# Catalyst Center-APIs mit Python

## Inhalt



## Einleitung

In diesem Dokument wird die Verwendung der verschiedenen APIs beschrieben, die in Cisco Catalyst Center mit Python zur Verfügung stehen.

## Voraussetzungen

### Anforderungen

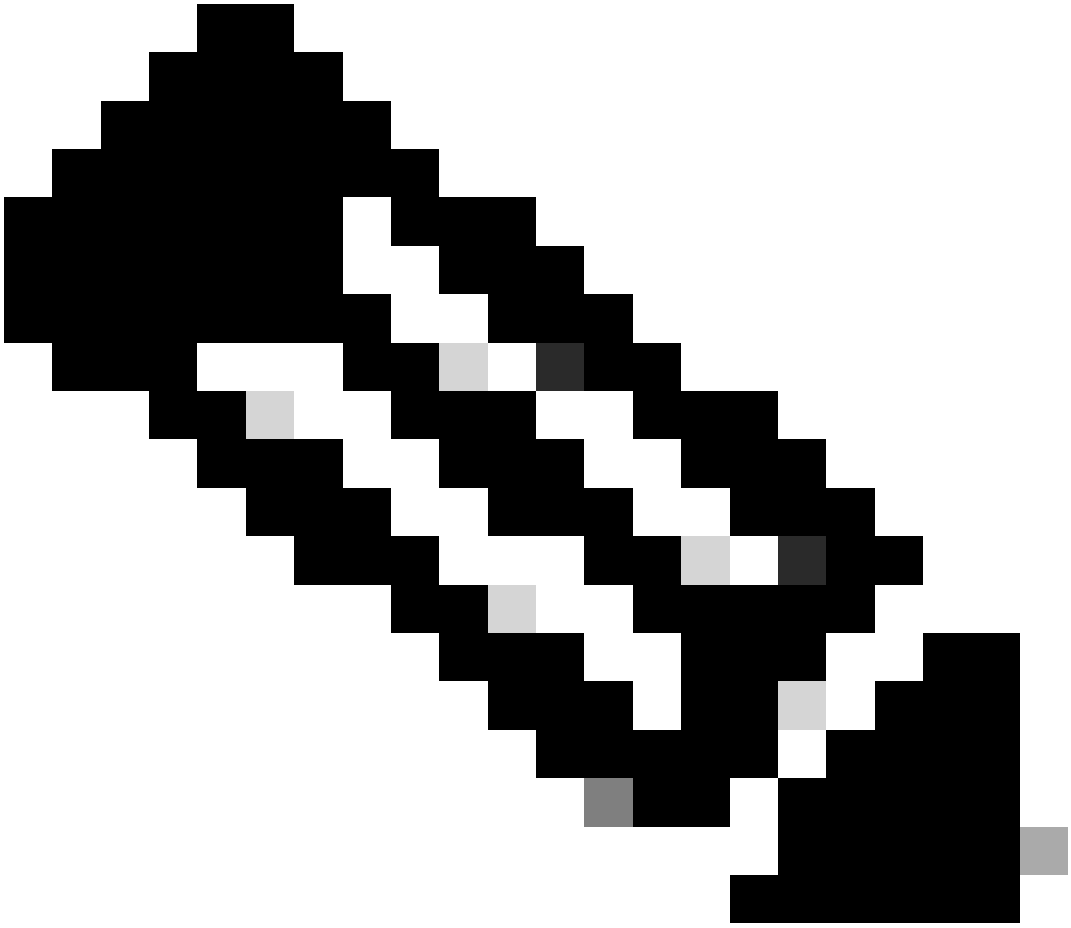
Grundlegendes Wissen zu:

- Cisco Catalyst Center
- APIs
- Python

### Verwendete Komponenten

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

Die Informationen in diesem Dokument beziehen sich auf Geräte in einer speziell eingerichteten Testumgebung. Alle Geräte, die in diesem Dokument benutzt wurden, begannen mit einer gelöschten (Nichterfüllungs) Konfiguration. Wenn Ihr Netzwerk in Betrieb ist, stellen Sie sicher, dass Sie die möglichen Auswirkungen aller Befehle kennen.

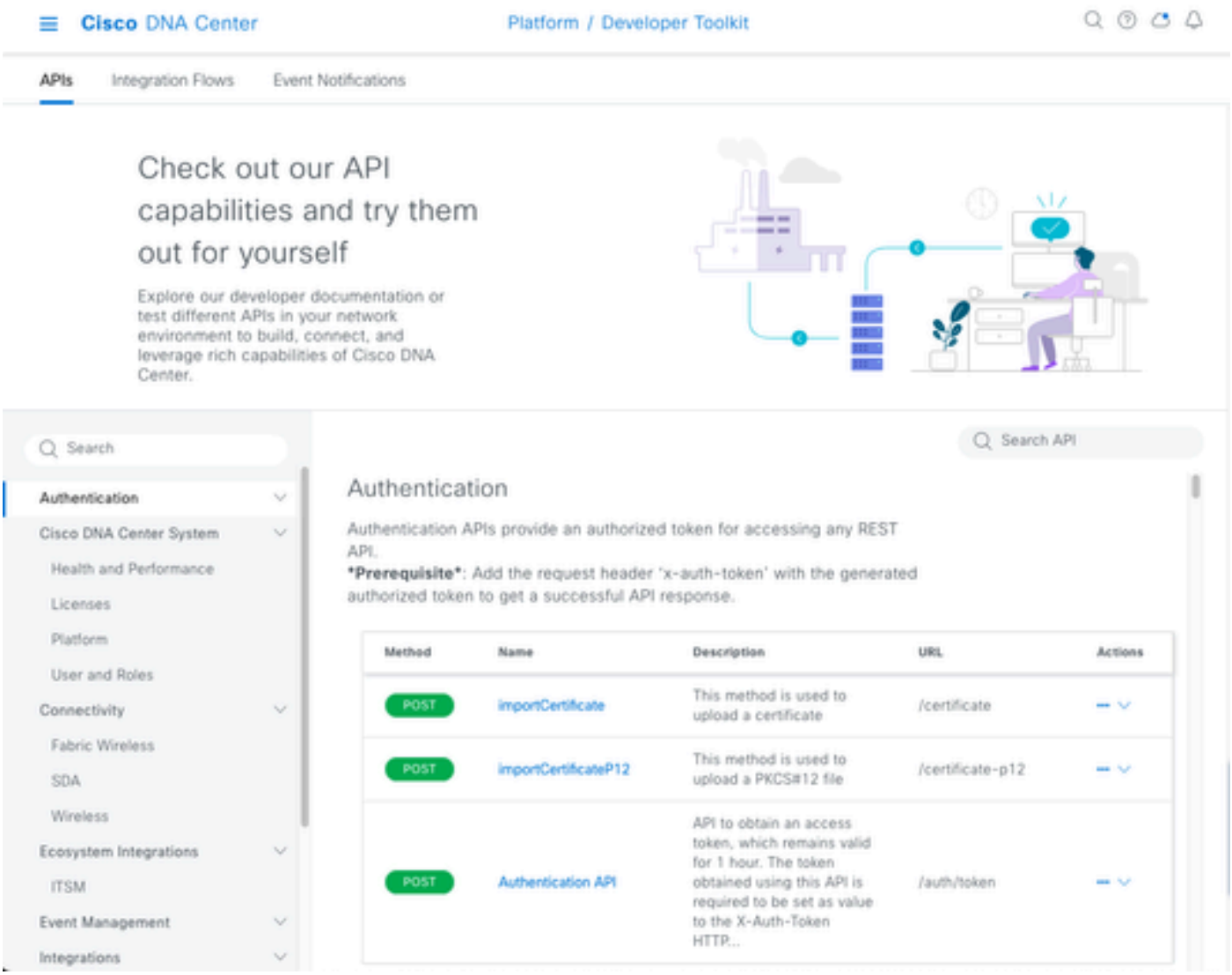


Hinweis: Das Cisco Technical Assistance Center (TAC) bietet keinen technischen Support für Python. Bei Problemen mit Python wenden Sie sich bitte an den Python-Support.

# Konfigurieren

## Überblick

Cisco Catalyst Center bietet eine Vielzahl von APIs. Um zu überprüfen, welche APIs verwendet werden können, navigieren Sie in Catalyst Center zu Platform > Developer Toolkit > APIs.

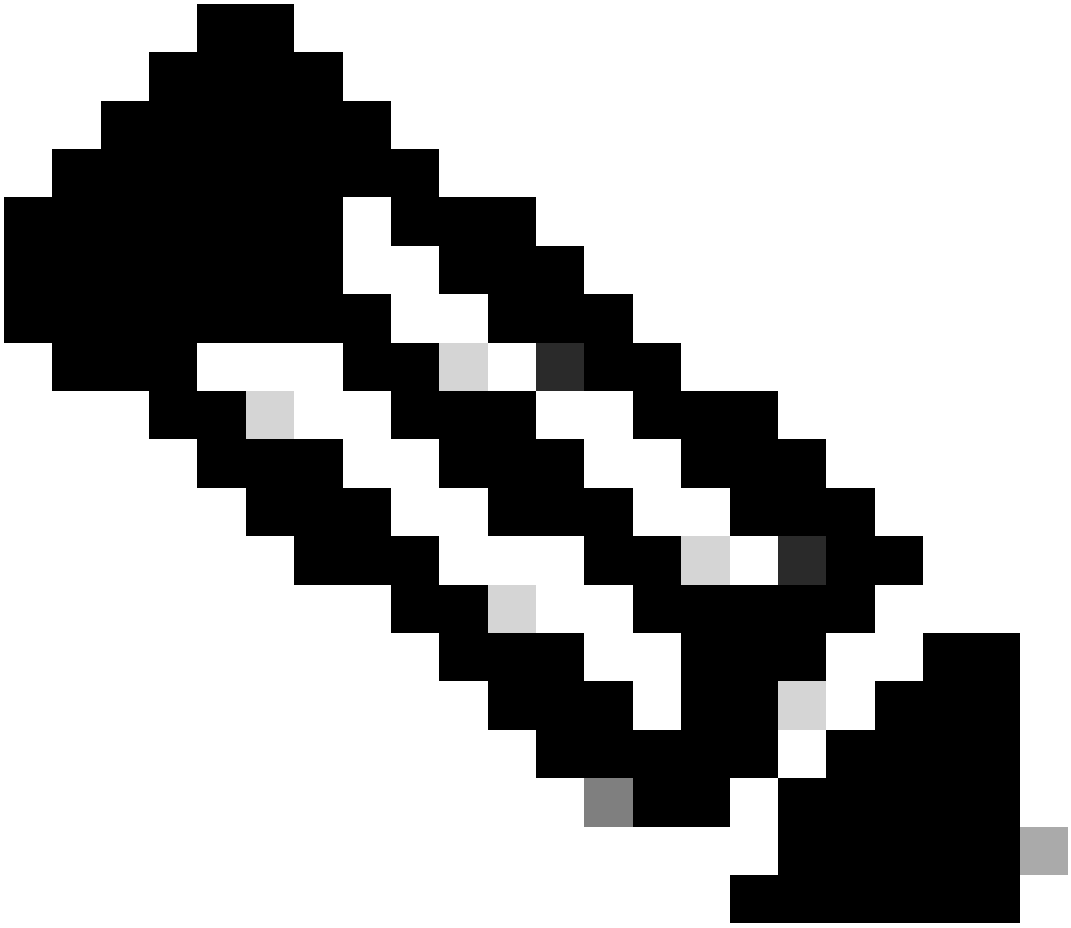



Catalyst Center APIs-Seite

Jede API hat ihren eigenen Zweck, je nach den Informationen oder der erforderlichen Maßnahme, die in Catalyst Center durchgeführt werden muss. Damit die APIs funktionieren, muss ein Token verwendet werden, um sich bei Catalyst Center ordnungsgemäß zu authentifizieren und eine erfolgreiche API-Antwort zu erhalten. Das Token identifiziert die Berechtigungen für den REST-Anrufer entsprechend.

Es ist auch wichtig, die folgenden Komponenten zu identifizieren, die eine API bilden:

- URL: Endpunkt, der Zugriff auf eine bestimmte Ressource bereitstellt.
- Methode: Alle APIs müssen eine Methode enthalten. Sie definiert die Aktion bzw. den Vorgang, die bzw. der der Client für den jeweiligen Endpunkt ausführen möchte. Beispiele: POST, GET, PUT, DELETE.
- Header: Stellt zusätzliche Informationen zur Anforderung im Format der Schlüssel-Wert-Paare bereit. Der Autorisierungsheader stellt beispielsweise eine Authentifizierungsmethode mit Anmeldeinformationen bereit.
- Parameter: Variablen, die spezifische Anweisungen für den Endpunkt mithilfe der API bereitstellen. Die Parameter können Teil der URL des Endpunkts sein.
- Payload: Daten, die während des API-Aufrufs an den Endpunkt gesendet werden müssen.

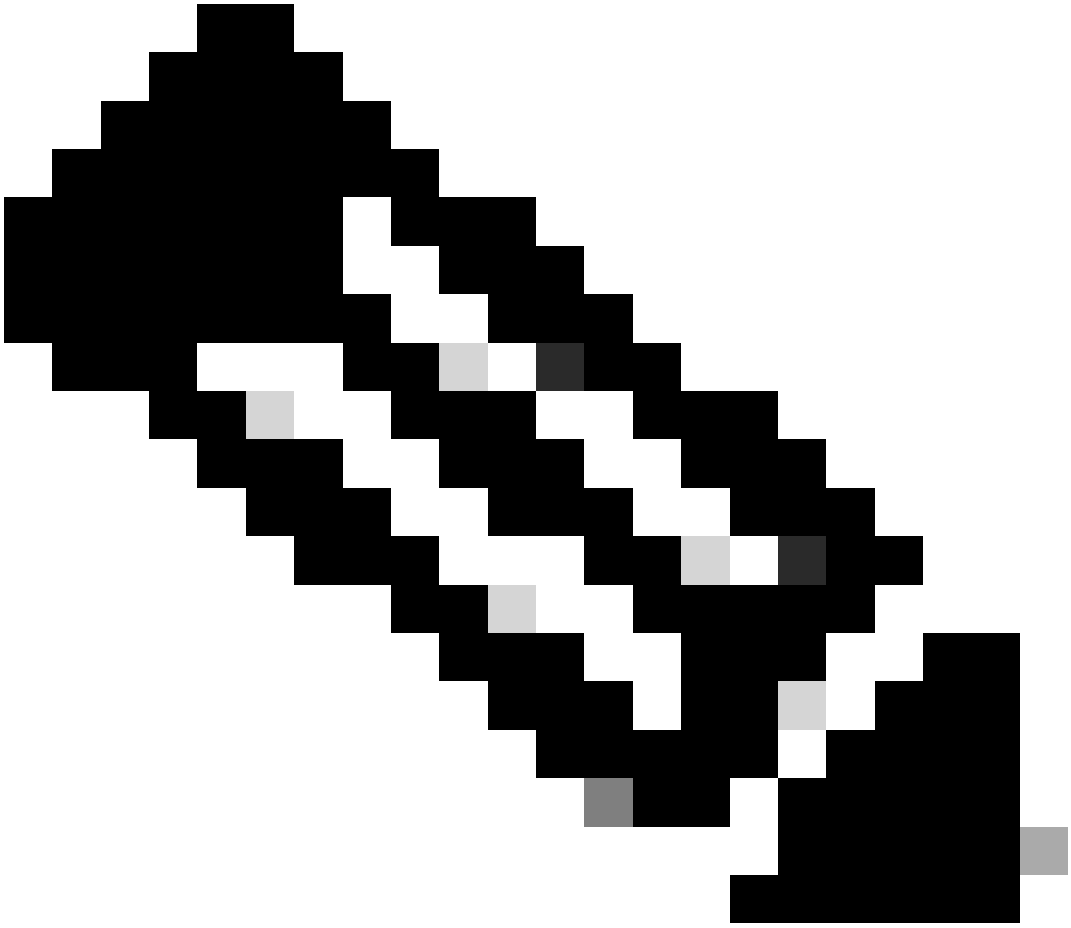


Hinweis: Weitere Informationen zu den einzelnen APIs von Catalyst Center finden Sie im API Reference Guide.

## Module

Verwendete Python-Module:

- Anfragen:Dieses Modul ermöglicht das Senden von HTTP/1.1-Anfragen an bestimmte URLs. Weitere Informationen zum Modul finden Sie im Anforderungsmodul-Handbuch.
- base64: Stellt Kodierungs- und Dekodierungsfunktionen bereit. Weitere Informationen zum Modul finden Sie in der base64-Modulanleitung.
- json: Dieses Modul ermöglicht das Abrufen spezifischer Daten aus der API-Antwort. Weitere Informationen zum Modul finden Sie in der json-Modulanleitung.



Hinweis: Weitere Informationen zur Installation von Python-Modulen finden Sie in der Dokumentation zur Installation von Python-Modulen.

## Token generieren

Die als Authentifizierungs-API bezeichnete API muss zum Generieren eines neuen Tokens verwendet werden.

Authentifizierung-API:

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

Es ist wichtig zu erwähnen, dass das generierte Token 1 Stunde gültig ist. Nach 1 Stunde muss ein neues Token mit der oben genannten API generiert werden.

Importieren Sie in einer neuen Python-Datei die Module (Requests, base64 und json), und erstellen Sie anschließend vier Variablen:

```
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"
```

Die Authentifizierungs-API unterstützt die Standardauthentifizierung als Autorisierungstoken im Header. Die Standardauthentifizierung ist eine Methode, mit der sich Endpunkte authentifizieren können. Dabei werden Benutzername und Kennwort durch einen Doppelpunkt (username:password) getrennt. Beide Werte sind Base64-codiert, und der Endpunkt decodiert die Anmeldedaten und überprüft, ob der Benutzer darauf zugreifen kann.

Zur Erstellung der Base64-konformen Zeichenfolge für unseren Benutzernamen und unser Kennwort wird das base64-Modul verwendet. Dazu wird die b64encode-Funktion verwendet.

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

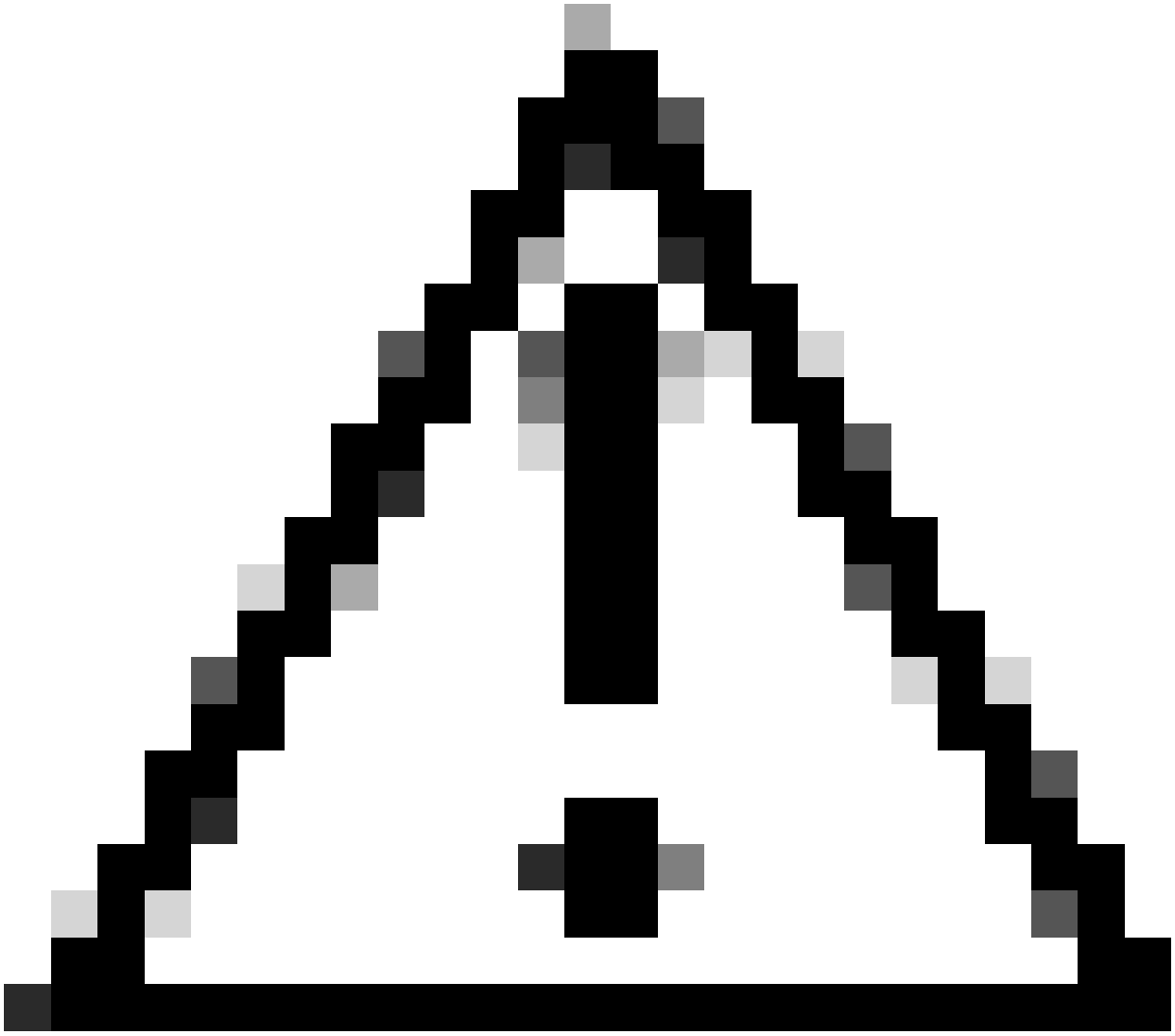
Aus dem obigen Code wurde eine byte_string-Variable mit der Funktion ‘encode("ascii") erstellt. Dies liegt daran, dass die base64.b64encode-Funktion ein byteähnliches Objekt erfordert. Beachten Sie außerdem, dass Variablen für Benutzer und Kennwort verwendet wurden, um das Zeichenfolgenformat "user:password" beizubehalten. Schließlich wurde eine Base64-kodierte Byte-Zeichenfolge mit dem Benutzer und dem Kennwort erstellt. Mit der decode()-Methode wurde der Wert in ein str-Objekt konvertiert.

Um dies zu überprüfen, können Sie den Wert für die authorizationBase64-Variable ausdrucken:

```
print(authorizationBase64)
```

Ausgabebeispiel:

```
am9yZ2QhbDI6Sm9yZ2VhbDXxXxXx
```



Achtung: base64 ist kein Verschlüsselungsalgorithmus. Es darf nicht zu Sicherheitszwecken verwendet werden. Die Authentifizierungs-API unterstützt auch die AES-Schlüsselverschlüsselung als Autorisierungs-Token im Header, was mehr Sicherheit bietet.

---

Nachdem eine mit Base64 verschlüsselte Zeichenfolge mit dem Benutzer und dem Kennwort für die Authentifizierung bei Catalyst Center erstellt wurde, ist es an der Zeit, den API-Authentifizierungs-API-Aufruf mithilfe der Modulanforderungen fortzusetzen. Außerdem ermöglicht die Funktion request, ein Antwortobjekt zu erhalten, das den Text der Anforderung enthält.

Syntaxis der Methode:

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

**kwargs bezeichnet jeden Parameter, der in die Anfrage übergeben wird, z. B. Cookies, User-

Agents, Payload, Header usw.

Die Authentifizierungs-API gibt an, dass die Methode POST ist, die URL "/dna/system/api/v1/auth/token" ist und die grundlegende Authentifizierung im Header angegeben werden muss.

Diese Variablen werden erstellt, um sie für die request() -Funktion zu verwenden.

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

Für die Variable headers wurden zwei Dinge angegeben. Der erste ist der Inhaltstyp, der den Medientyp der Ressource angibt, die an den Endpunkt gesendet wird (dies hilft dem Endpunkt, die Daten genau zu analysieren und zu verarbeiten). Die zweite ist Authorization, die in diesem Fall die Variable AuthorizationBase64 (die unsere Base64-kontierte Zeichenfolge speichert) als Parameter für die Authentifizierung an Catalyst Center gesendet wird.

Fahren Sie nun mit der request()-Funktion fort, um den API-Aufruf auszuführen. Der nächste Code zeigt die Syntaxis der Funktion:

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

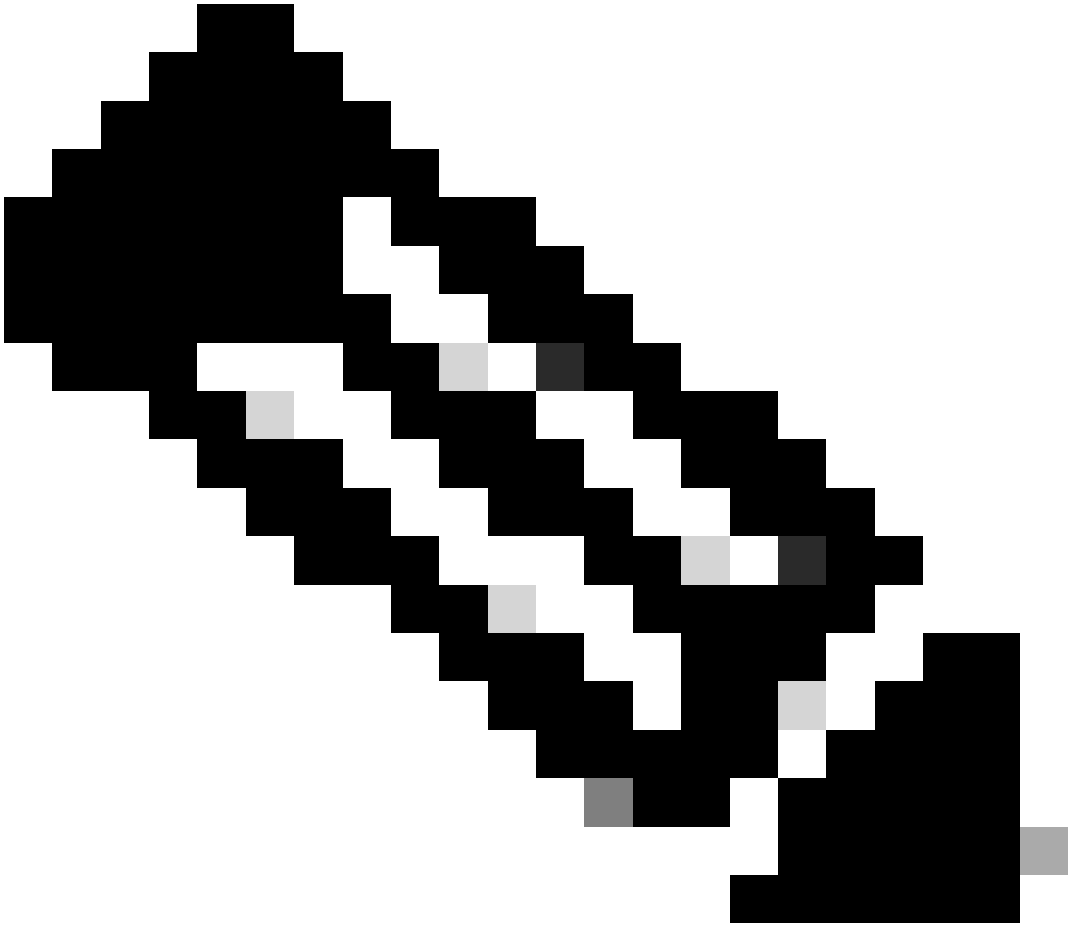
Die Antwortvariable wurde erstellt, um die Daten unseres API-Aufrufs zu speichern.

Um die erhaltene Antwort zu drucken, verwenden Sie die Druckfunktion zusammen mit der text()-Methode in der Response-Variable. Die text()-Methode generiert ein str-Objekt mit der Antwort, die vom Catalyst Center empfangen wird.

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

Ausgabebeispiel:

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



Hinweis: Wenn Catalyst Center ein selbstsigniertes Zertifikat verwendet, kann die API-Anforderung mit dem nächsten Fehler fehlschlagen:

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

Um dieses Problem zu beheben, müssen Sie der Anforderungsfunktion den Parameter verify als False hinzufügen. Dabei wird die Überprüfung des SSL-Zertifikats vom Endpunkt (Catalyst Center) ignoriert.

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

---

Aus der Antwort, die vom API-Authentifizierungsaufruf empfangen wurde, ist zu beachten, dass

die Struktur einem Dictionary in Python ähnelt, jedoch ein str-Objekt ist.

Um den Typ eines Objekts zu überprüfen, verwenden Sie die type()-Funktion.

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

Gibt die nächste Ausgabe zurück:

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

Aus praktischen Gründen muss nur der Tokenwert aus der Antwort extrahiert werden, die von der API empfangen wurde, nicht die gesamte Zeichenfolge, da für die Verwendung der anderen Catalyst Center-APIs nur das Token als Parameter übergeben werden muss.

Da die vom API-Aufruf empfangene Antwort eine Struktur ähnlich einem Dictionary in Python aufweist, der Objekttyp jedoch str ist, muss das genannte Objekt mithilfe des json-Moduls in ein Dictionary konvertiert werden. Dadurch wird der Tokenwert aus der gesamten, von der API empfangenen Zeichenfolge extrahiert.

Dazu konvertiert die Funktion json.loads() den String in ein Dictionary, um später nur den Tokenwert zu extrahieren und direkt unserer Token-Variable zuzuweisen.

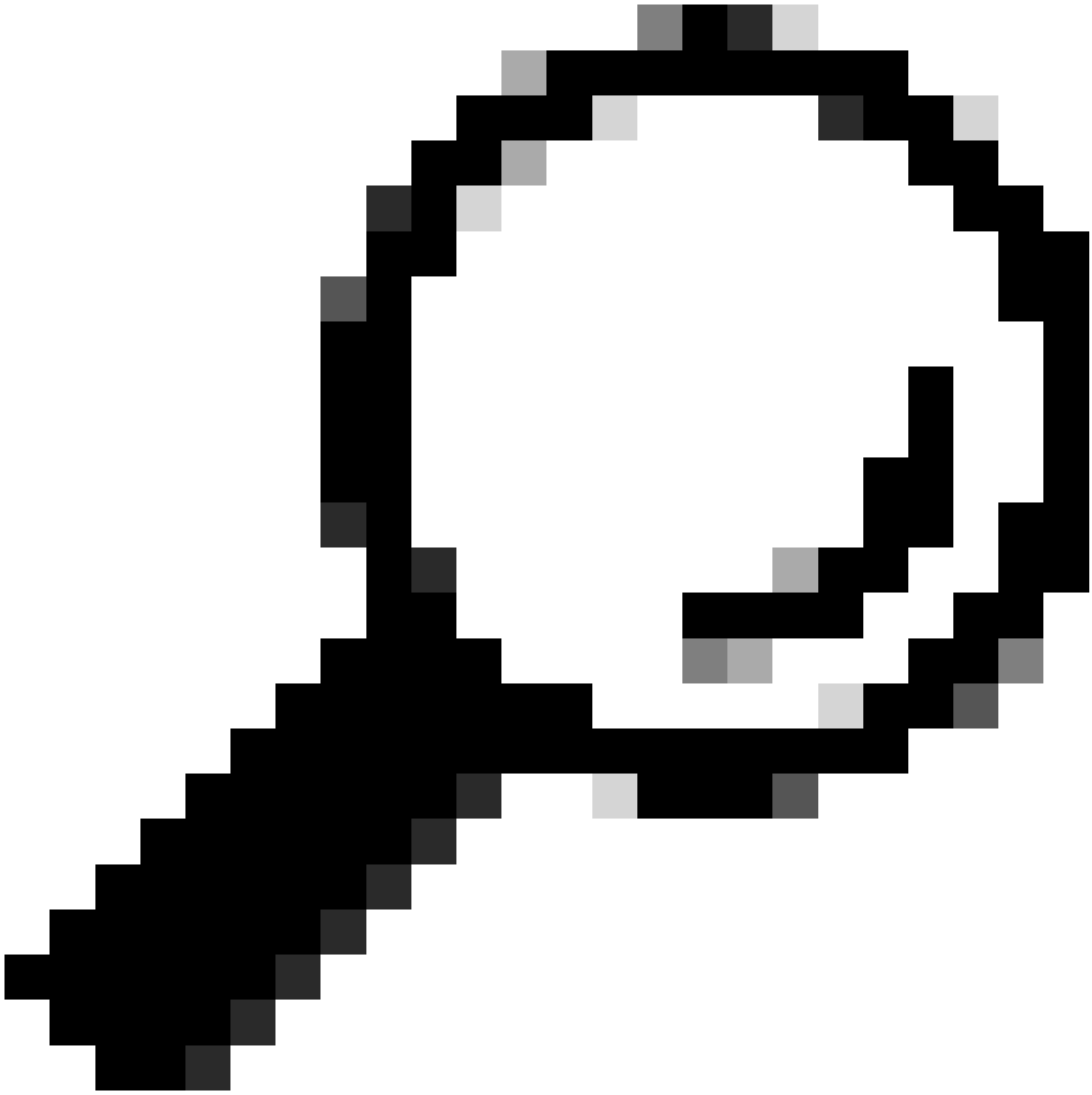
```
token = json.loads(response.text)  # Converting the response.text string value into a dictionary (It is
token = (token["Token"])  # Extracting just the token value by specifying the key as a parameter.
```

Um zu überprüfen, ob die Token-Variable nur das Token als Wert enthält, drucken Sie es aus.

```
print(token)
```

Ausgabebeispiel:

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



Tipp: Da jedes generierte Token standardmäßig in 1 Stunde abläuft, kann eine Python-Methode, die den Code zum Generieren eines Tokens enthält, jedes Mal erstellt und aufgerufen werden, wenn ein Token abläuft, ohne dass das gesamte Programm durch einen Aufruf der erstellten Methode ausgeführt werden muss.

## Testen einer API

Nachdem das Token der Token-Variablen erfolgreich zugewiesen wurde, können die verfügbaren Catalyst Center-APIs verwendet werden.

In diesem Fall wird die Cisco DNA Center Nodes Configuration Summary API getestet.

Cisco DNA Center Nodes - Konfigurationsübersicht

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

Diese API liefert Details zur aktuellen Konfiguration von Catalyst Center, z. B. konfigurierte NTP-Server, Knotenname, clusterinterne Verbindung, LACP-Modus usw.

Die Cisco DNA Center Nodes Configuration Summary API gibt in diesem Fall an, dass als Methode GET verwendet wird, die URL lautet "/dna/intent/api/v1/nodes-config" und da die Token-Zeichenfolge extrahiert und der Token-Variablen zugewiesen wurde, wird das Token diesmal als Variable im Header des API-Aufrufs als "X-Auth-Token" übergeben en': gefolgt vom Token.

Dadurch wird die Anforderung an Catalyst Center für jeden durchgeführten API-Aufruf authentifiziert. Beachten Sie, dass jeder Token eine Stunde dauert. Nach einer Stunde muss ein neues Token generiert werden, um weiterhin API-Aufrufe an Catalyst Center durchführen zu können.

Fahren Sie mit der Erstellung der Variablen zum Testen der API fort:

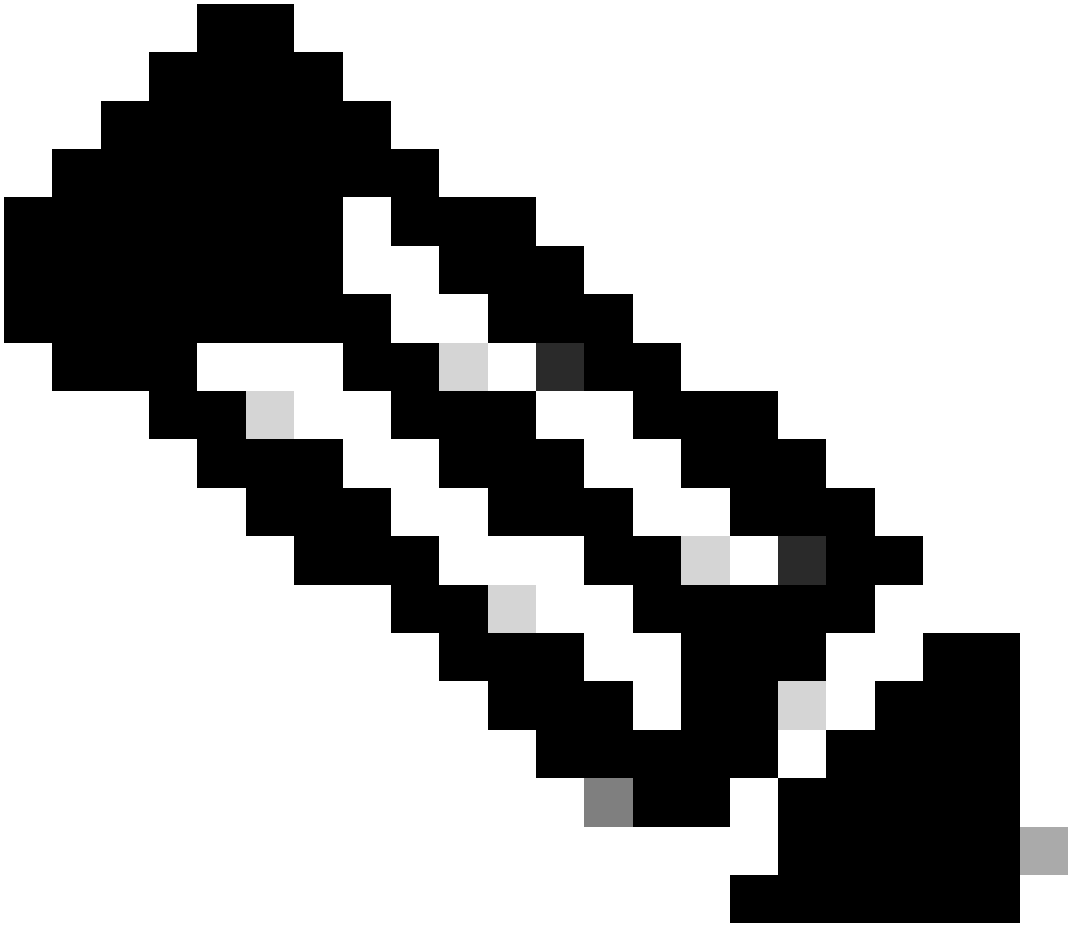
```
nodeInfo_url = "https://<CatalystCenterIP>/dna/intent/api/v1/nodes-config"
nodeInfo_headers = {
    'X-Auth-Token': token
}

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

nodeInfo_url-Variable wurde erstellt, um die URL unserer API zu speichern. nodeInfo_headers-Variable speichert die Header für unsere API. In diesem Fall wurden "X-Auth-Token" und die Token-Variable als Parameter übergeben, um die Anforderung erfolgreich an Catalyst Center zu authentifizieren. Schließlich speichert die nodeInfoResponse-Variable die Antwort der API.

Um die empfangene Antwort zu validieren, können Sie die print() -Funktion verwenden.

Ausgabebeispiel:

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



Hinweis: Wenn ein selbstsigniertes Zertifikat in Catalyst Center verwendet wird, kann die API-Anforderung mit dem nächsten Fehler fehlschlagen:

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

Um dieses Problem zu beheben, müssen Sie der Anforderung den verify-Parameter als False hinzufügen. Dadurch wird die Überprüfung des SSL-Zertifikats vom Endpunkt (Catalyst Center) unterdrückt.

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

Die von der API empfangene Antwort kann schwer zu lesen sein. Mithilfe des json()-Moduls kann

die Antwort in einer besser lesbaren Zeichenfolge ausgegeben werden. Zuerst muss die API-Antwort in ein JSON-Objekt geladen werden. Dazu wird die json.loads()-Funktion gefolgt von der json.dumps()-Funktion verwendet:

```
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: Diese Funktion gibt das JSON-Objekt zurück, das als Parameter in einer mit JSON formatierten Zeichenfolge verwendet wurde.

indown: Dieser Parameter definiert die Einzugsebene für die JSON-formatierte Zeichenfolge.

Ausgabebeispiel:

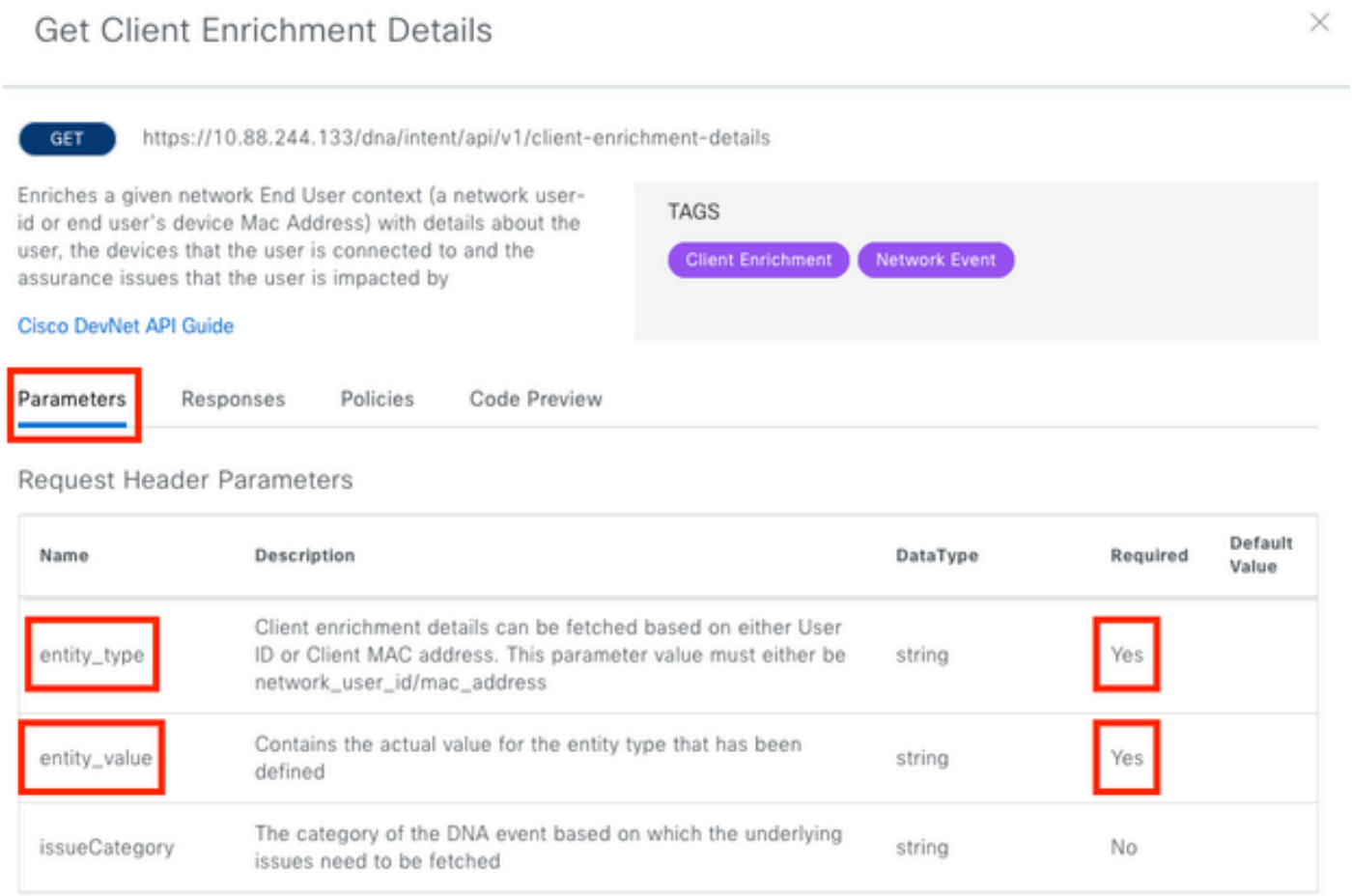
```
{
  "response": {
    "nodes": [
      {
        "name": "X.X.X.X",
        "id": "ea5dbec1-fbb6-4339-9242-7694eb1xXxX",
        "network": [
          {
            "slave": [
              "enp9s0"
            ],
            "lacp_supported": true,
            "intra_cluster_link": false,
!---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

## APIs mit Header-Parametern

Es gibt einige APIs, für die einige Parameter im Header gesendet werden müssen, damit sie wie erwartet funktionieren. In diesem Fall wird die Get Client Enrichment Details API getestet.

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

Um zu überprüfen, welche Header-Parameter erforderlich sind, damit die API wie erwartet funktioniert, navigieren Sie zu Platform > Developer Toolkit > APIs > Get Client Enrichment Details (Details zur Client-Bereicherung abrufen) und klicken Sie auf den Namen der API. Ein neues Fenster wird geöffnet, und unter der Option Parameter werden Headers-Parameter angezeigt, die für die Funktion der API erforderlich sind.




In diesem Fall kann für den Parameter entity_type gemäß der Beschreibung der Wert entweder network_user_id oder mac_address sein, und der Parameter entity_value muss den Wert für den definierten Entitätstyp enthalten.

Um fortzufahren, werden zwei neue Variablen definiert, entity_type und entity_value mit den entsprechenden Werten:

```
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
```

Außerdem werden neue Variablen erstellt, um den API-Aufruf auszuführen. Die URL des API-Aufrufs wird in der userEnrichment_url-Variable gespeichert. Header werden in der userEnrichmentHeaders-Variablen gespeichert. Die empfangene Antwort wird in der userEnrichmentResponse-Variable gespeichert.

```
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)
```

Wie Sie sehen können, wurden aus userEnrichmentHeaders die Variablen entity_type und entity_value zusammen mit der Token-Variablen als Header-Parameter für den API-Aufruf übergeben.

Um die empfangene Antwort zu validieren, verwenden Sie die print() -Funktion.

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

Ausgabebeispiel:

```
[ {
  "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
```

## APIs mit Abfrageparametern

Abfrageparameter können verwendet werden, um eine bestimmte Anzahl von Ergebnissen zu filtern, die von einer API zurückgegeben werden. Diese Parameter werden der URL der API

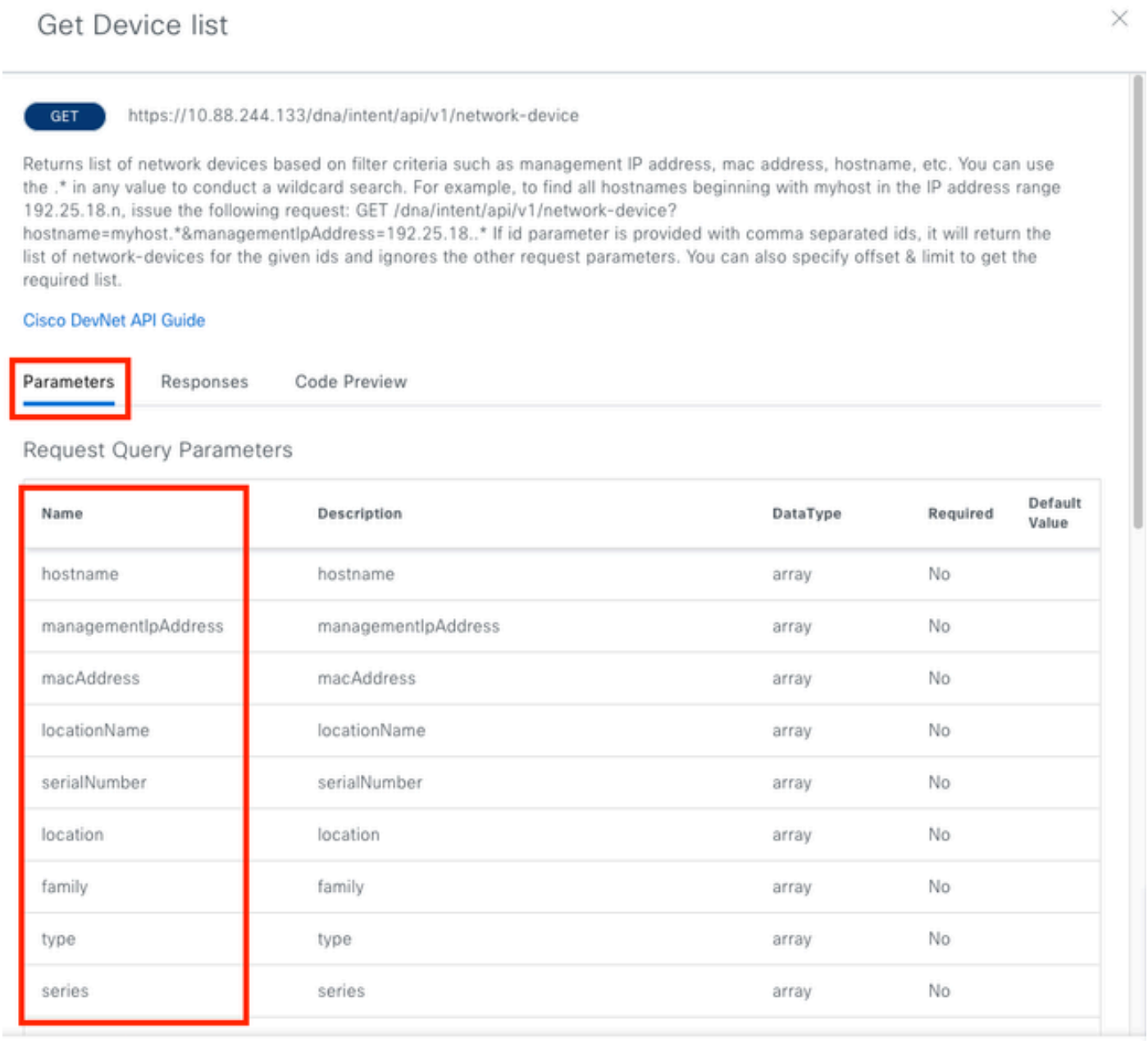
hinzugefügt.

Der API-Aufruf der Get Device List wurde getestet.

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

Die Get Device List-API gibt eine Liste aller Geräte zurück, die in Catalyst Center hinzugefügt wurden. Wenn Details für ein bestimmtes Gerät angefordert werden, können Abfrageparameter dazu beitragen, bestimmte Informationen zu filtern.

Um zu überprüfen, welche Abfrageparameter für die API verfügbar sind, navigieren Sie zu Platform > Developer Toolkit > APIs > Get Device List, und klicken Sie auf den Namen der API. Es wird ein neues Fenster geöffnet, und unter der Option Parameter werden die für die API verfügbaren Abfrageparameter angezeigt.




In diesem Beispiel werden managementIpAddress- und serialNumber-Abfrageparameter verwendet (berücksichtigen Sie, dass nicht alle Abfrageparameter für den API-Aufruf verwendet werden müssen). Fahren Sie mit dem Erstellen und Zuweisen der entsprechenden Werte für beide Abfrageparameter fort.

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

Wie bereits erwähnt, werden die Abfrageparameter in der URL der API hinzugefügt, und zwar am Ende der API. Dazu wird ein "?" gefolgt von den Abfrageparametern verwendet.

Wenn mehrere Abfrageparameter verwendet werden, wird dazwischen ein "&"-Zeichen eingefügt, um eine so genannte Abfragezeichenfolge zu bilden.

Im nächsten Beispiel wird veranschaulicht, wie die Abfrageparameter der deviceListUrl-Variablen hinzugefügt werden, in der die URL des API-Aufrufs gespeichert wird.

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

Beachten Sie, dass die zuvor erstellten Variablen an die URL-Zeichenfolge angehängt wurden. Anders ausgedrückt, sieht die gesamte Zeichenfolge der URL wie folgt aus:

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

Wenn Sie mit dem API-Aufruf fortfahren, wird die deviceListHeaders-Variable erstellt, um die API-Header zusammen mit der als Parameter übergebenen Token-Variable zu speichern, und die deviceListResponse-Variable speichert die API-Antwort.

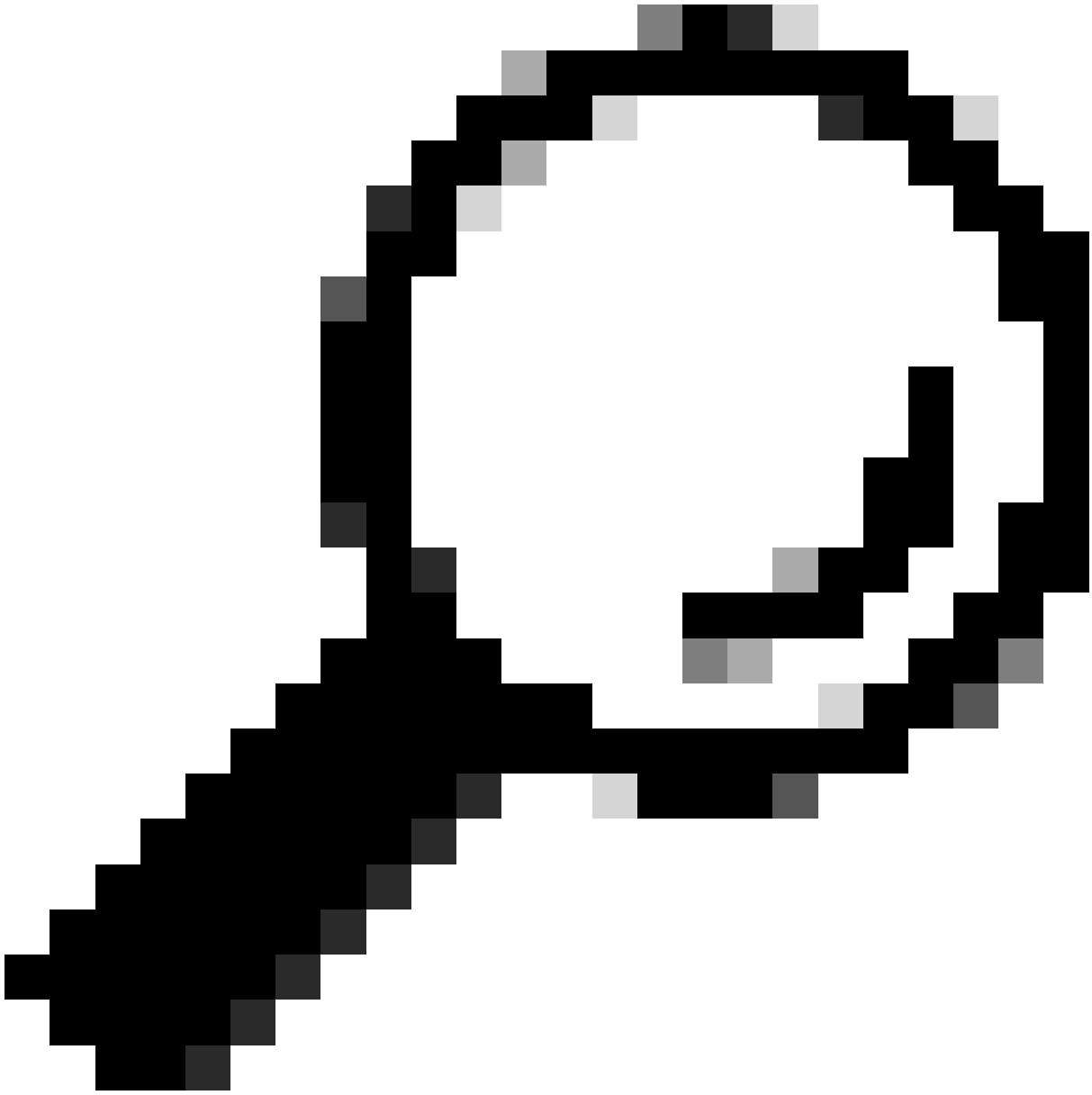
```
deviceListHeaders = {
            'X-Auth-Token': token,
        }

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

Um die empfangene Antwort zu validieren, können Sie die print() -Funktion verwenden.

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

Ausgabebeispiel:

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



Tipp: Um die Antwort lesbarer zu drucken, können Sie die Funktionen json.loads() und json.dumps() verwenden, die im Abschnitt Testing API beschrieben werden.

Informationen zu dieser Übersetzung

Cisco hat dieses Dokument maschinell übersetzen und von einem menschlichen Übersetzer editieren und korrigieren lassen, um unseren Benutzern auf der ganzen Welt Support-Inhalte in ihrer eigenen Sprache zu bieten. Bitte beachten Sie, dass selbst die beste maschinelle Übersetzung nicht so genau ist wie eine von einem professionellen Übersetzer angefertigte. Cisco Systems, Inc. übernimmt keine Haftung für die Richtigkeit dieser Übersetzungen und empfiehlt, immer das englische Originaldokument (siehe bereitgestellter Link) heranzuziehen.