Uso de las API de Catalyst Center con Python

Contenido

Introducción

Este documento describe cómo utilizar las diferentes API disponibles en Cisco Catalyst Center mediante Python.

Prerequisites

Requirements

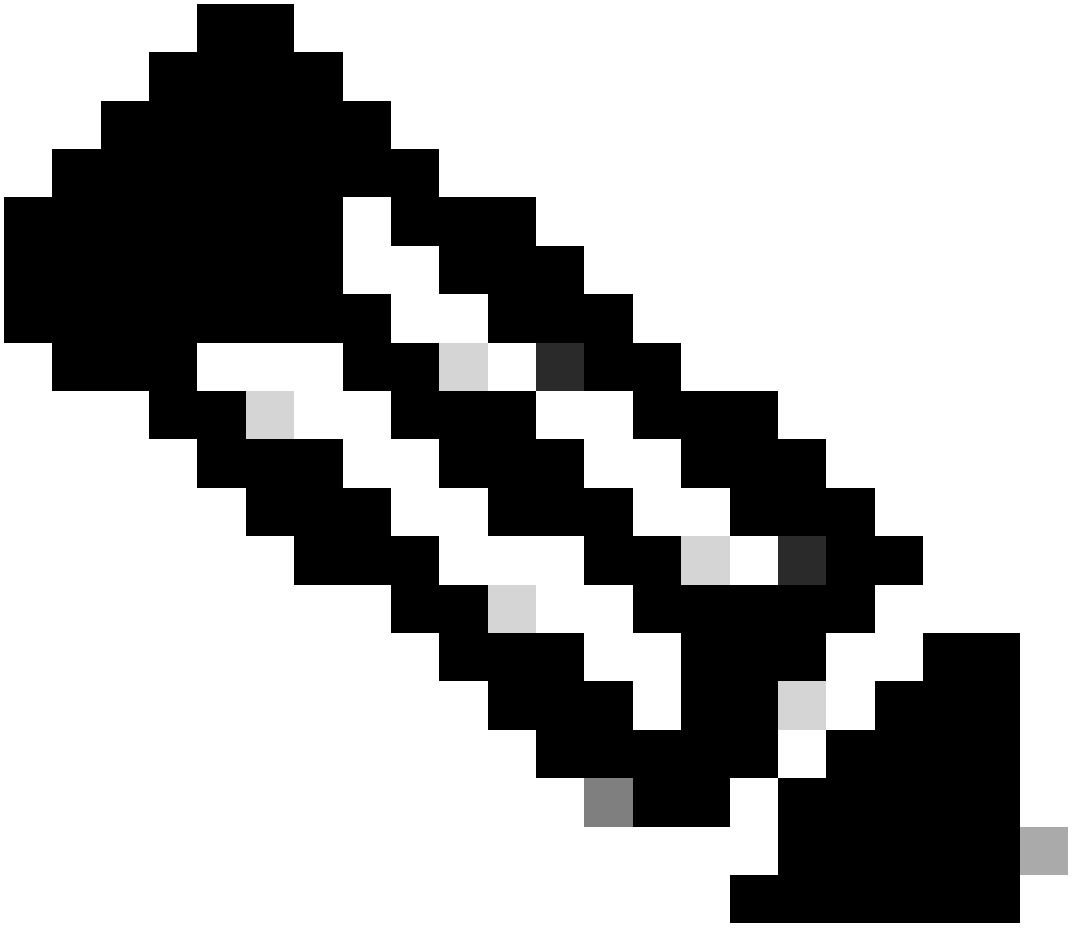
Conocimientos básicos sobre:

- Cisco Catalyst Center
- API
- Python

Componentes Utilizados

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

La información que contiene este documento se creó a partir de los dispositivos en un ambiente de laboratorio específico. Todos los dispositivos que se utilizan en este documento se pusieron en funcionamiento con una configuración verificada (predeterminada). Si tiene una red en vivo, asegúrese de entender el posible impacto de cualquier comando.

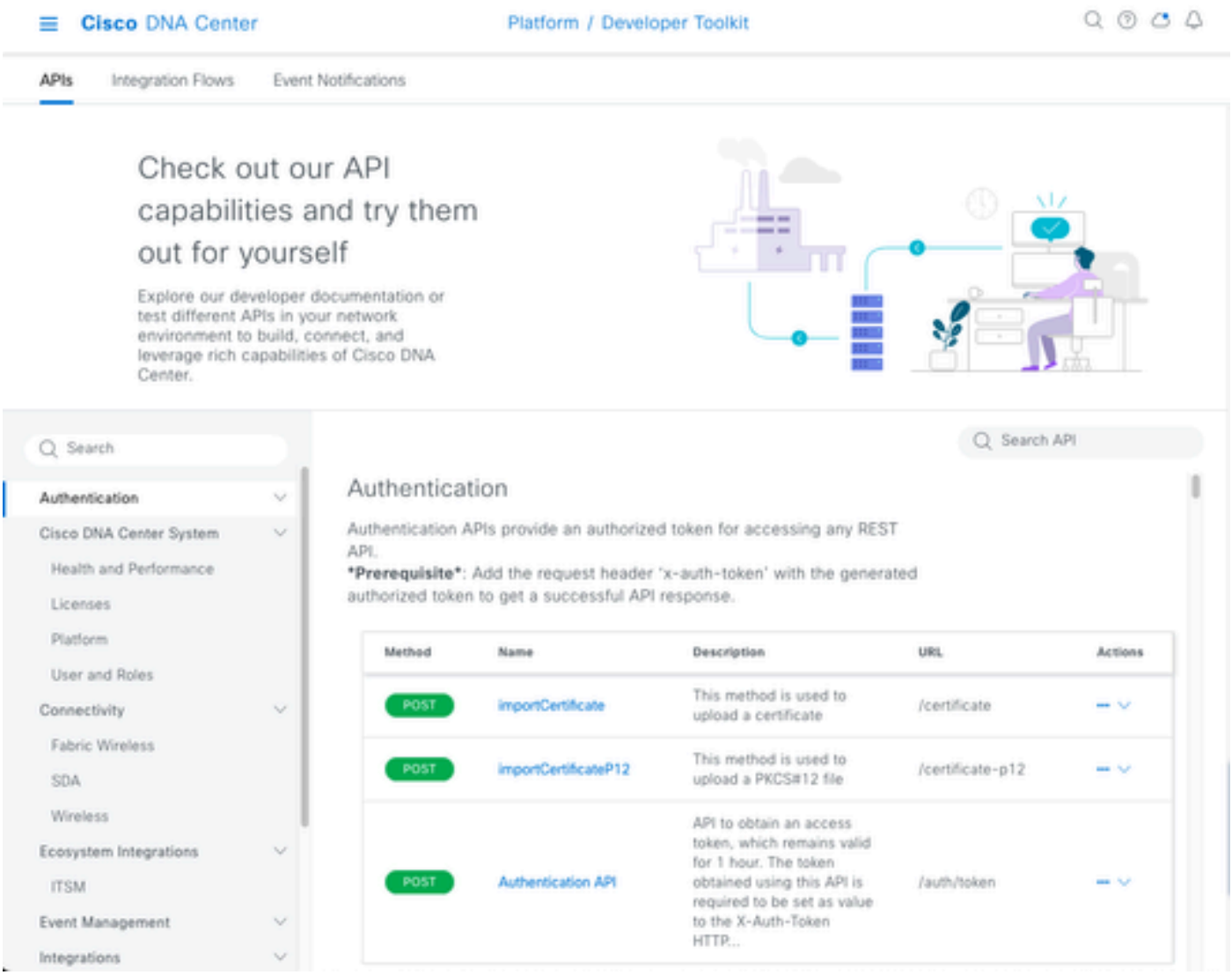


Nota: Cisco Technical Assistance Center (TAC) no proporciona asistencia técnica para Python. Si experimenta problemas con Python, póngase en contacto con el soporte técnico de Python para obtener asistencia técnica.

Configurar

Overview

Cisco Catalyst Center tiene muchas API disponibles. Para verificar qué API se pueden utilizar, en Catalyst Center, navegue hasta Plataforma > Kit de herramientas de desarrollador > API.

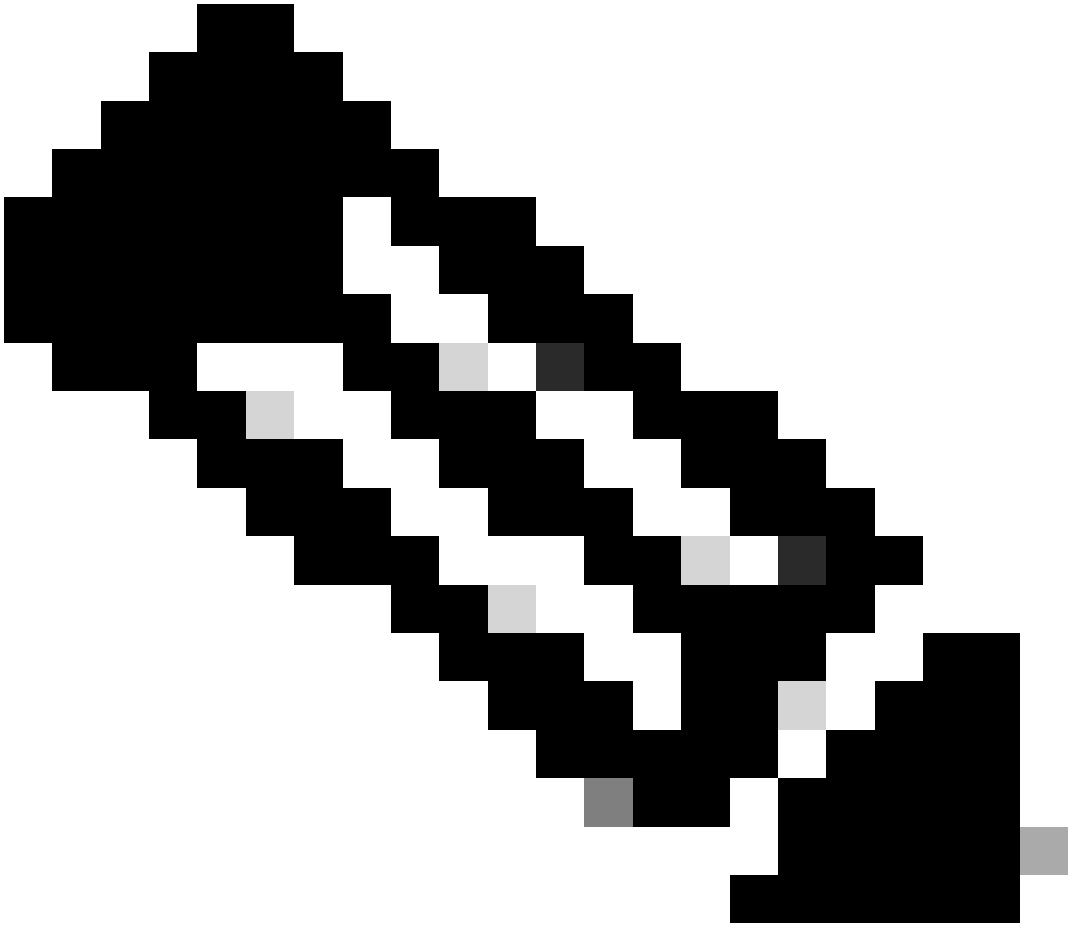


Página de API de Catalyst Center

Cada API tiene su propio propósito, en función de la información o la acción que se deba realizar en Catalyst Center. Para que las API funcionen, como requisito previo, se debe utilizar un token para autenticarse correctamente en el Catalyst Center y obtener una respuesta de API exitosa. El token identifica los privilegios para la persona que llama REST en consecuencia.

También es importante identificar los componentes que conforman una API que son los siguientes:

- URL: terminal que proporciona acceso a un recurso específico.
- Método: todas las API deben incluir un método. Define la acción/operación que el cliente desea realizar al punto final específico. Ejemplos: POST, GET, PUT, DELETE.
- Encabezado: proporciona información adicional sobre la solicitud en formato de pares clave-valor. Por ejemplo, el encabezado de autorización proporciona un método de autenticación mediante credenciales.
- Parámetros: variables que proporcionan instrucciones específicas al extremo mediante la API. Los parámetros pueden formar parte de la dirección URL del extremo.
- Carga: datos que deben enviarse al terminal durante la llamada a la API.

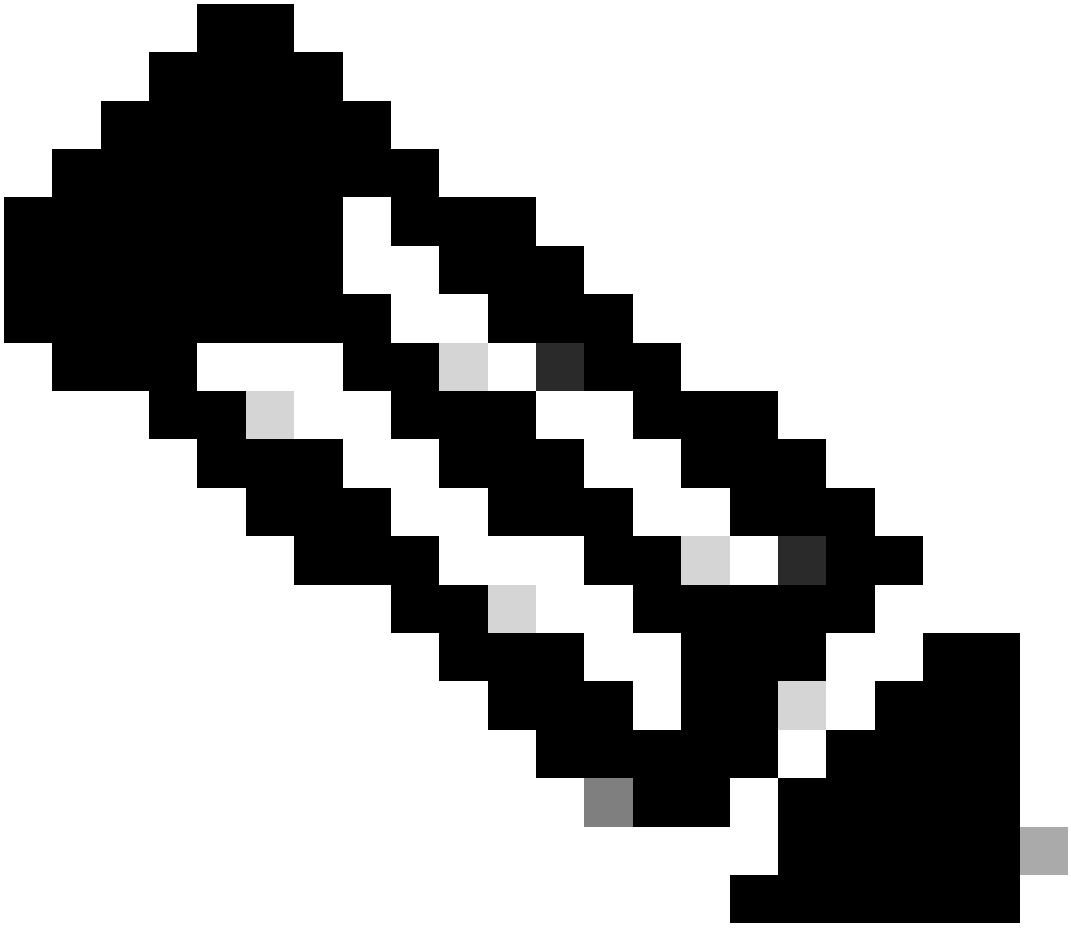


Nota: Para obtener información más detallada sobre cada API disponible en Catalyst Center, consulte la guía de referencia de API.

Módulos

Módulos Python utilizados:

- solicitudes:este módulo permite enviar solicitudes HTTP/1.1 a direcciones URL específicas. Para obtener más información sobre el módulo, consulte la guía del módulo de solicitud.
- base64: Proporciona funciones de codificación y descodificación. Para obtener más información sobre el módulo, consulte la guía del módulo base64.
- json: Este módulo permite obtener datos específicos de la respuesta de las API. Para obtener más información sobre el módulo, consulte la guía del módulo json.



Nota: Para obtener más información sobre cómo instalar módulos Python, consulte la documentación Instalación de módulos Python.

Generar token

La API llamada Authentication API se debe utilizar para generar un nuevo token.

API de autenticación:

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

Es importante mencionar que el token que se genera es válido durante 1 hora. Después de 1 hora, se debe generar un nuevo token usando la misma API mencionada anteriormente.

En un nuevo archivo de Python, importe los módulos (requests, base64 y json) seguidos de la

creación de cuatro variables:

```
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"
```

La API de autenticación admite la autenticación básica como token de autorización en el encabezado. La autenticación básica es un método que se puede utilizar para autenticarse en un extremo, proporcionando un nombre de usuario y una contraseña separados por dos puntos (username:password). Ambos valores están codificados en base64, y el punto final descodifica las credenciales de inicio de sesión y marca, si el usuario puede acceder o no.

Para crear la cadena enlazada a Base64 para nuestro nombre de usuario y contraseña, se utiliza el módulo base64. Para lograr esto, se utiliza la función b64encode.

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

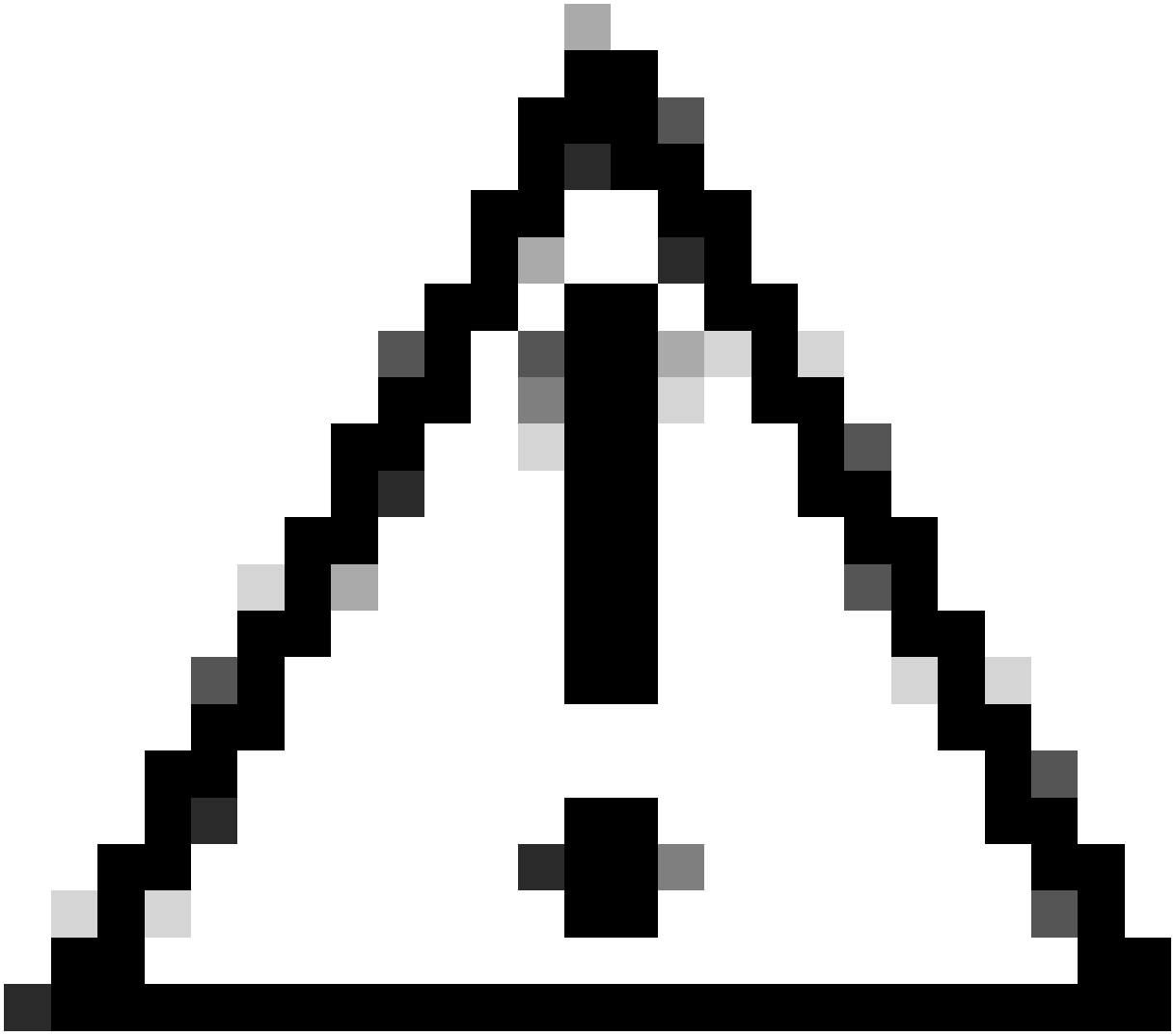
A partir del código anterior, se creó una variable byte_string utilizando la función '.encode("ascii")'. Esto se debe a que la función base64.b64encode requiere un objeto de tipo bytes. Observe también que las variables user y password se utilizaron para mantener el formato de cadena 'user:password'. Finalmente, se creó una cadena de bytes codificada en base64 con el usuario y la contraseña. Utilizando el método 'decode()', el valor se convirtió en el objeto str.

Para verificarlo, puede imprimir el valor de la variable authorizationBase64:

```
print(authorizationBase64)
```

Ejemplo de salida:

```
am9yZ2QhbDI6Sm9yZ2VhbDXxXxXx
```



Precaución: base64 no es un algoritmo de cifrado. No debe utilizarse con fines de seguridad. La API de autenticación también admite el cifrado de clave AES como token de autorización en el encabezado, lo que proporciona más seguridad.

Ahora que se ha creado una cadena codificada en base64 con el usuario y la contraseña para autenticarse en Catalyst Center, es el momento de continuar con la llamada API de autenticación de API mediante las solicitudes de módulo. Además, la función llamada request permite obtener un objeto de respuesta que contiene el texto de la solicitud.

Sintaxis del método:

```
requests.request(“method”, “url”, **kwargs)
```

**kwargs significa cualquier parámetro pasado a la solicitud, por ejemplo, cookies, agentes de usuario, carga útil, encabezados, etc.

La API de autenticación especifica que el método es POST, la URL es "/dna/system/api/v1/auth/token" y la autenticación básica debe especificarse en el encabezado de autenticación.

Estas variables se crean para utilizarlas en la función request().

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

Para la variable de encabezados, se especificaron dos cosas. El primero es content-type, que especifica el tipo de medio del recurso enviado al extremo (esto ayuda al extremo a analizar y procesar los datos con precisión). El segundo es Authorization, que, en este caso, la variable authorizationBase64 (que almacena nuestra cadena de base64) se envía como parámetro para autenticarse en Catalyst Center.

Ahora, proceda a utilizar la función request() para realizar la llamada a la API. El siguiente código muestra la sintaxis de la función:

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

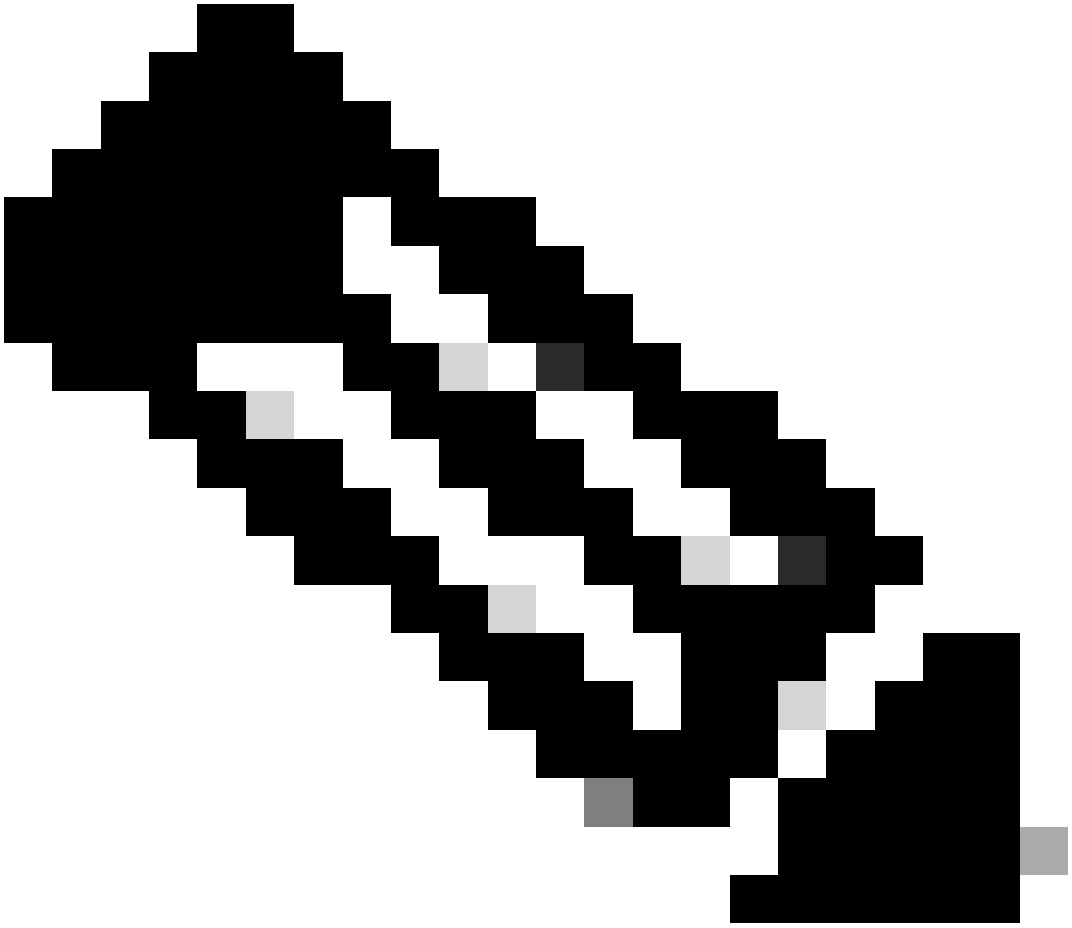
La variable response fue creada para almacenar los datos de nuestra llamada API realizada.

Para imprimir la respuesta obtenida, utilice la función print junto con el método text() en la variable response. El método text() genera un objeto str con la respuesta recibida desde el Centro Catalyst.

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

Ejemplo de salida:

```
{"Token":"eyJhbGciOiJSUzI1NiIsInR5JKO9s1zVmNjkONjhkNTFhNDJ1ZWeLCU291cmNlIjoiaW5OZXJuYWwiLCW2vMPUbUOJNlq
!--- Output is supressed
```



Nota: Si Catalyst Center utiliza un certificado autofirmado, la solicitud de la API puede fallar con el siguiente error:

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

Para solucionar este problema, debe agregar el parámetro verify como False a la función request. Esto ignora la verificación del certificado SSL del terminal (Catalyst Center).

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

De la respuesta recibida de la llamada de autenticación API, observe que la estructura es similar a un diccionario en Python, sin embargo, es un objeto str.

Para validar el tipo de un objeto, utilice la función type().

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

Que devuelve el siguiente resultado:

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

A efectos prácticos, sólo es necesario extraer el valor del token de la respuesta recibida de la API, no de la cadena completa, ya que, para utilizar las otras API de Catalyst Center, sólo se debe pasar el token como parámetro.

Dado que la respuesta recibida de la llamada API tiene una estructura similar a un diccionario en Python pero el tipo de objeto es str, dicho objeto necesita ser convertido en un diccionario usando el módulo json. Esto extrae el valor del token de toda la cadena recibida de la API.

Para lograr esto, la función json.loads() convierte la cadena en un diccionario para luego extraer sólo el valor del token y asignarlo directamente a nuestra variable token.

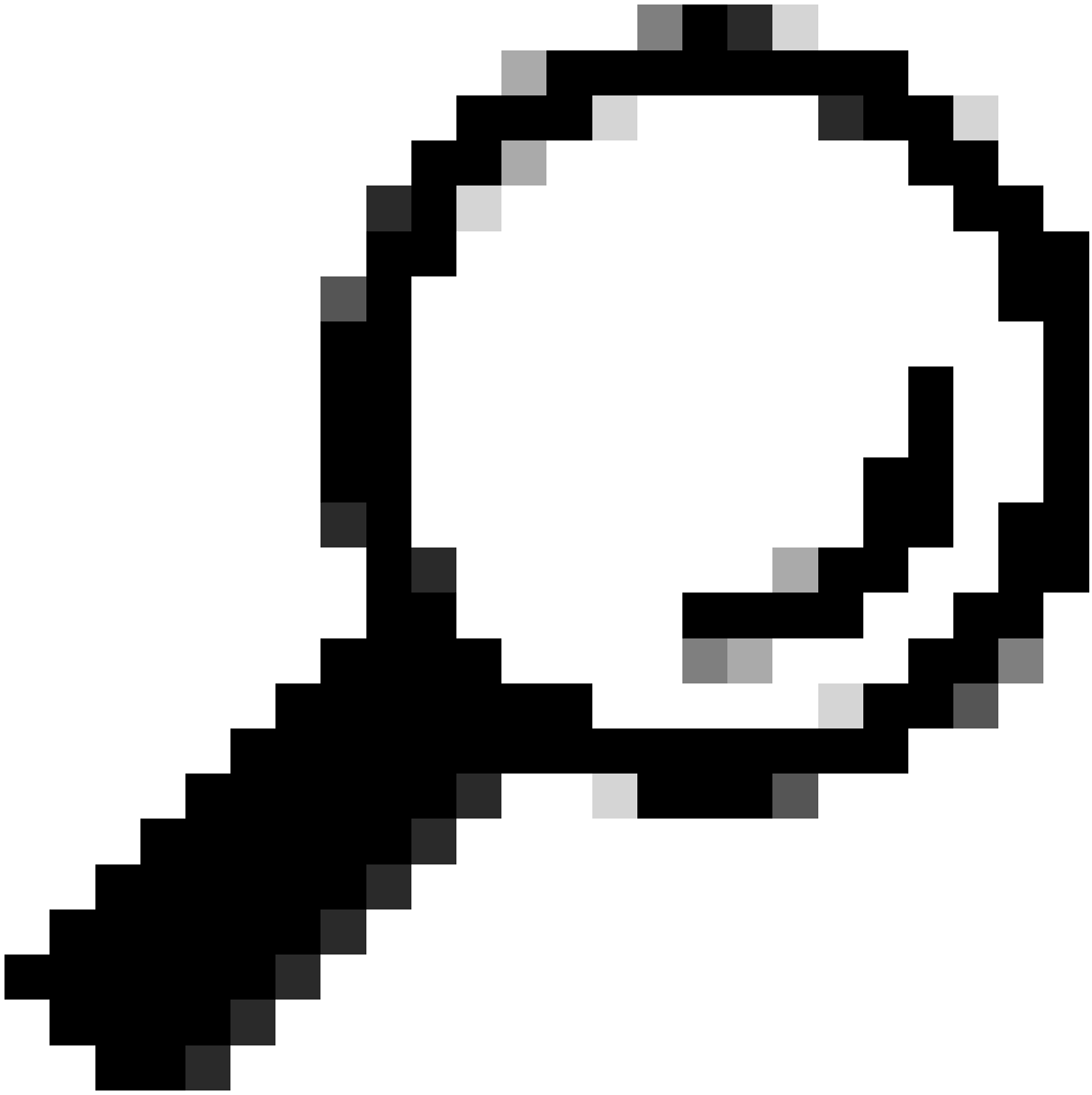
```
token = json.loads(response.text)  # Converting the response.text string value into a dictionary (It is
token = (token["Token"])  # Extracting just the token value by specifying the key as a parameter.
```

Para verificar que la variable token sólo tiene el token como valor, proceda a imprimirla.

```
print(token)
```

Ejemplo de salida:

```
eyJhbGciOiJSUzI1NiIsInR5JKO9s1zVmNjkONjhkNTFhNDJ1ZWeLCU291cmNlIjoiaW5OZXJuYWwiLCW2vMPUbUOJNlqxOXNe1jMzY
!--- Output is supressed
```



Sugerencia: dado que cada token generado caduca en 1 hora de forma predeterminada, se puede crear y llamar a un método Python que contenga el código para generar un token cada vez que caduca un token, sin tener que ejecutar todo el programa simplemente llamando al método creado.

Prueba de una API

Ahora que el token se ha asignado correctamente a la variable de token, se pueden utilizar las API de Catalyst Center disponibles.

En este caso, se prueba la API Resumen de la configuración de los nodos del centro de DNA de Cisco.

Resumen de configuración de nodos de Cisco DNA Center

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

Esta API proporciona detalles sobre la configuración actual de Catalyst Center, como, por ejemplo, el servidor NTP configurado, el nombre del nodo, el enlace dentro del clúster, el modo LACP, etc.

La API de resumen de configuración de nodos del centro de DNA de Cisco especifica, en este caso, que el método utilizado es GET, la URL es "/dna/intent/api/v1/nodes-config" y, dado que la cadena de token se ha extraído y asignado a la variable de token, esta vez el token se pasa como una variable en el encabezado de la llamada de API como ‘X-Auth-Token’:seguida por el token.

Esto autentica la solicitud al Centro Catalyst para cada llamada API que se realiza. Recuerde que cada token dura 1 hora. Después de 1 hora, se debe generar un nuevo token para continuar realizando llamadas API al centro de Catalyst.

Proceda a crear las variables para probar la API:

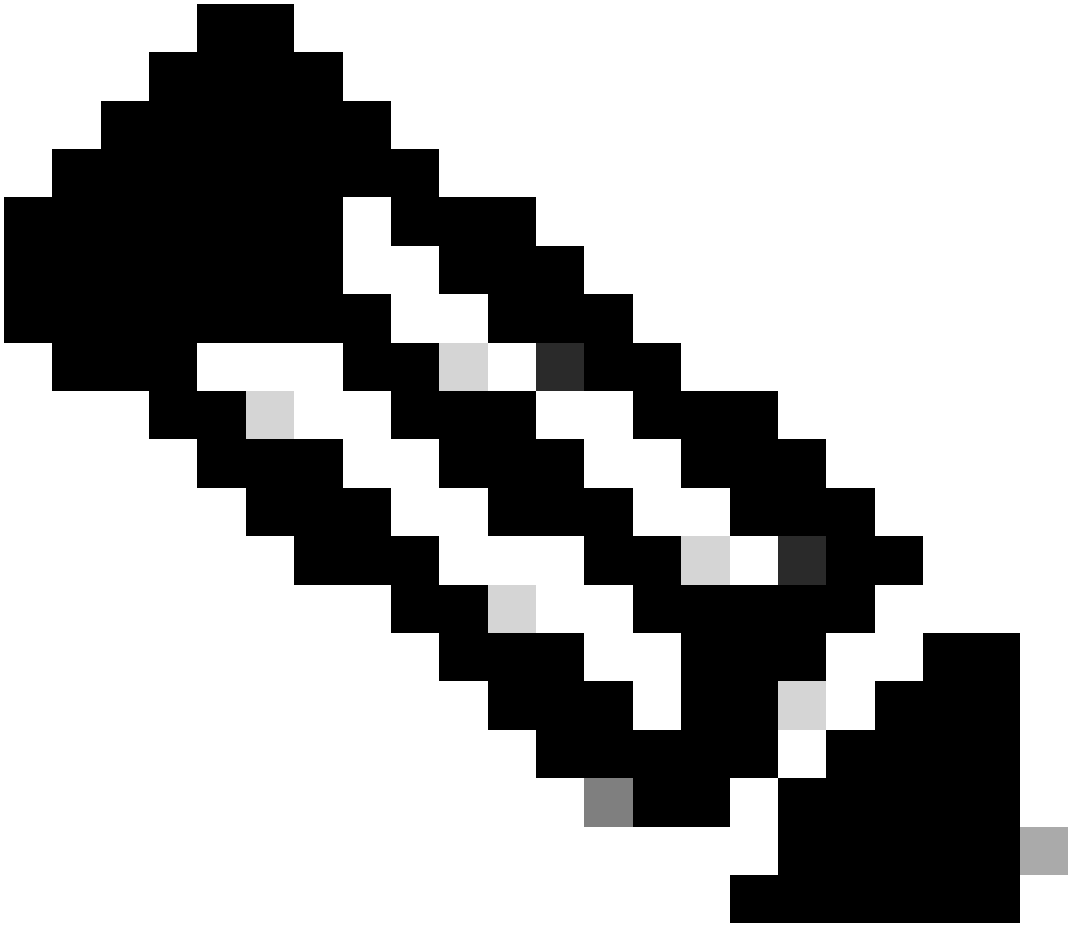
```
nodeInfo_url = "https://<CatalystCenterIP>/dna/intent/api/v1/nodes-config"
nodeInfo_headers = {
    'X-Auth-Token': token
}

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

La variable nodeInfo_url se creó para almacenar la dirección URL de nuestra API. La variable nodeInfo_encabezados almacena los encabezados de nuestra API. En este caso, ‘X-Auth-Token:’ y la variable de token se pasaron como parámetros para autenticar la solicitud correctamente al Catalyst Center. Por último, la variable nodeInfoResponse almacena la respuesta de la API.

Para validar la respuesta recibida, puede utilizar la función print().

Ejemplo de salida:

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



Nota: En caso de que se esté utilizando un certificado autofirmado en Catalyst Center, la solicitud de la API puede fallar con el siguiente error:

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

Para solucionar este problema, debe agregar el parámetro verify a la solicitud como False. Esto suprime la verificación del certificado SSL del terminal (Catalyst Center).

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

La respuesta recibida de la API puede ser difícil de leer. Con el módulo json(), la respuesta se puede imprimir en una cadena más legible. En primer lugar, la respuesta de la API debe cargarse

en un objeto JSON mediante la función json.loads() seguida de la función json.dumps():

```
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 dum
```

json.dumps: Esta función devuelve el objeto JSON tomado como parámetro en una cadena con formato JSON.

sangría: Este parámetro define el nivel de sangría de la cadena con formato JSON.

Ejemplo de salida:

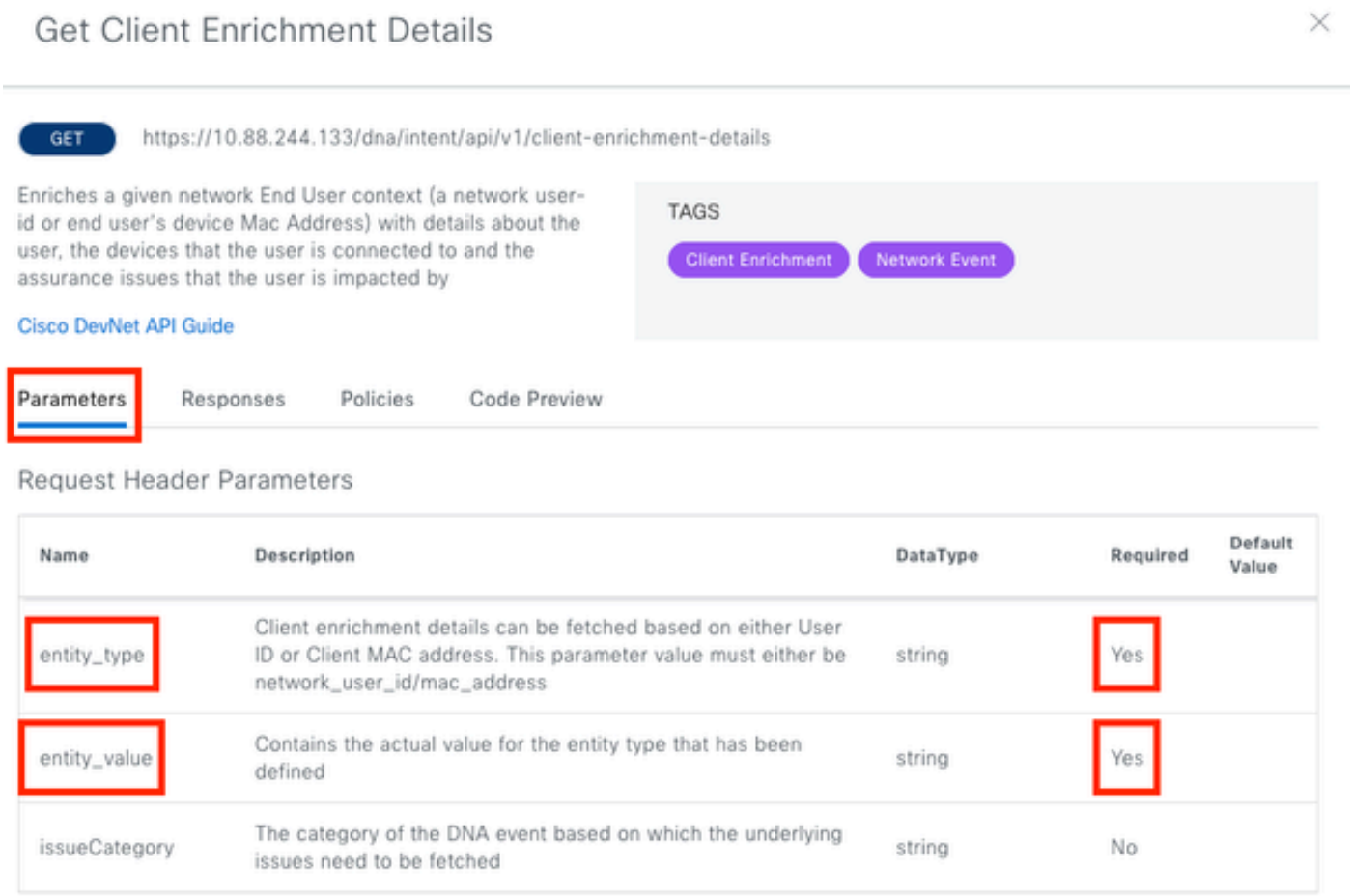
```
{
  "response": {
    "nodes": [
      {
        "name": "X.X.X.X",
        "id": "ea5dbec1-fbb6-4339-9242-7694eb1xXxX",
        "network": [
          {
            "slave": [
              "enp9s0"
            ],
            "lacp_supported": true,
            "intra_cluster_link": false,
!---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

API con parámetros de encabezado

Hay algunas API que requieren que se envíen algunos parámetros en el encabezado para que funcionen como se espera. En este caso, se prueba la API Get Client Enrichment Details.

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

Para verificar qué parámetros de encabezados son necesarios para que la API funcione según lo esperado, navegue hasta Plataforma > Developer Toolkit > API > Obtener detalles de enriquecimiento de cliente y haga clic en el nombre de la API. Se abre una nueva ventana y, en la opción Parameters, se muestran los parámetros de encabezados necesarios para que funcione la API.



En este caso, para el parámetro entity_type, según la descripción, el valor puede ser network_user_id o mac_address y el parámetro entity_value debe contener el valor para el tipo de entidad que se ha definido.

Para continuar, se definen dos nuevas variables, entity_type y entity_value, con sus valores correspondientes:

```
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
```

También se crean nuevas variables para realizar la llamada a la API. La URL de la llamada API se almacena en la variable userEnrichment_url. Los encabezados se almacenan en la variable userEnrichmentHeaders. La respuesta recibida se almacena en la variable userEnrichmentResponse.

```
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)
```

Como puede ver, en userEnrichmentHeaders, las variables entity_type y entity_value se pasaron como parámetros de encabezado para la llamada API, junto con la variable token.

Para validar la respuesta recibida, utilice la función print().

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

Ejemplo de salida:


```
[ {
  "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
```


API con parámetros de consulta

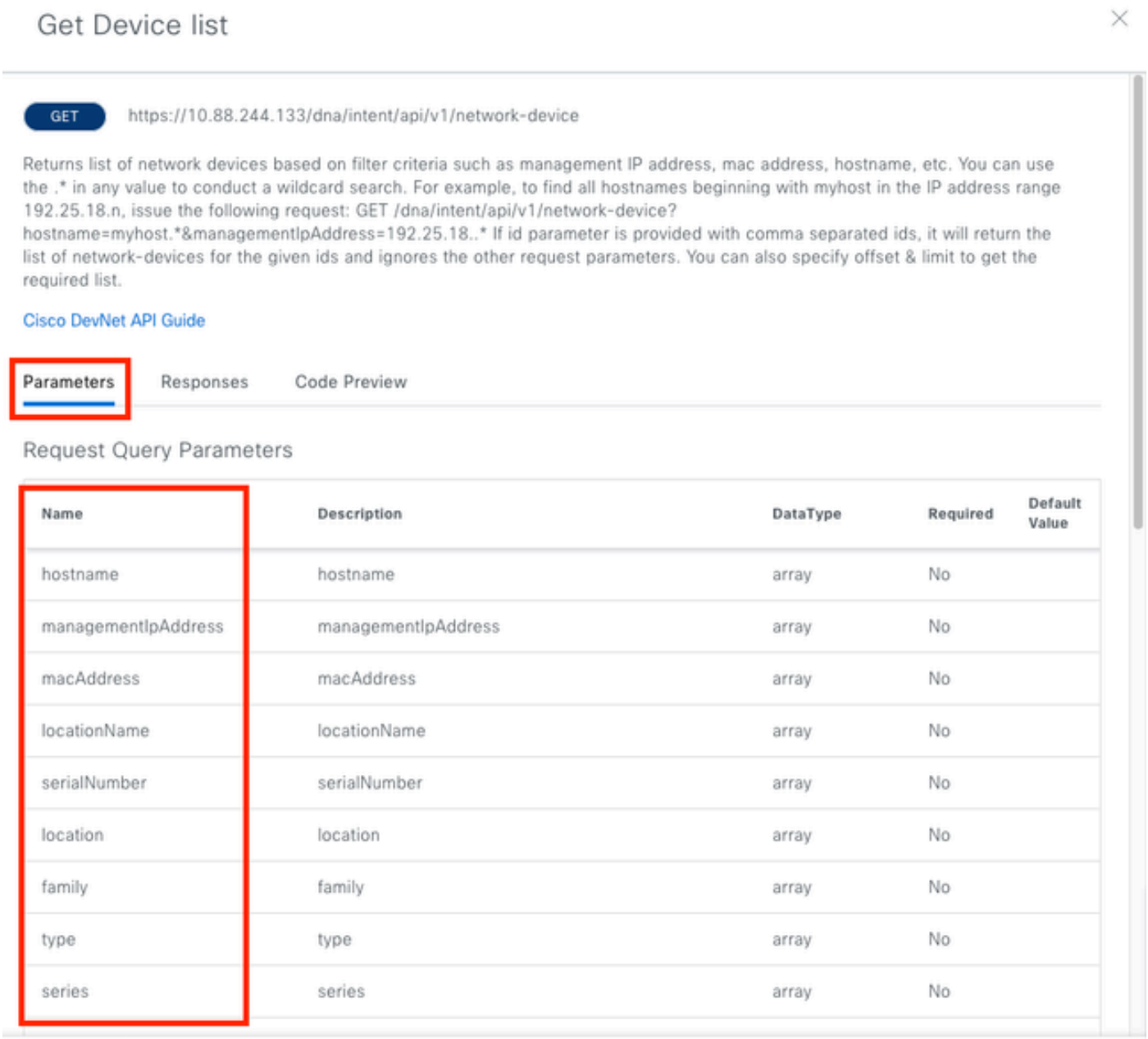
Los parámetros de consulta se pueden utilizar para filtrar un número específico de resultados devueltos por una API. Estos parámetros se agregan a la URL de la API.

Se prueba la llamada API Get Device List.

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

La API Get Device List devuelve una lista de todos los dispositivos que se agregan en Catalyst Center. Si se solicitan detalles para un dispositivo específico, los parámetros de consulta pueden ayudar a filtrar información específica.

Para verificar qué parámetros de consulta están disponibles para la API, navegue hasta Platform > Developer Toolkit > APIs > Get Device List y haga clic en el nombre de la API. Se abre una nueva ventana y, en la opción Parameters, se muestran los parámetros de consulta disponibles para la API.



En este ejemplo, se utilizan los parámetros de consulta managementIpAddress y serialNumber (tenga en cuenta que no es necesario utilizar todos los parámetros de consulta para la llamada API). Proceda a crear y asignar los valores correspondientes para ambos parámetros de consulta.

```
managementIpAddress = '10.82.143.250'
serialNumber = 'FDO25160X9L'
```

Como se mencionó anteriormente, los parámetros de consulta se agregan en la URL de la API, específicamente al final de la misma, utilizando un '?' seguido de los parámetros de consulta.

En caso de que se vayan a utilizar varios parámetros de consulta, se coloca un signo '&' entre ellos para formar lo que se denomina una cadena de consulta.

El siguiente ejemplo muestra cómo agregar los parámetros de consulta a la variable deviceListUrl que almacena la dirección URL de la llamada API.

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

Observe que las variables creadas anteriormente se agregaron a la cadena URL. En otras palabras, la cadena completa de la URL tiene el siguiente aspecto:

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

Continúe con la llamada API, la variable deviceListHeaders se crea para almacenar los encabezados API junto con la variable token pasada como parámetro y la variable deviceListResponse almacena la respuesta API.

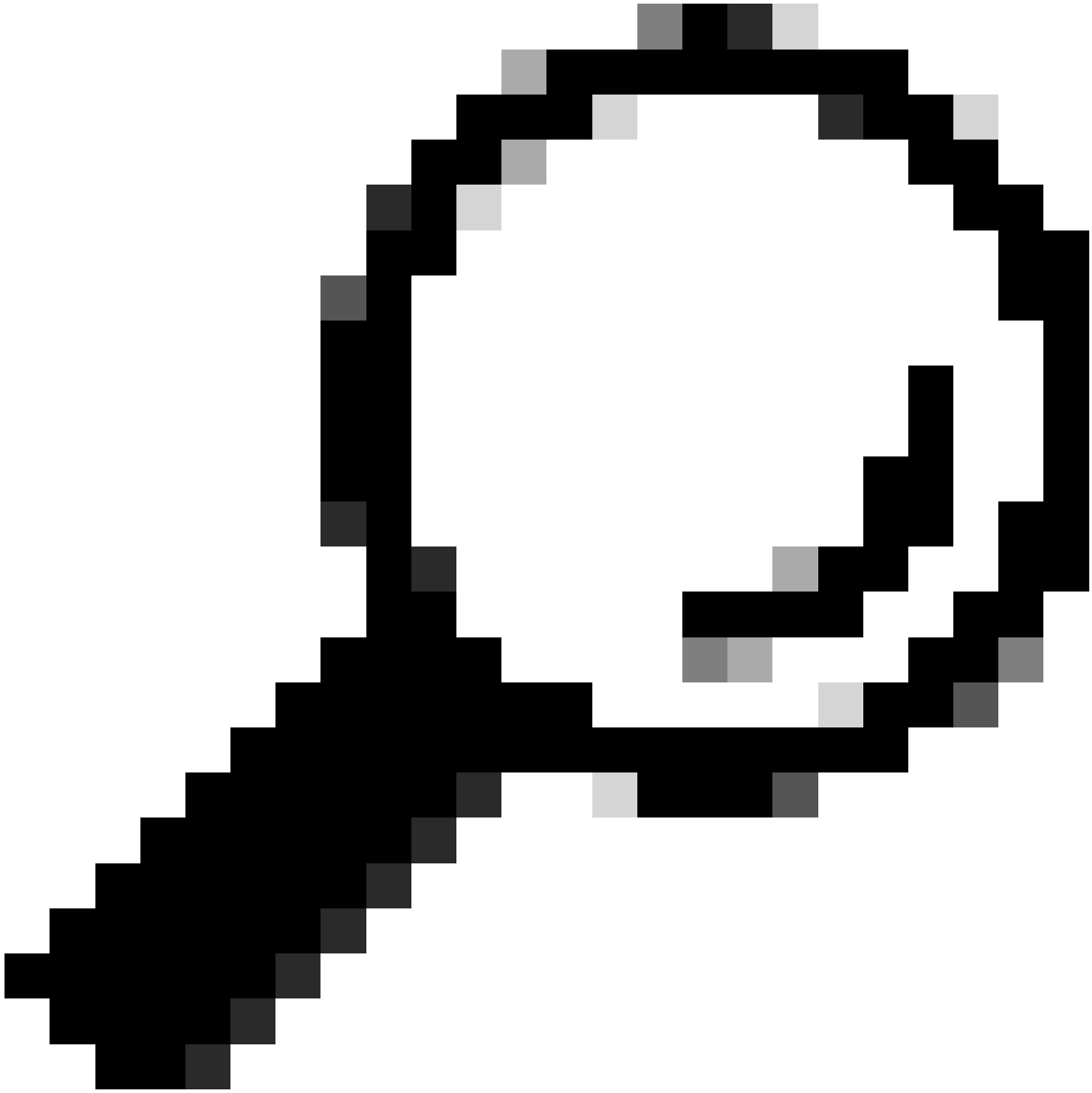
```
deviceListHeaders = {
            'X-Auth-Token': token,
        }

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

Para validar la respuesta recibida, puede utilizar la función print().

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

Ejemplo de salida:

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



Sugerencia: para imprimir la respuesta de una manera más legible, puede utilizar las funciones json.loads() y json.dumps() descritas en la sección Prueba de la API.

Acerca de esta traducción
Cisco ha traducido este documento combinando la traducción automática y los recursos humanos a fin de ofrecer a nuestros usuarios en todo el mundo contenido en su propio idioma.

Tenga en cuenta que incluso la mejor traducción automática podría no ser tan precisa como la proporcionada por un traductor profesional.

Cisco Systems, Inc. no asume ninguna responsabilidad por la precisión de estas traducciones y recomienda remitirse siempre al documento original escrito en inglés (insertar vínculo URL).