应的值：

```
entity_type = 'mac_address'    #This value could be either 'network_user_id' or 'mac_address'.
entity_value = 'e4:5f:02:ff:xx:xx'    #Depending of the 'entity_type' used, need to add the correspondi
```

还会创建新的变量来执行API调用。API调用的URL存储在userEnriculation_url变量中。信头存储在userEnfurationHeaders变量中。收到的响应存储在userEnriculationResponse变量中。

```
userEnrichment_url = "https://<CatalystCenterIP>/dna/intent/api/v1/user-enrichment-details"

userEnrichmentHeaders = {
'X-Auth-Token': token,
'entity_type': entity_type,
'entity_value': entity_value,
}

userEnrichmentResponse = requests.request("GET", userEnrichment_url, headers=userEnrichmentHeaders)
```

如您所见，从userEnfurationHeaders中，entity_type和entity_value变量已作为API调用的报头参数与token变量一起传递。

要验证收到的响应，请使用print()函数。

```
print(userEnrichmentResponse.text)
```

输出示例：

```
[ {
  "userDetails" : {
    "id" : "E4:5F:02:FF:xx:xx",
    "connectionStatus" : "CONNECTED",
    "tracked" : "No",
    "hostType" : "WIRELESS",
    "userId" : null,
    "duid" : "",
    "identifier" : "jonberrypi-1",
    "hostName" : "jonberrypi-1",
    "hostOs" : null,
    "hostVersion" : null,
    "subType" : "RaspberryPi-Device",
    "firmwareVersion" : null,
    "deviceVendor" : null,
    "deviceForm" : null,
    "salesCode" : null,
    "countryCode" : null,
    "lastUpdated" : 1721225220000,
    "healthScore" : [ {
      "healthType" : "OVERALL",
      "reason" : "",
      "score" : 10
    }, {
      "healthType" : "ONBOARDED",
      "reason" : "",
      "score" : 4
    }
  ]
} ]
!--- Output is suppressed
```

带查询参数的API

查询参数可用于过滤API返回的特定结果数。这些参数会被添加到API的URL中。

Get Device List API调用已测试。

GET <https://10.88.244.133/dna/intent/api/v1/network-device>

Get Device List API返回添加到Catalyst Center中的所有设备的列表。如果请求特定设备的详细信息，查询参数有助于过滤特定信息。

要验证哪些查询参数对于API可用，请导航到平台>开发人员工具包> APIs >获取设备列表，然后单击API的名称。此时将打开一个新窗口，并在Parameters选项下显示可用于API的查询参数。

Get Device list

GET https://10.88.244.133/dna/intent/api/v1/network-device

Returns list of network devices based on filter criteria such as management IP address, mac address, hostname, etc. You can use the .* in any value to conduct a wildcard search. For example, to find all hostnames beginning with myhost in the IP address range 192.25.18.n, issue the following request: GET /dna/intent/api/v1/network-device?hostname=myhost.*&managementIpAddress=192.25.18.* If id parameter is provided with comma separated ids, it will return the list of network-devices for the given ids and ignores the other request parameters. You can also specify offset & limit to get the required list.

[Cisco DevNet API Guide](#)

Parameters Responses Code Preview

Request Query Parameters

Name	Description	DataType	Required	Default Value
hostname	hostname	array	No	
managementIpAddress	managementIpAddress	array	No	
macAddress	macAddress	array	No	
locationName	locationName	array	No	
serialNumber	serialNumber	array	No	
location	location	array	No	
family	family	array	No	
type	type	array	No	
series	series	array	No	

在本示例中，将使用managementIpAddress和serialNumber查询参数（请考虑对于API呼叫使用所有查询参数并非必要的因素）。继续创建并分配两个查询参数的相应值。

```
managementIpAddress = '10.82.143.250'  
serialNumber = 'FD025160X9L'
```

如上所述，查询参数在API的URL中添加，特别是在API的末尾，使用“？”后跟查询参数。

如果要使用多个查询参数，将“&”符号置于它们之间以形成所谓的查询字符串。

下一个示例显示如何将查询参数添加到存储API调用的URL的deviceListUri变量。

```
deviceListUri = "https://<CatalystCenterIP>/dna/intent/api/v1/network-device?managementIpAddress=" + m
```

请注意，之前创建的变量已附加到URL字符串。换句话说，URL的整个字符串如下所示：

```
deviceListUri = "https://<CatalystCenterIP>/dna/intent/api/v1/network-device?managementIpAddress=10.82
```

继续API调用，将创建deviceListHeaders变量以存储API报头以及作为参数传递的token变量，并deviceListResponse变量存储API响应。

```
deviceListHeaders = {  
    'X-Auth-Token': token,  
}
```

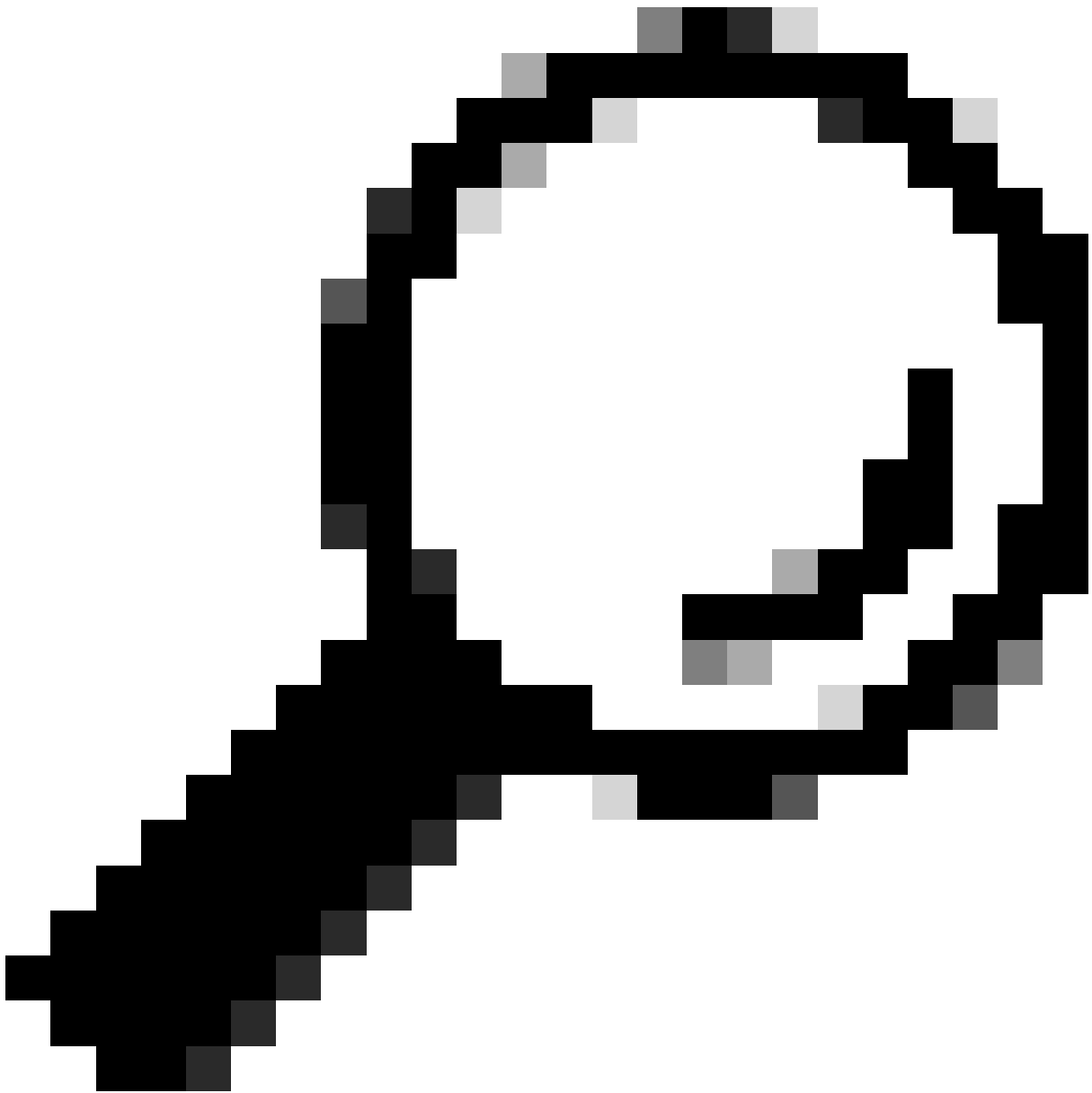
```
deviceListResponse = requests.request("GET", deviceListUri, headers=deviceListHeaders)
```

要验证收到的响应，您可以使用print()函数。

```
print(deviceListResponse.text)
```

输出示例：

```
{"response":[{"family":"Switches and Hubs","description":"Cisco IOS Software [Cupertino], Catalyst L3 S  
!--- Output is suppressed
```



提示：要以更具可读性的方式打印响应，您可以使用测试API部分中介绍的`json.loads()`和`json.dumps()`函数。

关于此翻译

思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。