Catalyst Center API’s met Python gebruiken

Inhoud

Inleiding

Dit document beschrijft hoe u de verschillende API's kunt gebruiken die op Cisco Catalyst Center met Python beschikbaar zijn.

Voorwaarden

Vereisten

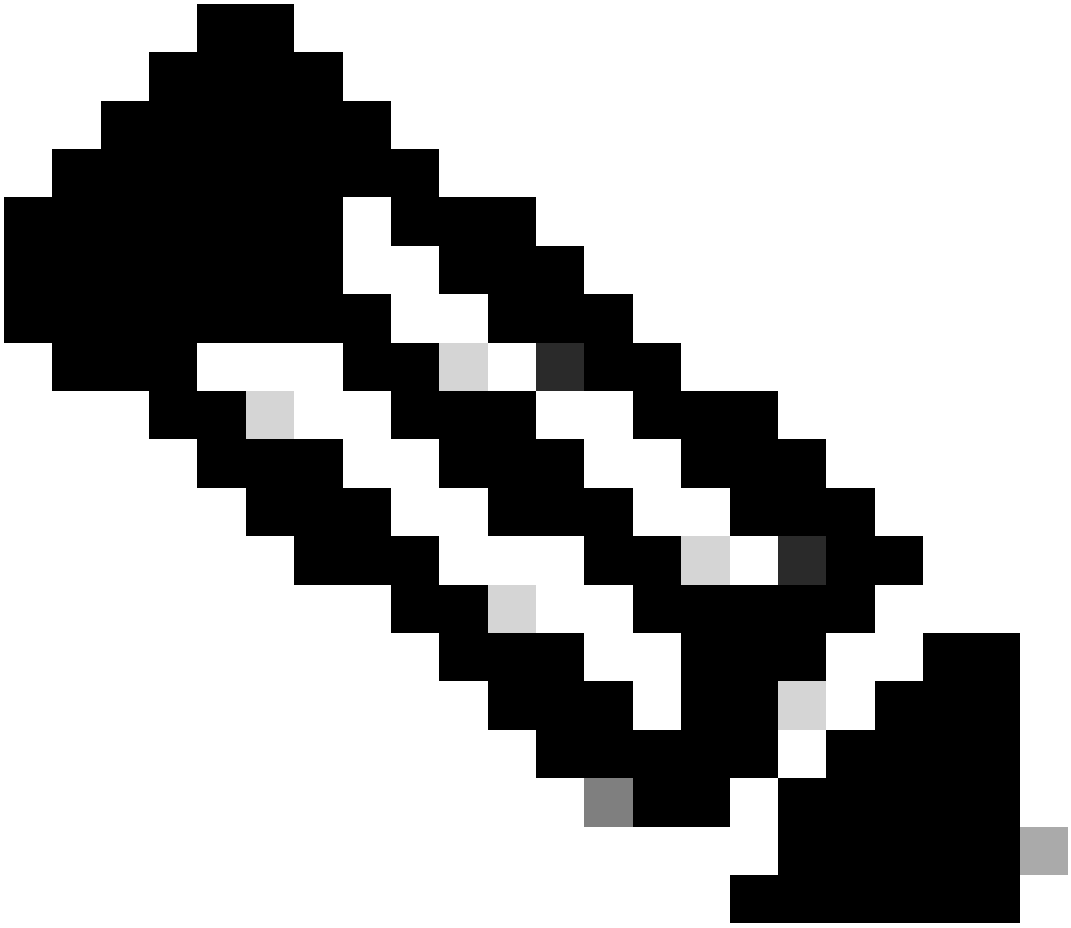
Basiskennis over:

- Cisco Catalyst 6500 Center
- API’s
- Python

Gebruikte componenten

- Cisco Catalyst Center 29.3.5.x
- Python 3.x.x

De informatie in dit document is gebaseerd op de apparaten in een specifieke laboratoriumomgeving. Alle apparaten die in dit document worden beschreven, hadden een opgeschoonde (standaard)configuratie. Als uw netwerk live is, moet u zorgen dat u de potentiële impact van elke opdracht begrijpt.

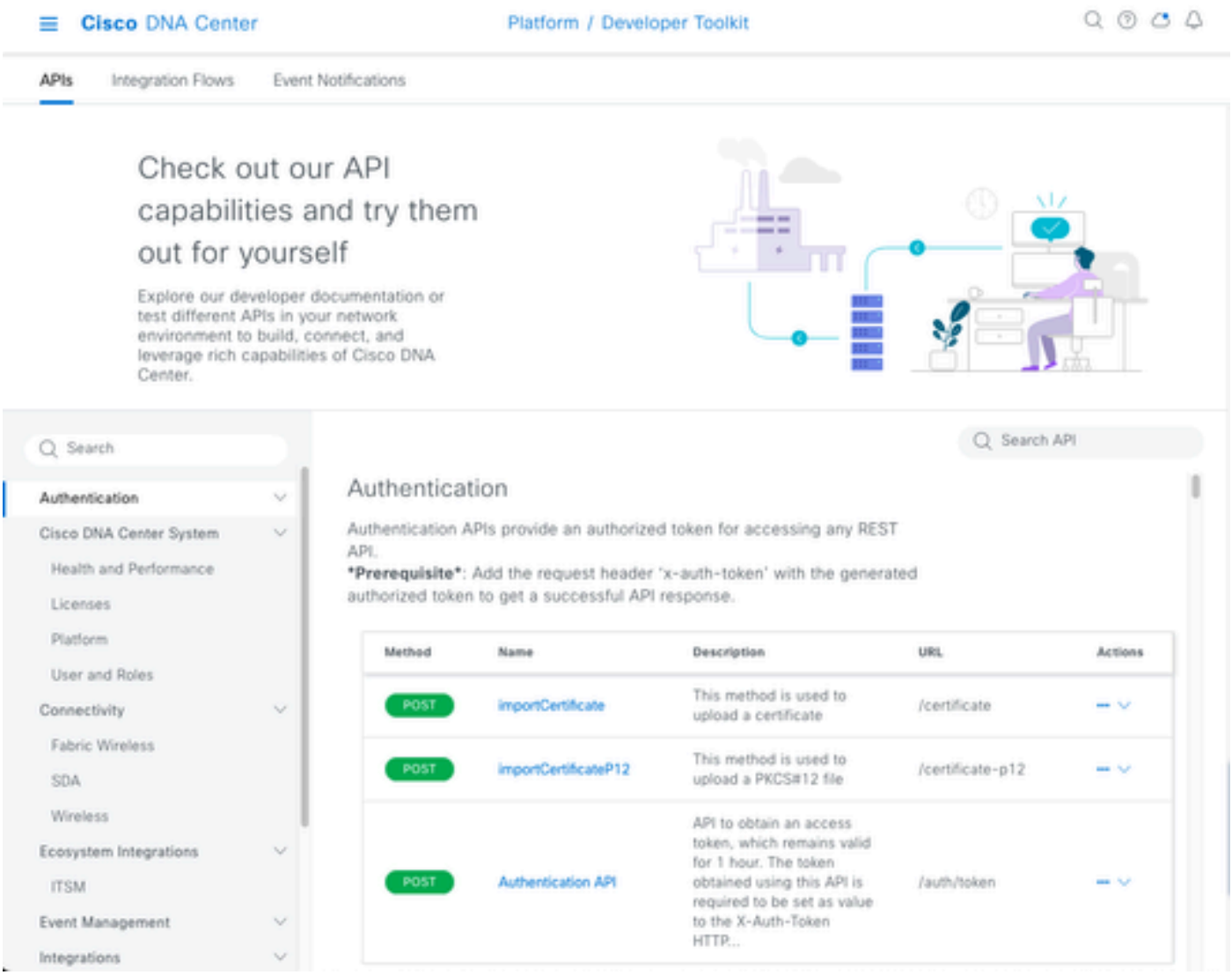


Opmerking: het Cisco Technical Assistance Center (TAC) biedt geen technische ondersteuning voor Python. Als u problemen met Python ondervindt, neemt u contact op met Python Support voor technische ondersteuning.

Configureren

Overzicht

Cisco Catalyst Center heeft veel API's beschikbaar. Om te verifiëren welke API's kunnen worden gebruikt, navigeer je op Catalyst Center naar Platform > Developer Toolkit > API's.

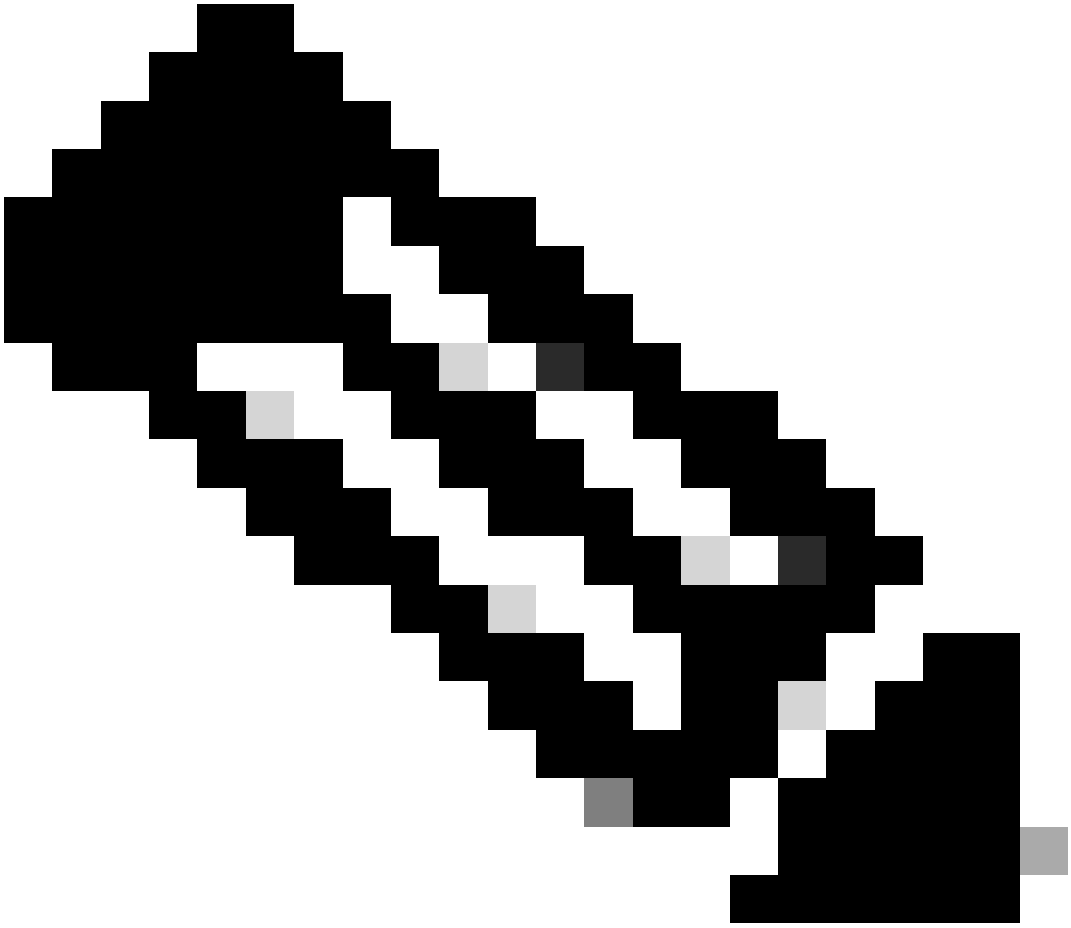


Catalyst Center API's pagina

Elke API heeft een eigen doel, afhankelijk van de informatie of de actie die moet worden uitgevoerd op Catalyst Center. Als voorwaarde voor de API's om te werken, moet een token worden gebruikt om op de juiste manier naar Catalyst Center te authenticeren en een succesvolle API-respons te krijgen. De token geeft de rechten van de REST-beller aan.

Het is ook van belang de volgende componenten van een API te identificeren:

- URL: eindpunt dat toegang biedt tot een specifieke bron.
- Methode: Alle API's moeten een methode bevatten. Het definieert de actie/handeling die de client zou willen uitvoeren naar het specifieke eindpunt. Voorbeelden: POST, GET, PUT, DELETE.
- Kop: Hier vindt u aanvullende informatie over het verzoek in de indeling voor sleutelwaardeparen. De header van de autorisatie biedt bijvoorbeeld een verificatiemethode met behulp van referenties.
- Parameters: Variabelen die specifieke instructies aan het eindpunt geven door de API te gebruiken. De parameters kunnen deel uitmaken van de URL van het eindpunt.
- payload: gegevens die tijdens de API-aanroep naar het eindpunt moeten worden verzonden.

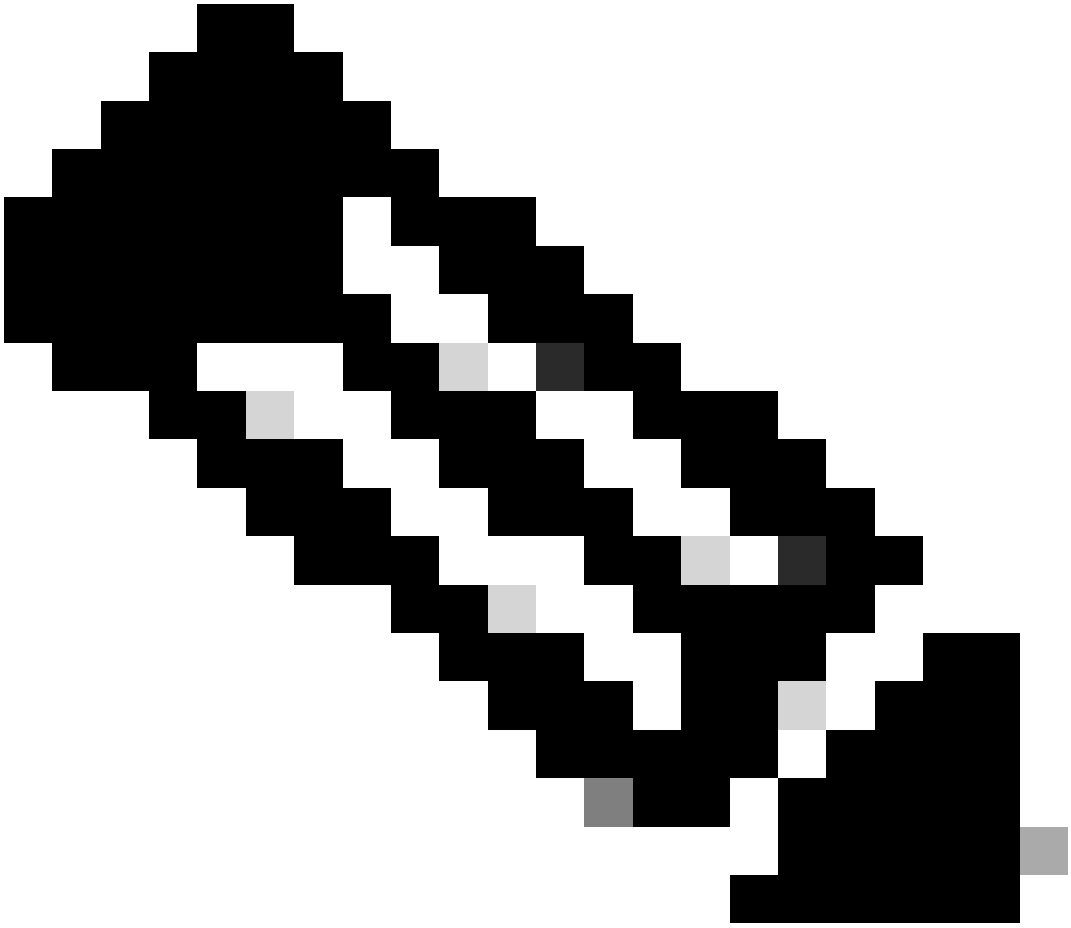


Opmerking: voor meer informatie over elke API die beschikbaar is op Catalyst Center, raadpleegt u de API Reference guide.

Modules

Gebruikte Python-modules:

- Verzoeken:Deze module maakt het mogelijk HTTP/1.1-verzoeken naar specifieke URL's te verzenden. Voor meer informatie over de module, raadpleeg de handleiding van de verzoekmodule.
- base64: Biedt coderings- en decoderingsfuncties. Voor meer informatie over de module, raadpleeg de base64 modulehandleiding.
- json: Deze module maakt het mogelijk om specifieke gegevens van APIs respons te krijgen. Voor meer informatie over de module, raadpleeg de json modulegids.



Opmerking: voor meer informatie over het installeren van Python-modules raadpleegt u documentatie over Python-modules installeren.

Generate Token

De API die verificatie-API wordt genoemd, moet worden gebruikt om een nieuw token te genereren.

Verificatie-API:

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

Het is belangrijk om te vermelden dat de token die wordt gegenereerd 1 uur geldig is. Na 1 uur moet een nieuwe token worden gegenereerd met behulp van de hierboven genoemde API.

Importeer in een nieuw Python-bestand de modules (verzoeken, base64 en json), gevolgd door vier variabelen:

```
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"
```

Verificatie-API ondersteunt Basis-autorisatie als autorisatietoken in de header. Basic Auth is een methode die kan worden gebruikt voor het verifiëren van een eindpunt, waarbij een gebruikersnaam en wachtwoord worden opgegeven, gescheiden door een dubbele punt (gebruikersnaam:wachtwoord). Beide waarden zijn base64-encoded, en het eindpunt decodeert de login referenties en controle, als de gebruiker kan toegang, of niet.

Om de Base64-gecondenseerde string voor onze gebruikersnaam en wachtwoord te maken, wordt de base64 module gebruikt. Om dit te bereiken wordt de b64encode functie gebruikt.

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

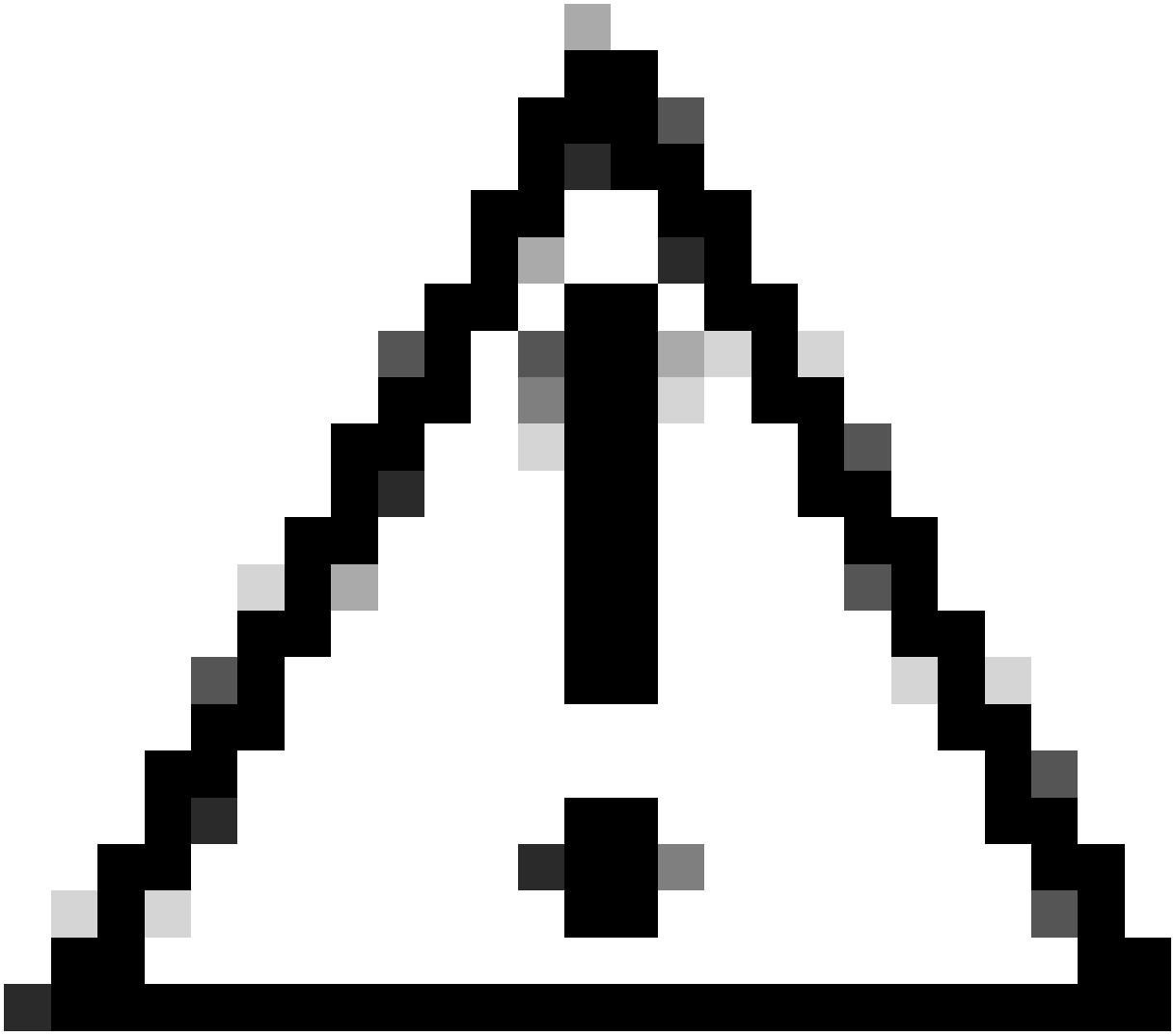
Uit de bovenstaande code is een byte_string variabele gemaakt met behulp van de '.encode("ascii")' functie. Dit komt doordat de functie base64.b64encode een byteachtig object vereist. Merk ook op dat gebruiker en wachtwoord variabelen werden gebruikt om de string formaat 'user:password' te behouden. Uiteindelijk is er een base64-encoded byte string gemaakt met de gebruiker en het wachtwoord. Met behulp van de methode 'decode()' werd de waarde omgezet in str object.

Om het te verifiëren, kunt u de waarde voor de variabele authorisationBase64 afdrukken:

```
print(authorizationBase64)
```

Voorbeeld uitvoer:

```
am9yZ2QhbDI6Sm9yZ2VhbDXxXxXx
```



Let op: base64 is geen encryptie-algoritme. Het mag niet voor veiligheidsdoeleinden worden gebruikt. De verificatie API ondersteunt ook AES-sleutelencryptie als autorisatie-token in de header die meer beveiliging biedt.

Nu een base64-encoded string is gemaakt met behulp van de gebruiker en het wachtwoord om te authenticeren naar, Catalyst Center, is het tijd om verder te gaan met de API-verificatie API aanroep met behulp van de module aanvragen. De functie genaamd request staat ook toe om een response object te krijgen dat de tekst van het request bevat.

Syntaxis van de methode:

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

**kwargs betekent elke parameter die in het verzoek wordt doorgegeven, bijvoorbeeld cookies, user-agents, payload, headers, etc.

De API voor verificatie specificeert dat de methode POST is, de URL "/dna/system/api/v1/auth/token" is en dat de basisauth in de header moet worden gespecificeerd.

Deze variabelen worden aangemaakt om ze te gebruiken voor de request() functie.

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

Voor de variabele headers zijn twee dingen gespecificeerd. De eerste is het content-type, dat het mediatype aangeeft van de resource die naar het eindpunt wordt verzonden (dit helpt bij het nauwkeurige parseren en verwerken van de gegevens). De tweede is Autorisatie, die, in dit geval, de variabele authorisationBase64 (die onze base64-gecondenseerde string opslaat) wordt verzonden als parameter om te authenticeren naar Catalyst Center.

Ga nu verder met het gebruik van de request() functie om de API aanroep uit te voeren. De volgende code toont de syntaxis van de functie:

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

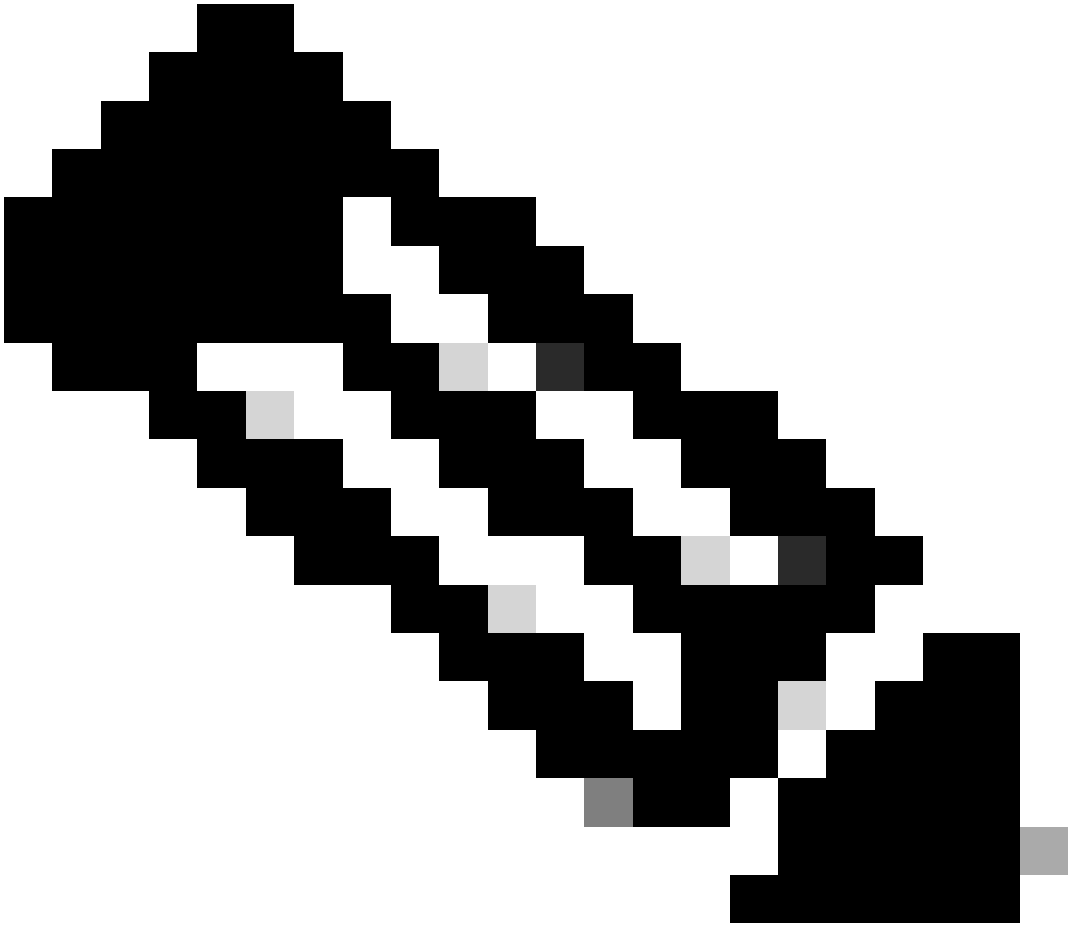
De responsvariabele is gemaakt om de gegevens van onze API-oproep op te slaan.

Om de verkregen respons af te drukken, gebruikt u de afdrukfunctie in combinatie met de methode text() in de responsvariabele. De methode text() genereert een str-object met de respons van Catalyst Center.

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

Voorbeeld uitvoer:

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



Opmerking: als Catalyst Center een zelfondertekend certificaat gebruikt, kan de API-aanvraag mislukken bij de volgende fout:

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

Om dit probleem op te lossen moet u de verify-parameter als fout toevoegen aan de aanvraagfunctie. Dit gaat voorbij aan het verifiëren van het SSL-certificaat vanaf het eindpunt (Catalyst Center).

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

Van de reactie die van de API authenticatie oproep wordt ontvangen, merk op dat de structuur

vergelijkbaar is met een woordenboek in Python, maar het is een str object.

Om het type van een object te valideren, gebruikt u de functie type().

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

Wat de volgende output retourneert:

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

Voor praktische doeleinden hoeft alleen de token-waarde te worden afgeleid uit de respons die van de API wordt ontvangen, niet uit de hele string, omdat, om de andere Catalyst Center API's te kunnen gebruiken, alleen de token als parameter moeten worden meegegeven.

Aangezien de reactie van de API-oproep een structuur heeft die lijkt op een woordenboek in Python, maar het objecttype is str, moet het genoemde object worden omgezet in een woordenboek met behulp van de json module. Dit haalt de token waarde uit de hele string die ontvangen wordt van de API.

Om dit te bereiken, converteert de functie json.loads() de string in een woordenboek om later alleen de token-waarde te extraheren en deze direct toe te wijzen aan onze token-variabele.

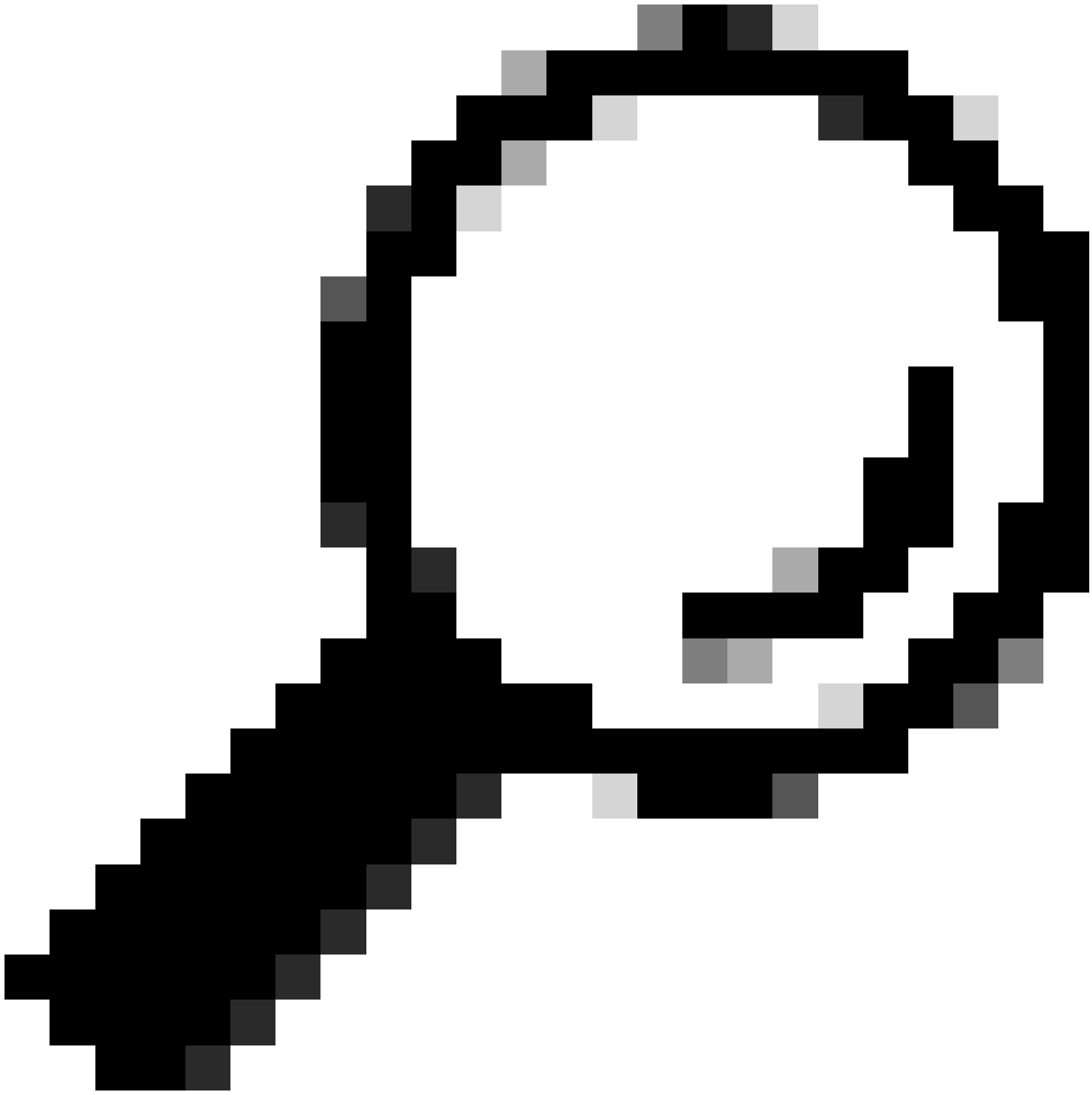
```
token = json.loads(response.text)  # Converting the response.text string value into a dictionary (It is
token = (token["Token"])  # Extracting just the token value by specifying the key as a parameter.
```

Om te verifiëren dat de token variabele alleen de token als waarde heeft, gaat u verder met afdrukken.

```
print(token)
```

Voorbeeld uitvoer:

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



Tip: Aangezien elk gegenereerde token standaard binnen 1 uur verloopt, kan een Python-methode die de code bevat om een token te genereren, worden aangemaakt en aangeroepen telkens als een token vervalt, zonder dat het gehele programma hoeft te worden uitgevoerd door de aangelegde methode aan te roepen.

Een API testen

Nu het token met succes is toegewezen aan de token-variabele, kunnen Catalyst Center API's worden gebruikt die beschikbaar zijn.

In dit geval wordt de samenvattende API voor configuratie van Cisco DNA Center-knooppunten getest.

Configuratieoverzicht van Cisco DNA Center-knooppunten

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

Deze API biedt informatie over de huidige configuratie van Catalyst Center zoals, NTP-server geconfigureerd, nodenaam, intra-cluster link, LACP-modus, enzovoort.

De API voor configuratieoverzichten van Cisco DNA Center Nodes specificeert in dit geval dat de gebruikte methode GET is, de URL is "/dna/intent/api/v1/nodes-config" en omdat de token string is geëxtraheerd en toegewezen aan de token-variabele, wordt deze keer de token doorgegeven als een variabele in de header van de API-aanroep als ‘X-Auth-Token’: gevolgd door de token.

Dit verifieert het verzoek aan Catalyst Center voor elke API-oproep die wordt uitgevoerd. Onthoud dat elke token 1 uur duurde. Na 1 uur moet een nieuwe token worden gegenereerd om API-oproepen naar Catalyst Center te kunnen blijven maken.

Maak de variabelen om de API te testen:

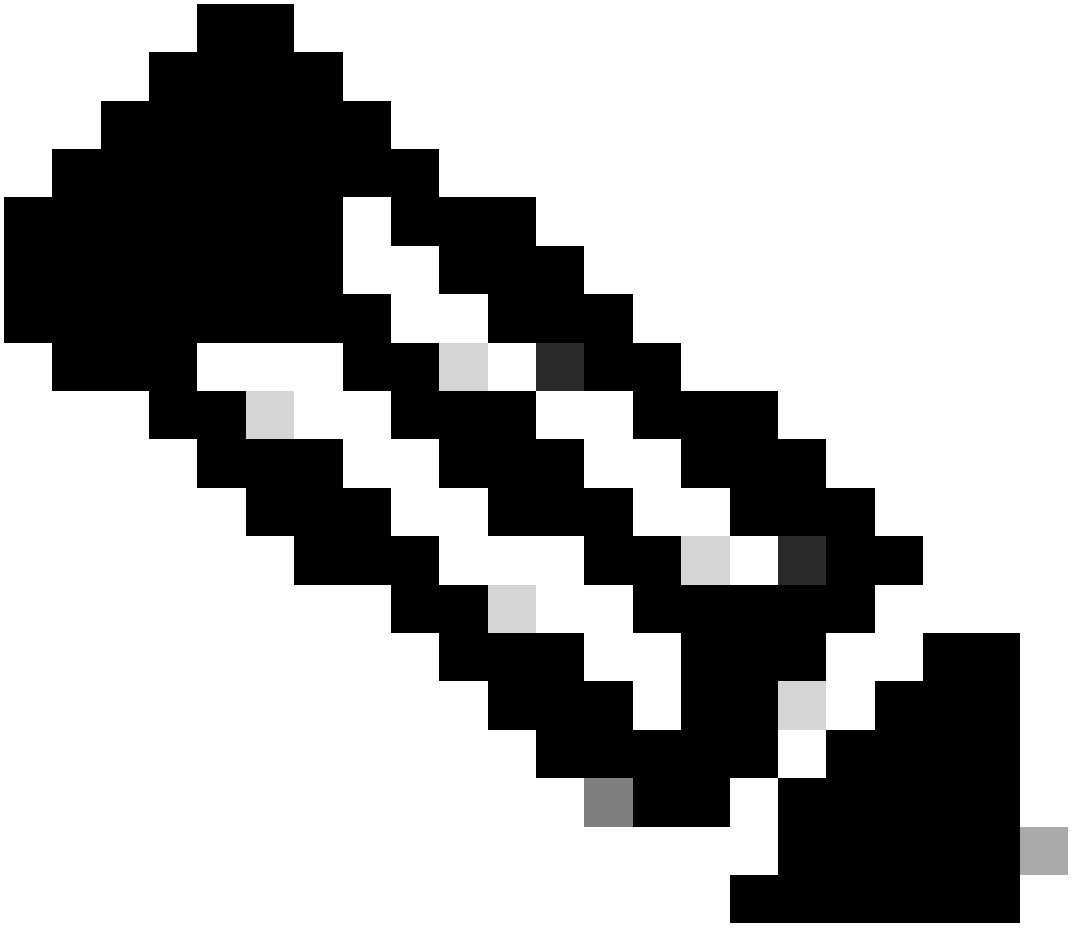
```
nodeInfo_url = "https://<CatalystCenterIP>/dna/intent/api/v1/nodes-config"
nodeInfo_headers = {
    'X-Auth-Token': token
}

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

nodeInfo_url variabele is gemaakt om de URL van onze API op te slaan. nodeInfo_headers variabele slaat de headers voor onze API op. In dit geval werden ‘X-Auth-Token:’ en de token-variabele als parameters doorgegeven om het verzoek met succes te authenticeren op Catalyst Center. Tot slot slaat de variabele nodeInfoResponse de respons van de API op.

Om de ontvangen reactie te valideren, kan je de print() functie gebruiken.

Voorbeeld uitvoer:

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



Opmerking: als er een zelfondertekend certificaat wordt gebruikt in Catalyst Center, kan de API-aanvraag mislukken bij de volgende fout:

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

Om dit probleem op te lossen moet u de verify-parameter als fout toevoegen aan het verzoek. Dit onderdrukt de verificatie van SSL-certificaat vanaf het eindpunt (Catalyst Center).

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

De antwoorden die van de API worden ontvangen kunnen moeilijk leesbaar zijn. Met behulp van

de json() module kan de response worden afgedrukt in een beter leesbare string. Eerst moet de API-respons in een JSON-object worden geladen met behulp van de functie json.loads() gevolgd door de functie 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: Deze functie retourneert het JSON object genomen als een parameter in een JSON opgemaakte string.

paragraaf: Deze parameter definieert het inspringing niveau voor de JSON opgemaakte string.

Voorbeeld uitvoer:

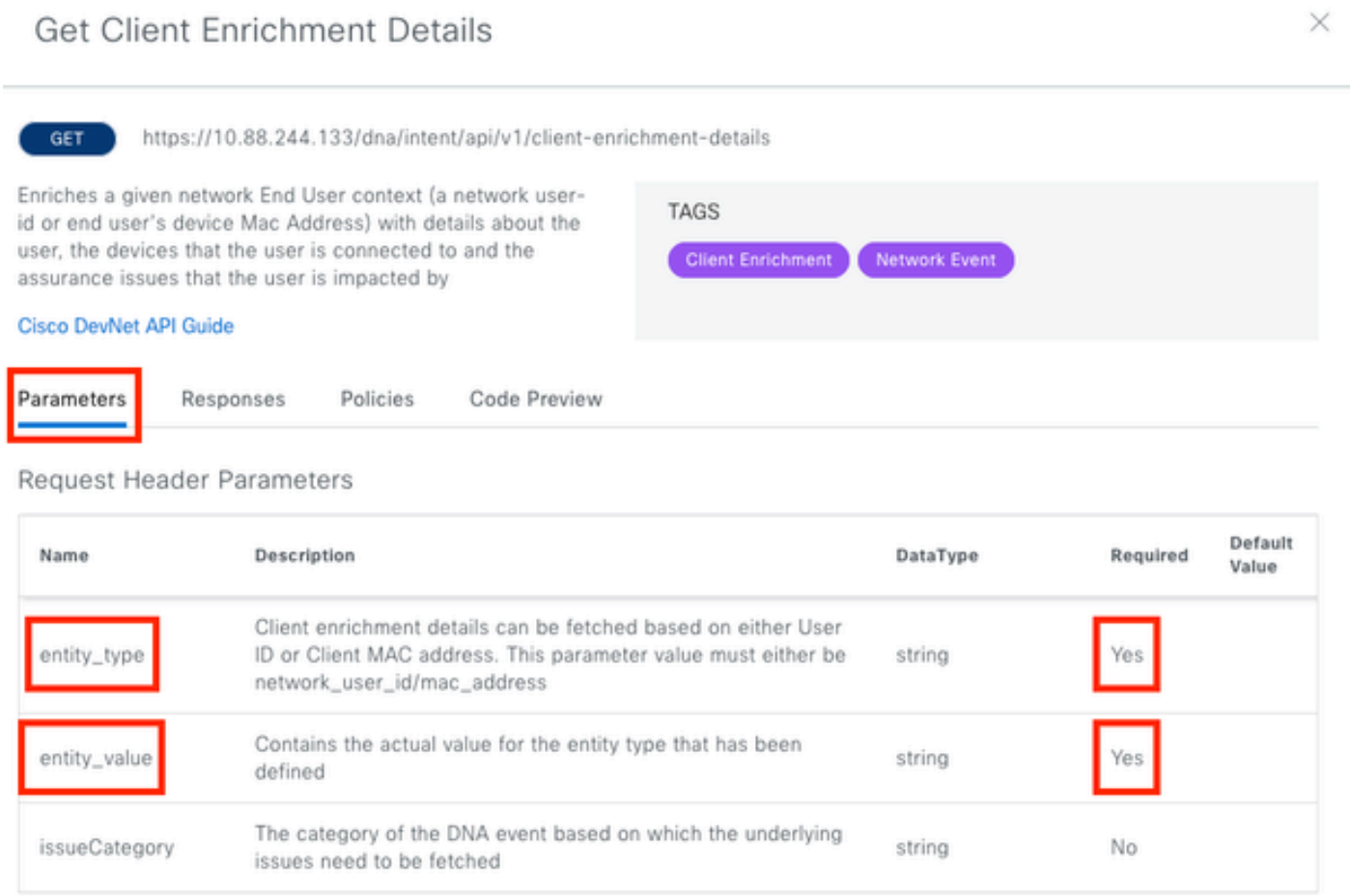
```
{
  "response": {
    "nodes": [
      {
        "name": "X.X.X.X",
        "id": "ea5dbec1-fbb6-4339-9242-7694eb1xXxX",
        "network": [
          {
            "slave": [
              "enp9s0"
            ],
            "lacp_supported": true,
            "intra_cluster_link": false,
!---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

API’s met headerparameters

Er zijn enkele API's die vereisen dat sommige parameters worden verzonden in de Kop om te werken zoals verwacht. In dit geval wordt de API Get Client Enrichment Details getest.

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

Om te verifiëren welke Headers Parameters vereist zijn om de API te laten werken zoals verwacht, navigeer naar Platform > Developer Toolkit > API's > Get Client Enrichment Details en klik op de naam van de API. Er wordt een nieuw venster geopend en onder de optie Parameters worden Headers Parameters (Koppen) weergegeven die nodig zijn om de API te laten werken.



In dit geval kan voor de entity_type parameter, volgens de beschrijving, de waarde netwerk_user_id of mac_address zijn en de entity_value parameter moet de waarde bevatten voor het entiteit type dat is gedefinieerd.

Om ermee door te gaan, worden twee nieuwe variabelen gedefinieerd, entity_type en entity_value met hun corresponderende waarden:

```
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
```

Er worden ook nieuwe variabelen gemaakt om de API-aanroep uit te voeren. De URL van de API aanroep wordt opgeslagen in de userEnrichment_url variabele. Koppen worden opgeslagen in de variabele userEnrichmentHeaders. De ontvangen reactie wordt opgeslagen in userEnrichmentResponse variabele.

```
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)
```

Zoals u kunt zien, zijn van de userEnrichmentHeaders, entity_type en entity_value variabelen doorgegeven als Kop parameters voor de API aanroep, samen met de token variabele.

Om de ontvangen reactie te valideren gebruikt u de functie print().

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

Voorbeeld uitvoer:

```
[ {
  "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’s met queryparameters

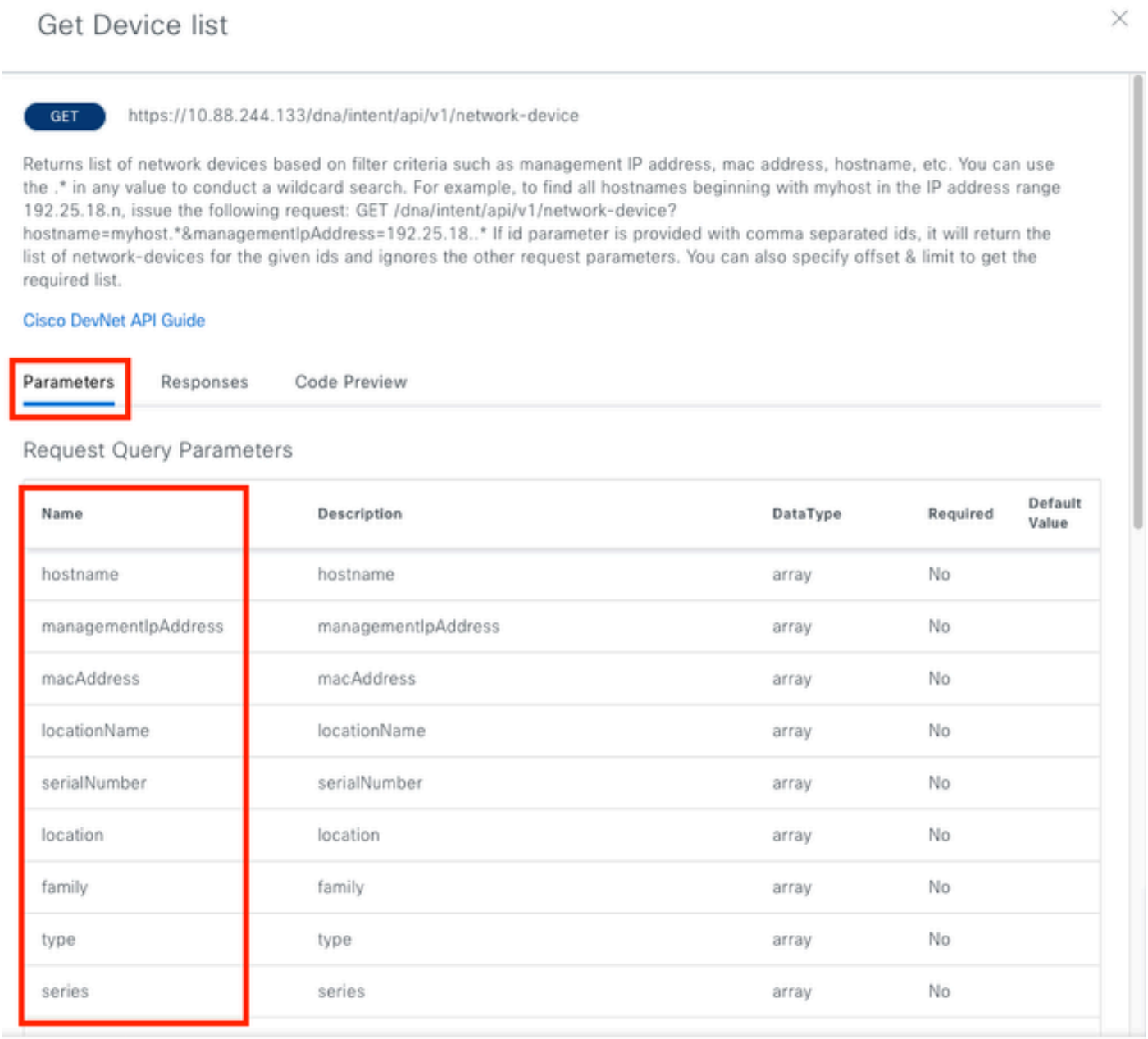
Query-parameters kunnen worden gebruikt om een specifiek aantal resultaten te filteren dat door een API wordt geretourneerd. Deze parameters worden toegevoegd in de URL van de API.

De API-aanroep Get Device List wordt getest.

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

De API Get Device List retourneert een lijst met alle apparaten die zijn toegevoegd in Catalyst Center. Als de details voor een specifiek apparaat worden gevraagd, kunnen de vraagparameters helpen om specifieke informatie te filteren.

Om te verifiëren welke vraagparameters voor API beschikbaar zijn, navigeer aan Platform > Ontwikkelaar Toolkit > APIs > krijg de Lijst van het Apparaat en klik op de naam van API. Er wordt een nieuw venster geopend en onder de optie Parameters worden de zoekparameters die voor de API beschikbaar zijn weergegeven.



In dit voorbeeld worden managementIpAddress en serienummer query parameters gebruikt (houd er rekening mee dat het niet nodig is om alle query parameters voor de API aanroep te gebruiken). Ga verder met het maken en toewijzen van de corresponderende waarden voor beide queryparameters.

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

Zoals hierboven vermeld, worden de zoekparameters toegevoegd in de URL van de API, met name aan het eind van de API, met behulp van een "?" gevolgd door de zoekparameters.

Als er meerdere query parameters worden gebruikt, wordt er een &'-teken tussen deze parameters geplaatst om een query string te vormen.

Het volgende voorbeeld toont hoe de vraagparameters aan de variabele deviceListUrl toe te voegen die URL van de API vraag opslaat.

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

Bericht dat de eerder gemaakte variabelen aan de URL-string zijn toegevoegd. Met andere woorden, de hele string van de URL ziet er als volgt uit:

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

Doorgaan met de API-aanroep, de variabele deviceListHeaders is gemaakt om de API-koppen op te slaan samen met de als parameter doorgegeven token-variabele en de variabele deviceListResponse slaat de API-respons op.

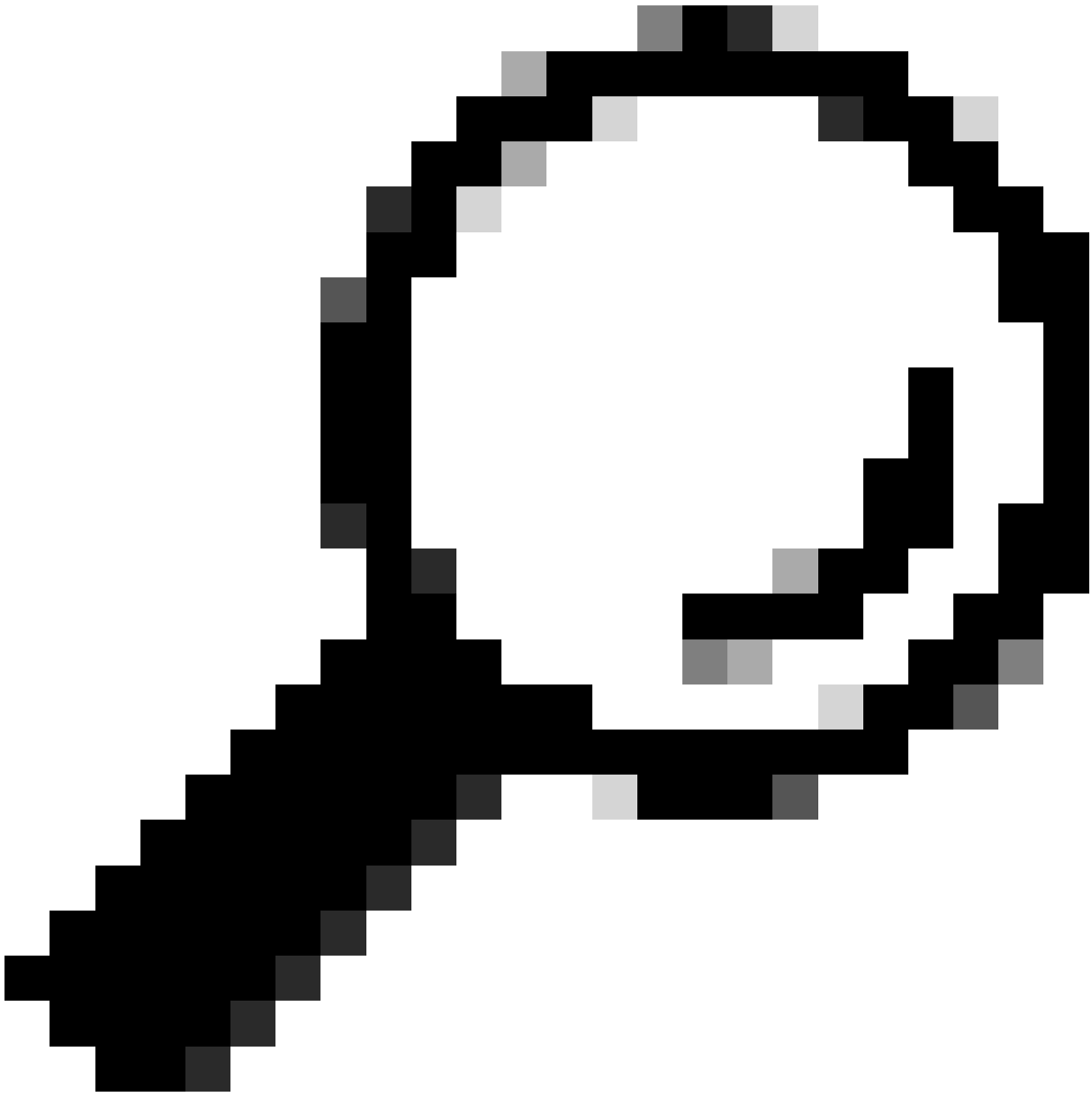
```
deviceListHeaders = {
            'X-Auth-Token': token,
        }

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

Om de ontvangen reactie te valideren, kan je de print() functie gebruiken.

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

Voorbeeld uitvoer:

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



Tip: om de respons op een meer leesbare manier af te drukken, kunt u json.loads() en json.dumps() functies gebruiken die worden beschreven in het gedeelte Test API.

Over deze vertaling

Cisco heeft dit document vertaald via een combinatie van machine- en menselijke technologie om onze gebruikers wereldwijd ondersteuningscontent te bieden in hun eigen taal. Houd er rekening mee dat zelfs de beste machinevertaling niet net zo nauwkeurig is als die van een professionele vertaler. Cisco Systems, Inc. is niet aansprakelijk voor de nauwkeurigheid van deze vertalingen en raadt aan altijd het oorspronkelijke Engelstalige document (link) te raadplegen.