Uso delle API Catalyst Center con Python

Sommario

Introduzione

Questo documento descrive come utilizzare le diverse API disponibili su Cisco Catalyst Center con Python.

Prerequisiti

Requisiti

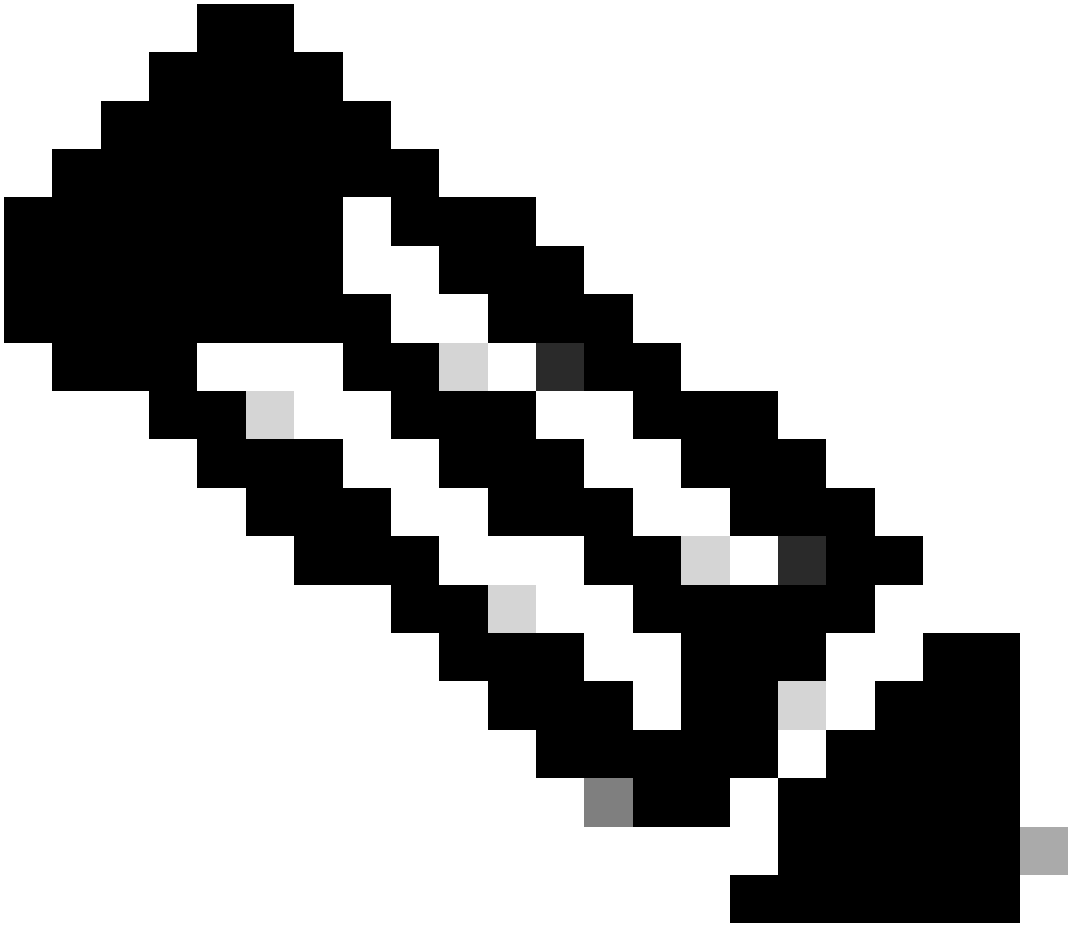
Conoscenze di base su:

- Cisco Catalyst Center
- API
- Python

Componenti usati

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

Le informazioni discusse in questo documento fanno riferimento a dispositivi usati in uno specifico ambiente di emulazione. Su tutti i dispositivi menzionati nel documento la configurazione è stata ripristinata ai valori predefiniti. Se la rete è operativa, valutare attentamente eventuali conseguenze derivanti dall'uso dei comandi.

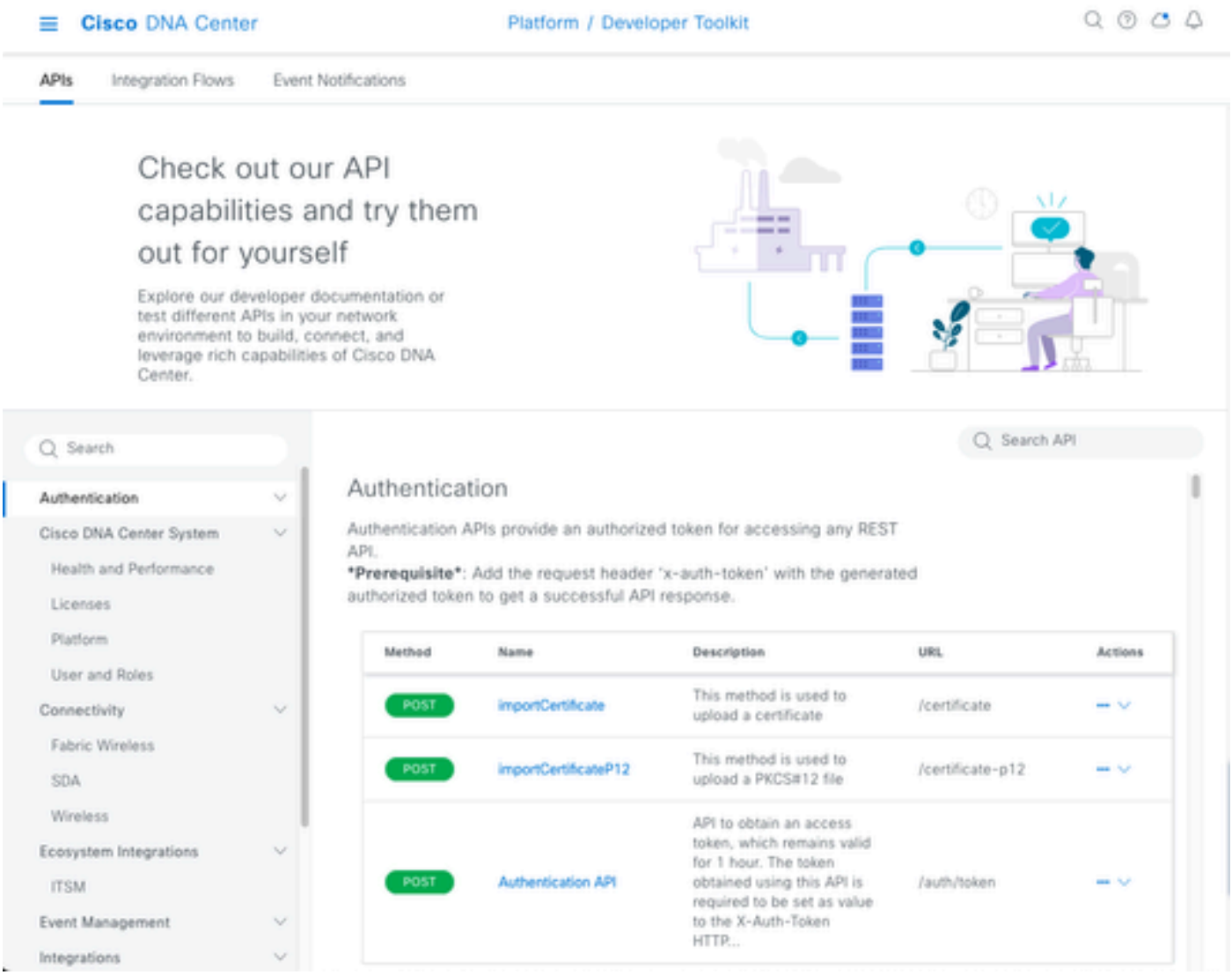


Nota: il Cisco Technical Assistance Center (TAC) non fornisce supporto tecnico per Python. In caso di problemi con Python, contattare il supporto Python per assistenza tecnica.

Configurazione

Panoramica

Cisco Catalyst Center dispone di molte API. Per verificare quali API è possibile utilizzare, in Catalyst Center passare a Piattaforma > Developer Toolkit > API.

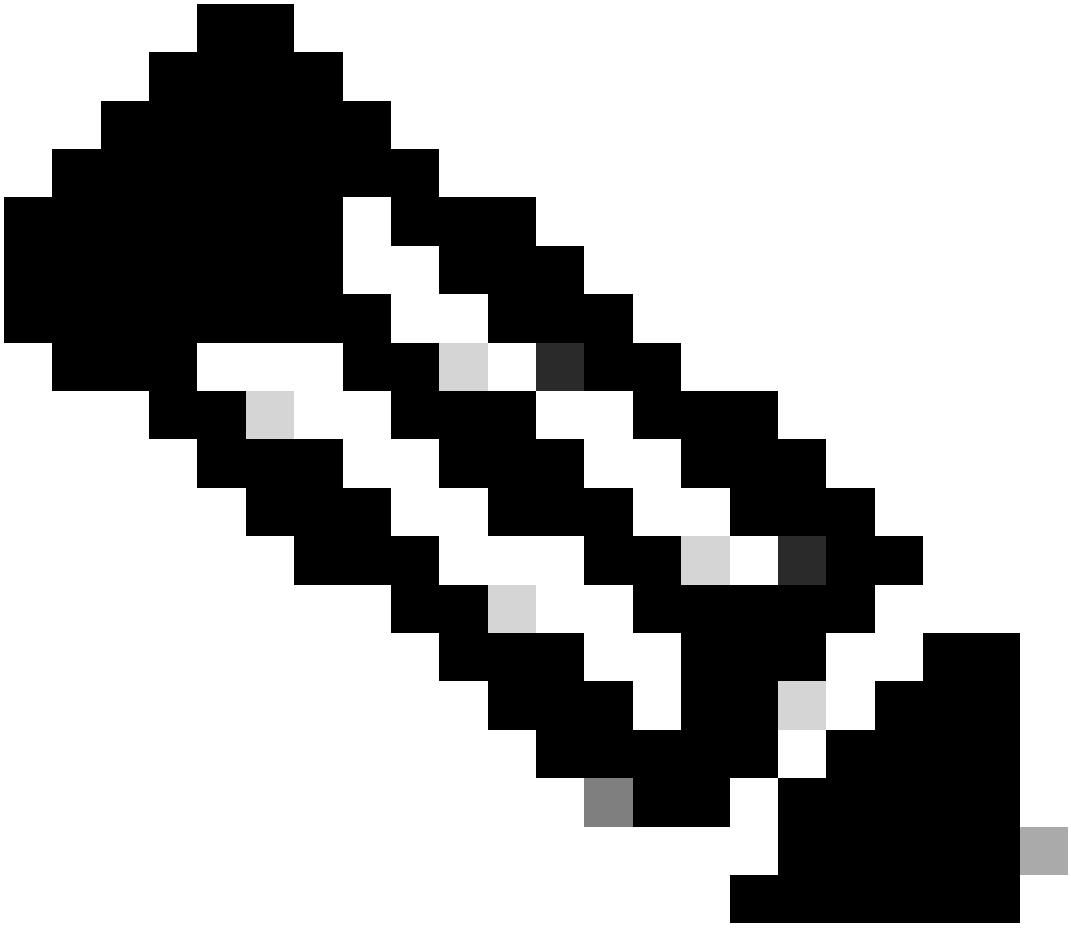


Pagina API di Catalyst Center

Ogni API ha uno scopo specifico, a seconda delle informazioni o dell'azione da eseguire su Catalyst Center. Affinché le API funzionino, come prerequisito, è necessario utilizzare un token per autenticarsi correttamente in Catalyst Center e ottenere una risposta API corretta. Il token identifica di conseguenza i privilegi per il chiamante REST.

È inoltre importante identificare i componenti che compongono un'API che sono i seguenti:

- URL: endpoint che fornisce l'accesso a una risorsa specifica.
- Metodo: tutte le API devono includere un metodo. Definisce l'azione o l'operazione che il client desidera eseguire sull'endpoint specifico. Esempi: POST, GET, PUT, DELETE.
- Intestazione: fornisce informazioni aggiuntive sulla richiesta in formato coppia chiave-valore. L'intestazione di autorizzazione, ad esempio, fornisce un metodo di autenticazione che utilizza le credenziali.
- Parametri: variabili che forniscono istruzioni specifiche per l'endpoint tramite l'API. I parametri possono far parte dell'URL dell'endpoint.
- Payload: dati che devono essere inviati all'endpoint durante la chiamata API.

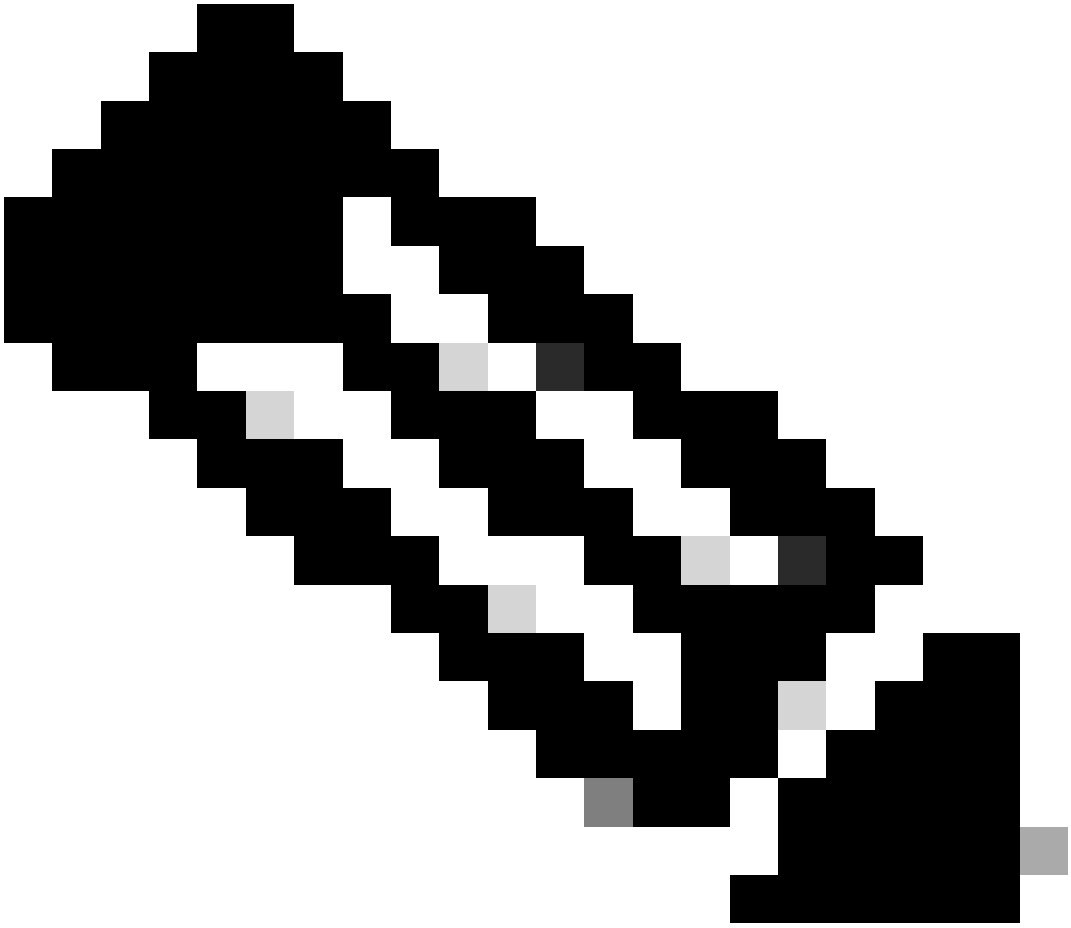


Nota: per informazioni più dettagliate su ciascuna API disponibile in Catalyst Center, consultare la guida di riferimento per le API.

Moduli

Moduli Python utilizzati:

- richieste:questo modulo consente di inviare richieste HTTP/1.1 a URL specifici. Per ulteriori informazioni sul modulo, consultare la guida al modulo di richiesta.
- base64: fornisce funzioni di codifica e decodifica. Per ulteriori informazioni sul modulo, consultare la guida del modulo base64.
- json: questo modulo consente di ottenere dati specifici dalla risposta delle API. Per ulteriori informazioni sul modulo, consultare la guida al modulo json.



Nota: per ulteriori informazioni su come installare i moduli Python, consultare la documentazione sull'installazione dei moduli Python.

Genera token

Per generare un nuovo token, è necessario utilizzare l'API Authentication API.

API di autenticazione:

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

È importante ricordare che il token generato è valido per 1 ora. Dopo 1 ora, deve essere generato un nuovo token utilizzando la stessa API menzionata sopra.

In un nuovo file Python, importare i moduli (request, base64 e json) seguiti dalla creazione di

quattro variabili:

```
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"
```

L'API di autenticazione supporta Basic Auth come token di autorizzazione nell'intestazione. L'autenticazione di base è un metodo che può essere utilizzato per autenticare un endpoint, fornendo un nome utente e una password separati da due punti (nomeutente:password). Entrambi i valori sono codificati in base64 e l'endpoint decodifica le credenziali di accesso e verifica se l'utente può accedere o meno.

Per creare la stringa Base64 per il nome utente e la password, viene utilizzato il modulo base64. A tale scopo, viene utilizzata la funzione b64encode.

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

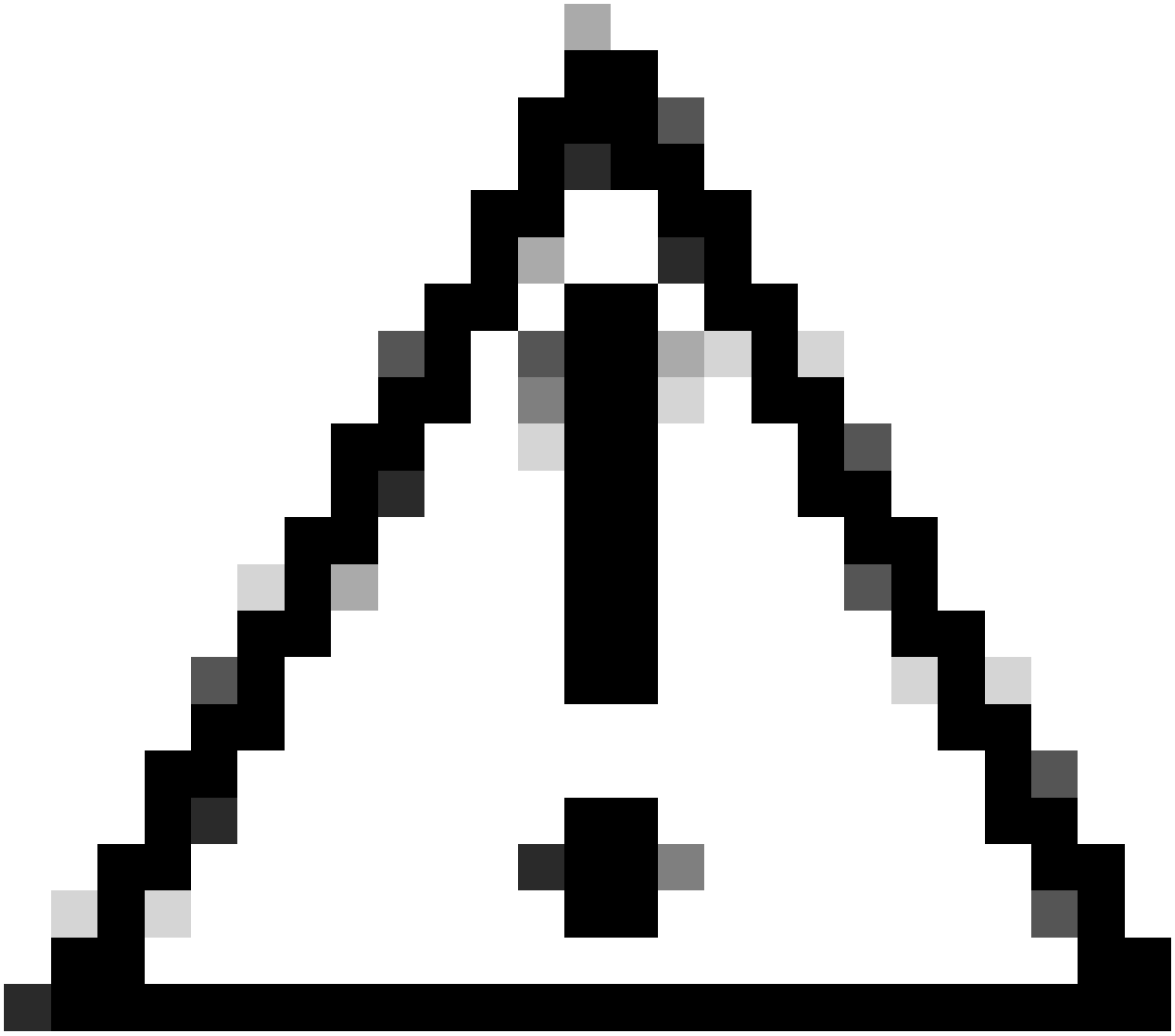
Dal codice sopra riportato, è stata creata una variabile byte_string utilizzando la funzione '.encode("ascii")'. Ciò è dovuto al fatto che la funzione base64.b64encode richiede un oggetto di tipo byte. Si noti inoltre che le variabili user e password sono state utilizzate per mantenere il formato stringa 'user:password'. Infine, è stata creata una stringa di byte con codifica Base64 con l'utente e la password. Utilizzando il metodo 'decode()', il valore è stato convertito in oggetto str.

Per verificarlo, è possibile stampare il valore della variabile authorizationBase64:

```
print(authorizationBase64)
```

Esempio di output:

```
am9yZ2QhbDI6Sm9yZ2VhbDXxXxXx
```



Attenzione: base64 non è un algoritmo di crittografia. Non deve essere utilizzata per scopi di sicurezza. L'API di autenticazione supporta inoltre la crittografia con chiave AES come token di autorizzazione nell'intestazione, offrendo una maggiore sicurezza.

Ora che è stata creata una stringa con codifica base64 utilizzando l'utente e la password per l'autenticazione in Catalyst Center, è possibile procedere con la chiamata API di autenticazione API utilizzando le richieste del modulo. La funzione request consente inoltre di ottenere un oggetto risposta contenente il testo della richiesta.

Sintassi del metodo:

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

**kwargs indica qualsiasi parametro passato nella richiesta, ad esempio cookie, agenti utente, payload, intestazioni e così via.

L'API di autenticazione specifica che il metodo è POST, l'URL è "/dna/system/api/v1/auth/token" ed è necessario specificare l'autenticazione di base nell'intestazione.

Queste variabili vengono create per utilizzarle per la funzione request().

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

Per la variabile headers sono stati specificati due elementi. Il primo è il tipo di contenuto, che specifica il tipo di supporto della risorsa inviata all'endpoint. Ciò consente all'endpoint di analizzare ed elaborare in modo accurato i dati. Il secondo è Authorization, che in questo caso, la variabile authorizationBase64 (in cui è archiviata la stringa in base64) viene inviata come parametro per l'autenticazione in Catalyst Center.

A questo punto, continuare a utilizzare la funzione request() per eseguire la chiamata API. Nel codice successivo viene illustrata la sintassi della funzione:

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

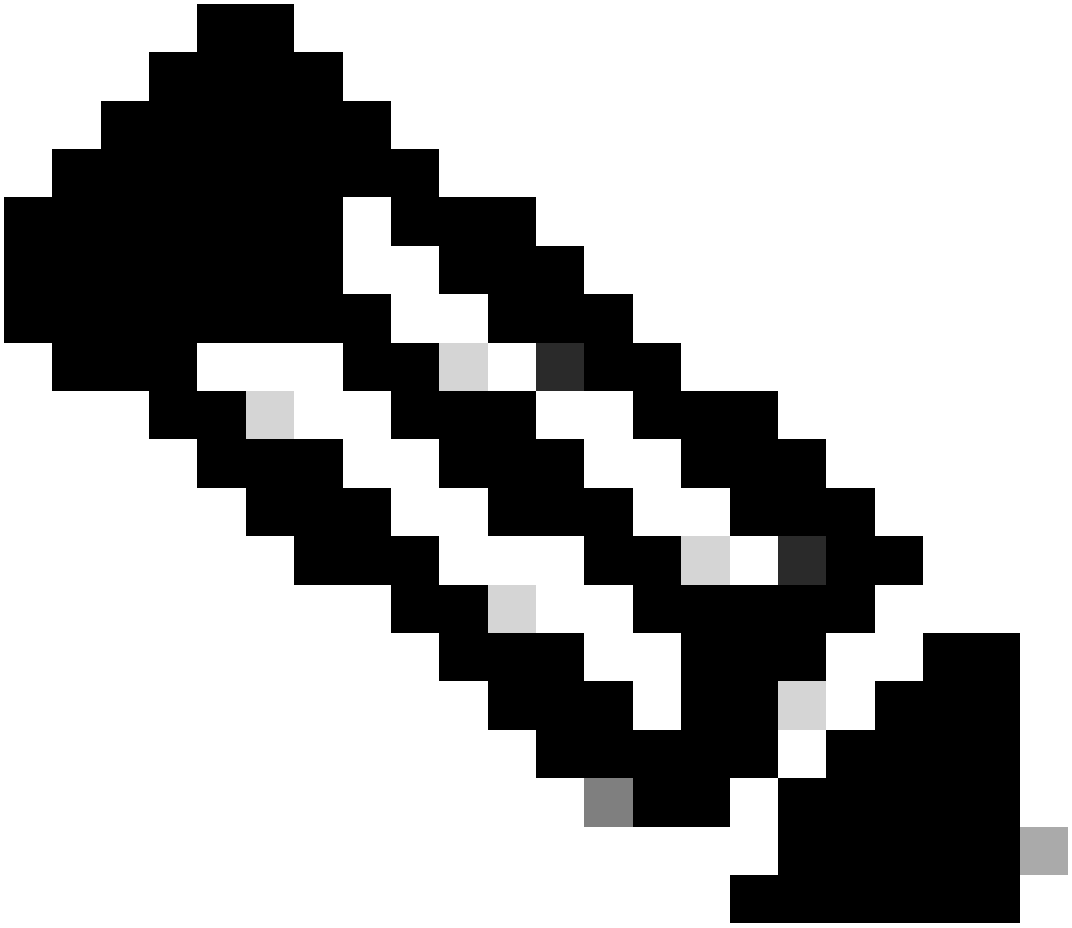
La variabile di risposta è stata creata per archiviare i dati della chiamata API effettuata.

Per stampare la risposta ottenuta, utilizzare la funzione print insieme al metodo text() nella variabile response. Il metodo text() genera un oggetto str con la risposta ricevuta da Catalyst Center.

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

Esempio:

```
{“Token”:“eyJhbGciOiJSUzI1NiIsInR5JKO9s1zVmNjkONjhkNTFhNDJ1ZWeLCU291cmNlIjoiaW5OZXJuYWwiLCW2vMPUbUOJNlq
!--- Output is supressed
```



Nota: se Catalyst Center utilizza un certificato autofirmato, la richiesta API può avere esito negativo con il seguente errore:

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

Per risolvere il problema, aggiungere il parametro verify come False alla funzione di richiesta. In questo modo viene ignorata la verifica del certificato SSL dall'endpoint (Catalyst Center).

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

Dalla risposta ricevuta dalla chiamata di autenticazione API, si noti che la struttura è simile a un

dizionario in Python, ma è un oggetto str.

Per convalidare il tipo di un oggetto, utilizzare la funzione type().

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

che restituisce l'output successivo:

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

A fini pratici, solo il valore del token deve essere estratto dalla risposta ricevuta dall'API, non l'intera stringa, poiché, per utilizzare le altre API Catalyst Center, solo il token deve essere passato come parametro.

Poiché la risposta ricevuta dalla chiamata API ha una struttura simile a un dizionario in Python ma il tipo di oggetto è str, l'oggetto deve essere convertito in un dizionario utilizzando il modulo json. In questo modo il valore del token viene estratto dall'intera stringa ricevuta dall'API.

A tale scopo, la funzione json.loads() converte la stringa in un dizionario per estrarre in seguito solo il valore del token e assegnarlo direttamente alla nostra variabile token.

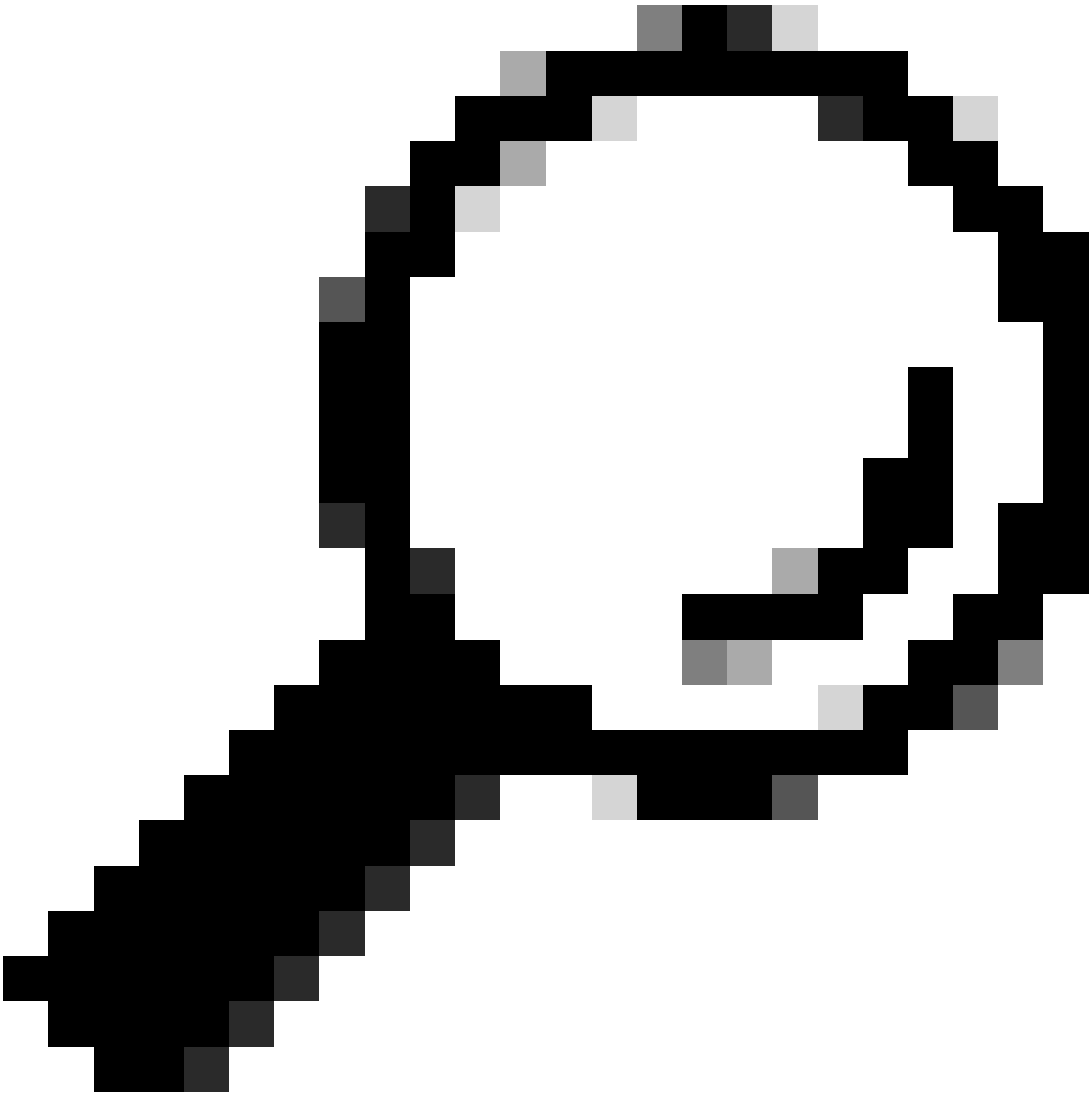
```
token = json.loads(response.text)  # Converting the response.text string value into a dictionary (It is
token = (token["Token"])  # Extracting just the token value by specifying the key as a parameter.
```

Per verificare che il valore della variabile token sia solo il token, procedere con la stampa.

```
print(token)
```

Esempio:

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



Suggerimento: poiché ogni token generato scade tra un'ora per impostazione predefinita, è possibile creare e chiamare un metodo Python che contiene il codice per generare un token ogni volta che un token scade, senza dover eseguire l'intero programma semplicemente chiamando il metodo creato.

Test di un'API

Una volta assegnato correttamente il token alla variabile token, è possibile utilizzare le API Catalyst Center disponibili.

In questo caso, viene verificata l'API di riepilogo configurazione dei nodi Cisco DNA Center.

Riepilogo della configurazione dei nodi Cisco DNA Center

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

Questa API fornisce dettagli sulla configurazione corrente di Catalyst Center, ad esempio il server NTP configurato, il nome del nodo, il collegamento all'interno del cluster, la modalità LACP e così via.

L'API di riepilogo della configurazione dei nodi Cisco DNA Center specifica, in questo caso, che il metodo utilizzato è GET, l'URL è "/dna/intent/api/v1/nodes-config" e, poiché la stringa del token è stata estratta e assegnata alla variabile, questa volta il token viene passato come variabile nell'intestazione della chiamata API come ‘X-Auth-Token’: seguito dal token.

In questo modo viene autenticata la richiesta al Catalyst Center per ogni chiamata API eseguita. Ricorda che ogni token dura un'ora. Dopo 1 ora, è necessario generare un nuovo token per continuare a eseguire chiamate API al Catalyst Center.

Procedere con la creazione delle variabili per il test dell'API:

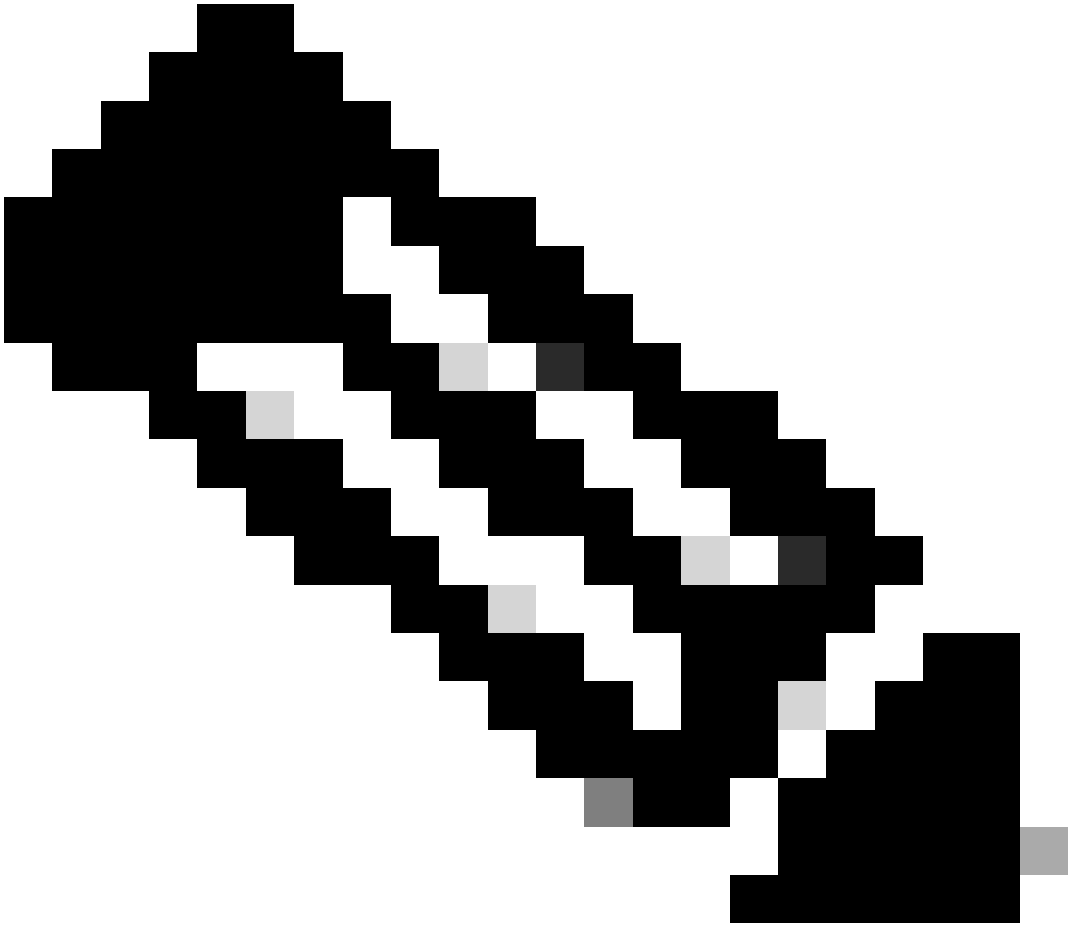
```
nodeInfo_url = "https://<CatalystCenterIP>/dna/intent/api/v1/nodes-config"
nodeInfo_headers = {
    'X-Auth-Token': token
}

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

La variabile nodeInfo_url è stata creata per memorizzare l'URL dell'API. La variabile nodeInfo_headers memorizza le intestazioni per l'API. In questo caso, ‘X-Auth-Token:’ e la variabile token sono stati passati come parametri per autenticare correttamente la richiesta in Catalyst Center. Infine, la variabile nodeInfoResponse memorizza la risposta dell'API.

Per convalidare la risposta ricevuta, è possibile utilizzare la funzione print().

Esempio:

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



Nota: se in Catalyst Center viene utilizzato un certificato autofirmato, la richiesta API può non riuscire con l'errore successivo:

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

Per risolvere il problema, aggiungere il parametro verify come False alla richiesta. In questo modo viene eliminata la verifica del certificato SSL dall'endpoint (Catalyst Center).

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

La risposta ricevuta dall'API può essere di difficile lettura. Utilizzando il modulo json(), la risposta può essere stampata in una stringa più leggibile. In primo luogo, la risposta API deve essere

caricata in un oggetto JSON utilizzando la funzione json.loads() seguita dalla funzione 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: questa funzione restituisce l'oggetto JSON preso come parametro in una stringa in formato JSON.

rientro: questo parametro definisce il livello di rientro per la stringa in formato JSON.

Esempio:

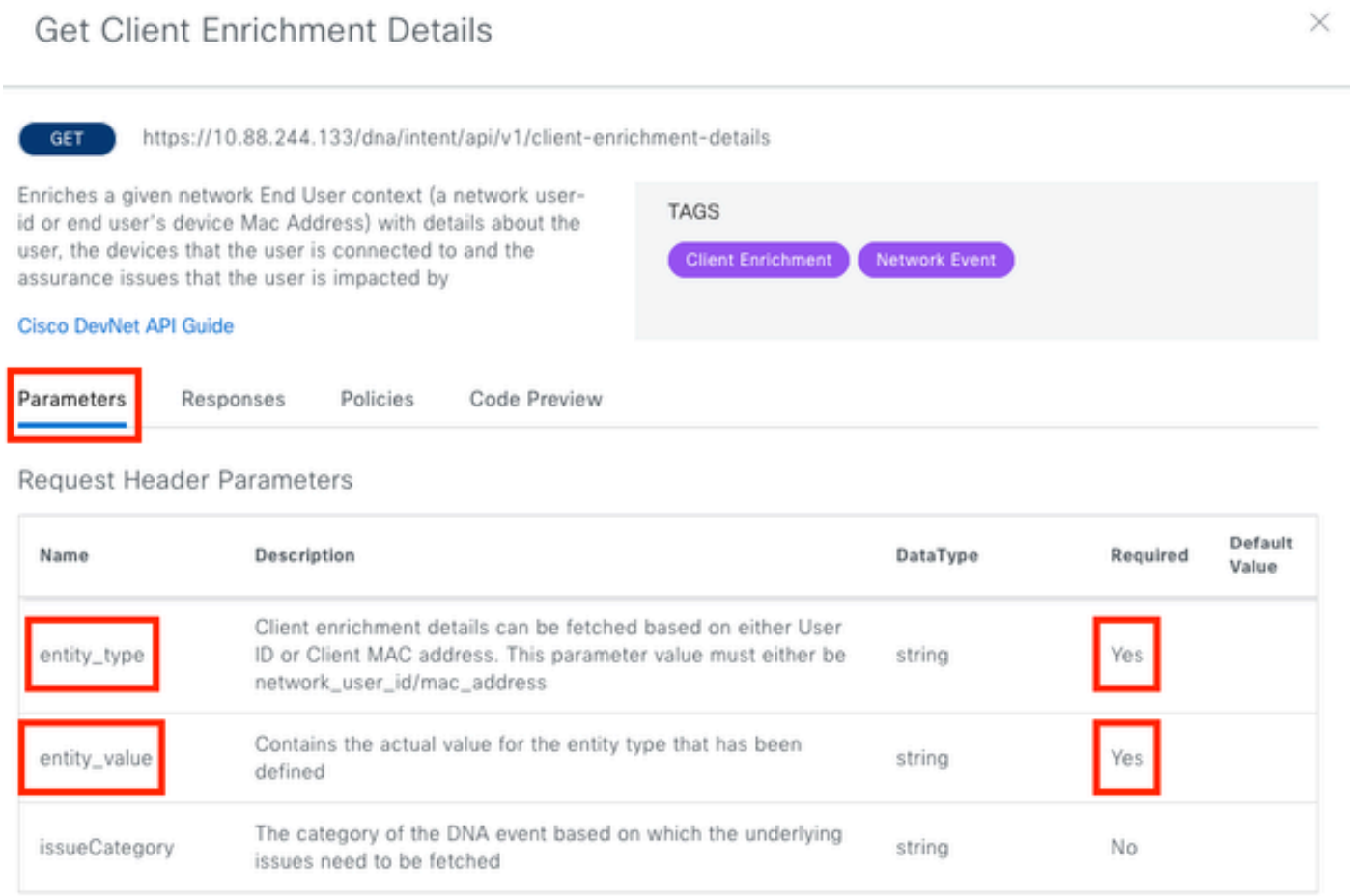
```
{
  "response": {
    "nodes": [
      {
        "name": "X.X.X.X",
        "id": "ea5dbec1-fbb6-4339-9242-7694eb1xXxX",
        "network": [
          {
            "slave": [
              "enp9s0"
            ],
            "lacp_supported": true,
            "intra_cluster_link": false,
!---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

API con parametri di intestazione

Alcune API richiedono l'invio di alcuni parametri nell'intestazione per funzionare come previsto. In questo caso, viene verificata l'API Get Client Enrichment Details.

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

Per verificare quali parametri delle intestazioni sono necessari affinché l'API funzioni come previsto, passare a Piattaforma > Toolkit sviluppatori > API > Ottieni dettagli di arricchimento client e fare clic sul nome dell'API. Viene aperta una nuova finestra e sotto l'opzione Parametri vengono visualizzati i parametri delle intestazioni necessari per il funzionamento dell'API.



In questo caso, per il parametro entity_type, in base alla descrizione, il valore può essere network_user_id o mac_address e il parametro entity_value deve contenere il valore per il tipo di entità definito.

Per procedere, vengono definite due nuove variabili, entity_type e entity_value con i valori corrispondenti:

```
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
```

Vengono inoltre create nuove variabili per eseguire la chiamata API. L'URL della chiamata API è archiviato nella variabile userEnrichment_url. Le intestazioni vengono archiviate nella variabile userEnrichmentHeaders. La risposta ricevuta viene archiviata nella variabile 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)
```

Come si può vedere, dalle variabili userEnrichmentHeaders, entity_type ed entity_value sono stati passati come parametri di intestazione per la chiamata API, insieme alla variabile token.

Per convalidare la risposta ricevuta, utilizzare la funzione print().

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

Esempio:


```
[ {
  "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 parametri di query

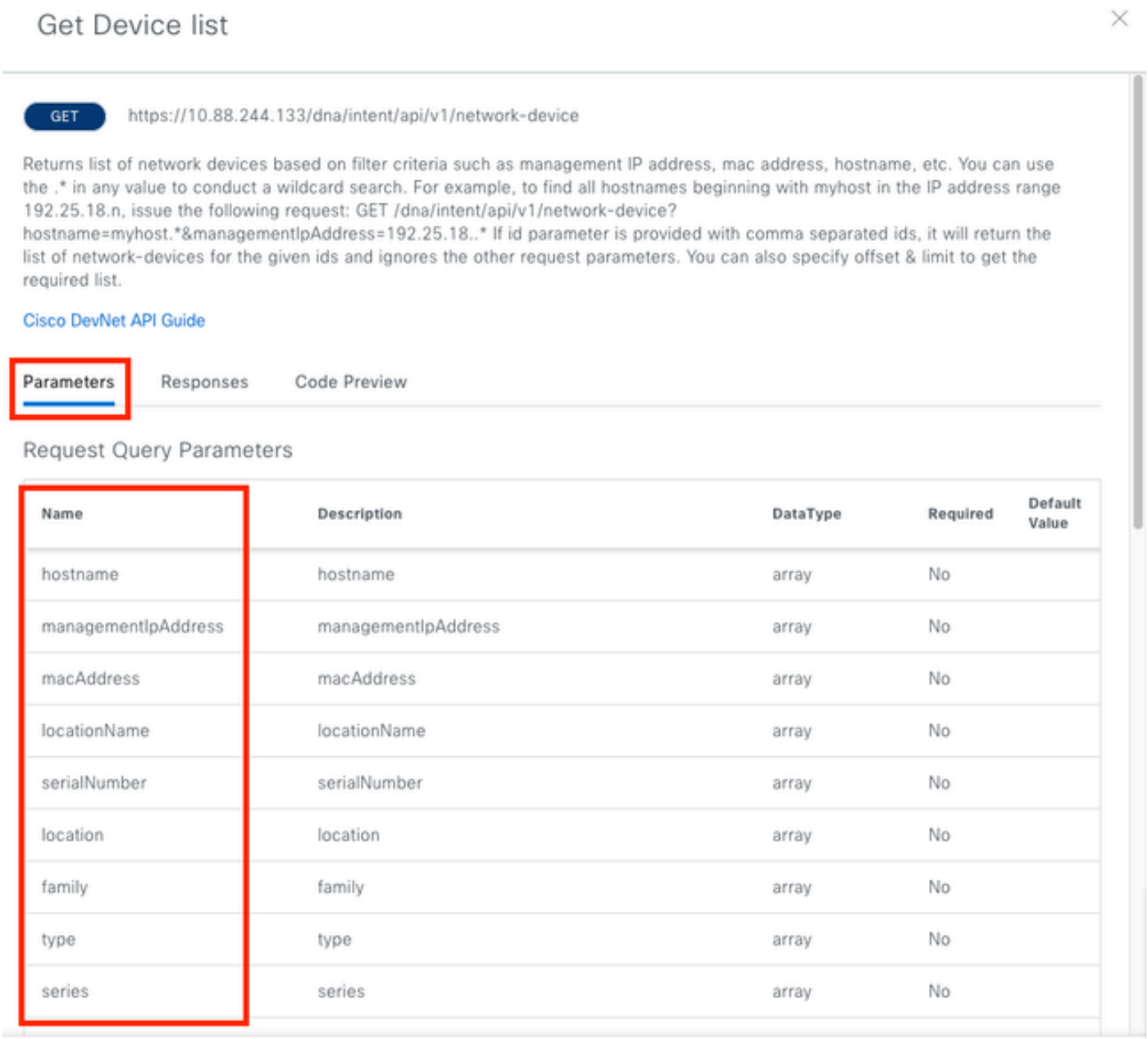
I parametri di query possono essere utilizzati per filtrare un numero specifico di risultati restituiti da un'API. Questi parametri vengono aggiunti all'URL dell'API.

La chiamata API Get Device List è stata testata.

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

L'API Get Device List restituisce un elenco di tutti i dispositivi aggiunti in Catalyst Center. Se sono richiesti i dettagli relativi a una periferica specifica, i parametri di query consentono di filtrare informazioni specifiche.

Per verificare quali parametri di query sono disponibili per l'API, selezionare Piattaforma > Toolkit per sviluppatori > API > Ottieni elenco dispositivi e fare clic sul nome dell'API. Viene aperta una nuova finestra e sotto l'opzione Parametri vengono visualizzati i parametri di query disponibili per l'API.



In questo esempio vengono utilizzati i parametri di query managementIpAddress e serialNumber, tenendo presente che non è necessario utilizzare tutti i parametri di query per la chiamata API. Continuare a creare e assegnare i valori corrispondenti per entrambi i parametri di query.

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

Come accennato in precedenza, i parametri della query vengono aggiunti nell'URL dell'API, in particolare alla fine dell'API, utilizzando un punto interrogativo (?) seguito dai parametri della query.

Se si utilizzano più parametri di query, tra di essi viene posizionato un segno & per formare la stringa di query.

Nell'esempio seguente viene illustrato come aggiungere i parametri della query alla variabile deviceListUrl in cui è archiviato l'URL della chiamata API.

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

Si noti che le variabili create in precedenza sono state aggiunte alla stringa dell'URL. In altre parole, l'intera stringa dell'URL avrà il seguente aspetto:

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

Continuando con la chiamata API, la variabile deviceListHeaders viene creata per memorizzare le intestazioni API insieme alla variabile token passata come parametro e la variabile deviceListResponse memorizza la risposta API.

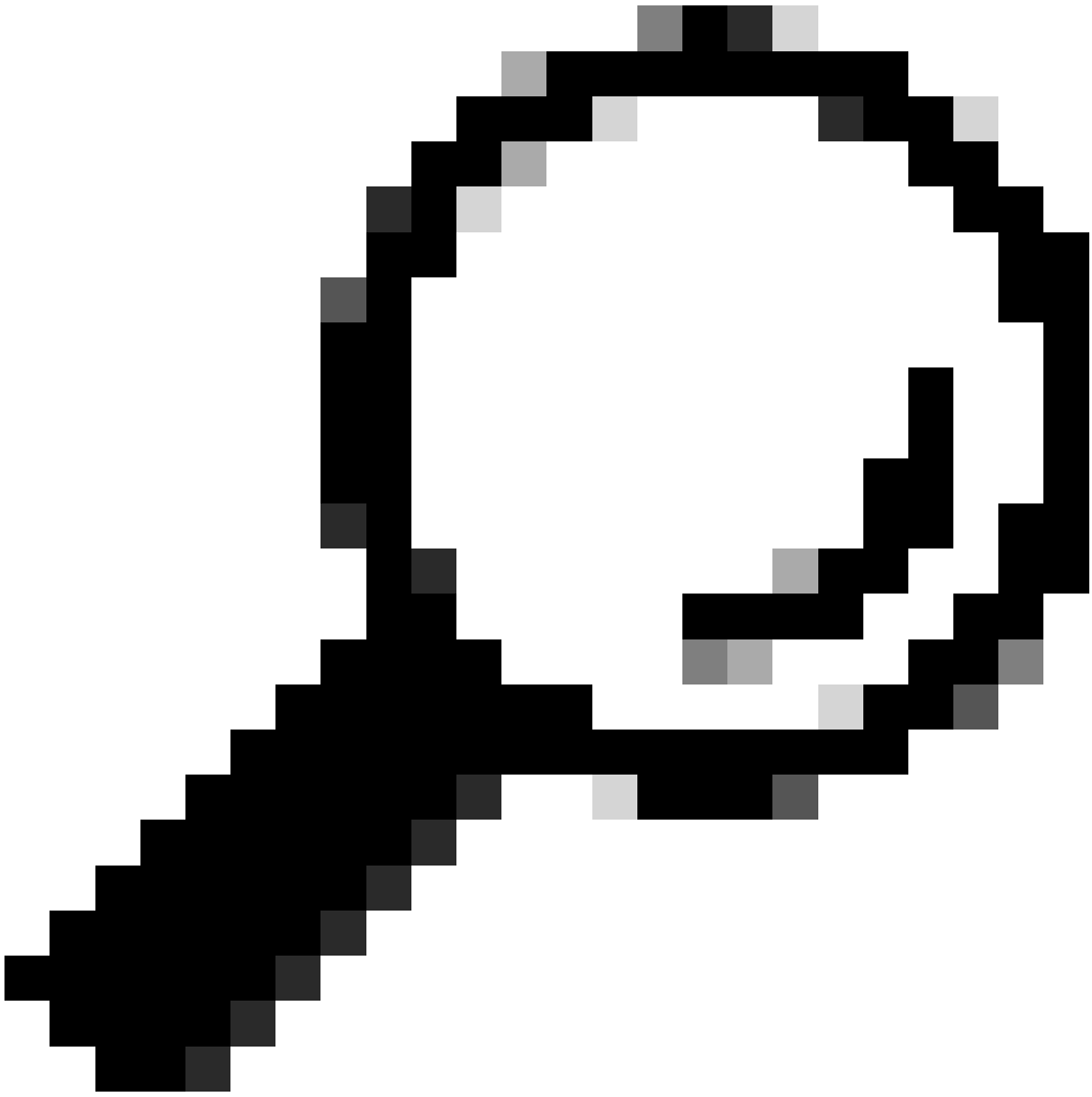
```
deviceListHeaders = {
            'X-Auth-Token': token,
        }

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

Per convalidare la risposta ricevuta, è possibile utilizzare la funzione print().

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

Esempio:

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



Suggerimento: per stampare la risposta in modo più leggibile, è possibile utilizzare le funzioni json.loads() e json.dumps() descritte nella sezione API di test.

Informazioni su questa traduzione

Cisco ha tradotto questo documento utilizzando una combinazione di tecnologie automatiche e umane per offrire ai nostri utenti in tutto il mondo contenuti di supporto nella propria lingua. Si noti che anche la migliore traduzione automatica non sarà mai accurata come quella fornita da un traduttore professionista. Cisco Systems, Inc. non si assume alcuna responsabilità per l’accuratezza di queste traduzioni e consiglia di consultare sempre il documento originale in inglese (disponibile al link fornito).