Python에서 Catalyst Center API 사용

목차

소개

이 문서에서는 Cisco Catalyst Center에서 Python을 사용하여 사용할 수 있는 다양한 API를 사용하는 방법에 대해 설명합니다.

사전 요구 사항

요구 사항

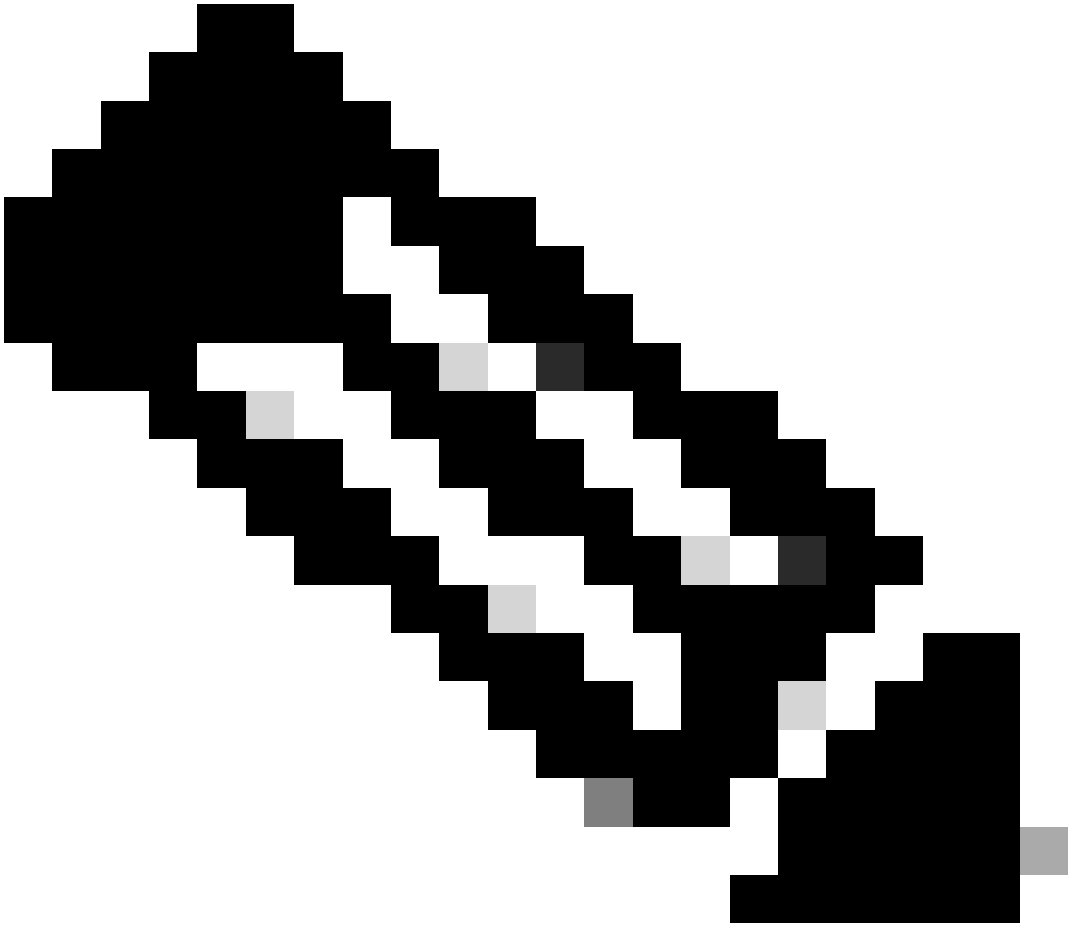
기본 지식:

- Cisco Catalyst 센터
- API
- 비단뱀

사용되는 구성 요소

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

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 미리 숙지하시기 바랍니다.

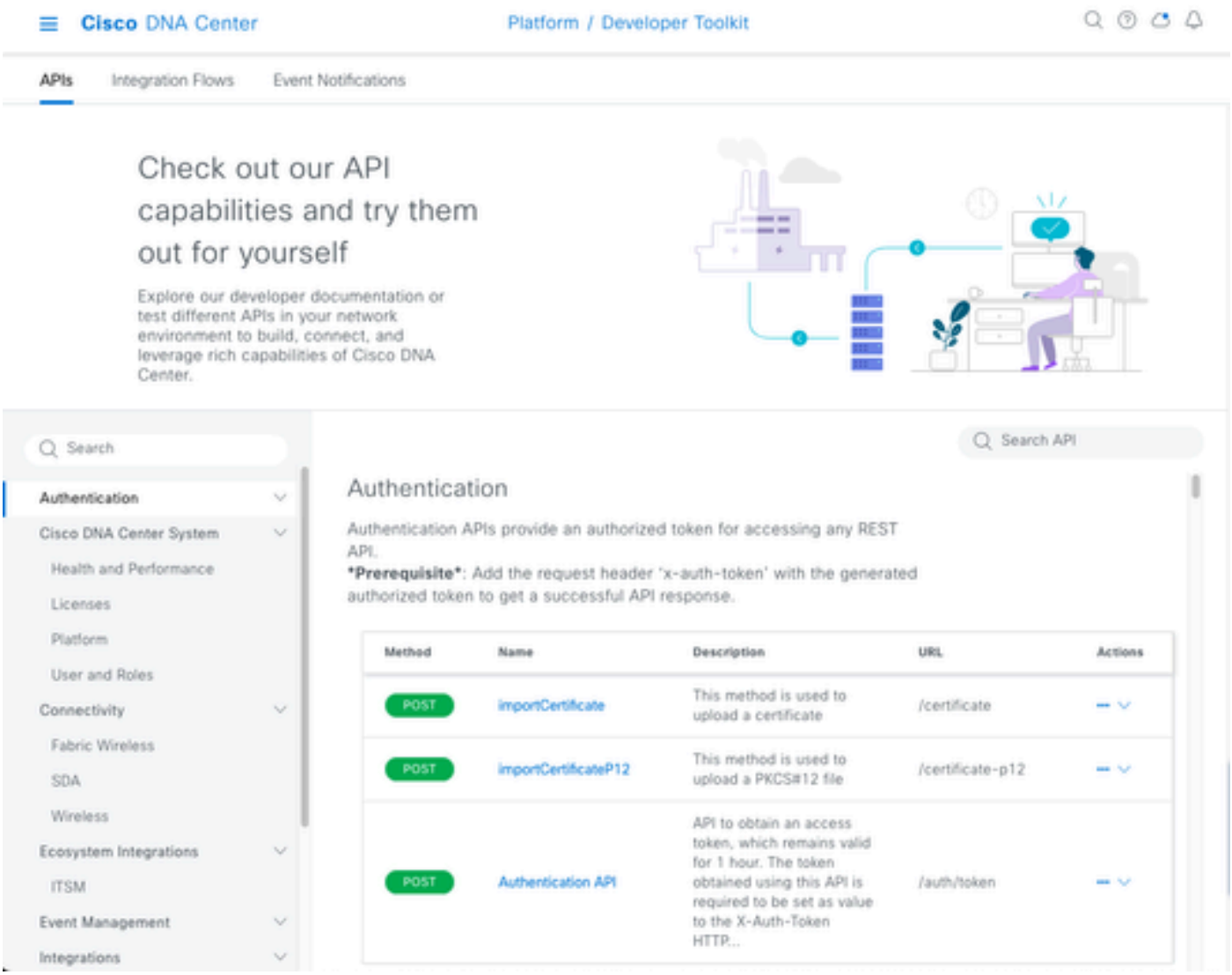


참고: Cisco TAC(Technical Assistance Center)에서는 Python에 대한 기술 지원을 제공하지 않습니다. Python에 문제가 있는 경우 Python 지원에 기술 지원을 문의하십시오.

구성

개요

Cisco Catalyst Center에는 여러 API가 있습니다. 사용할 수 있는 API를 확인하려면 Catalyst Center에서 Platform(플랫폼) > Developer Toolkit(개발자 툴킷) > APIs로 이동합니다.

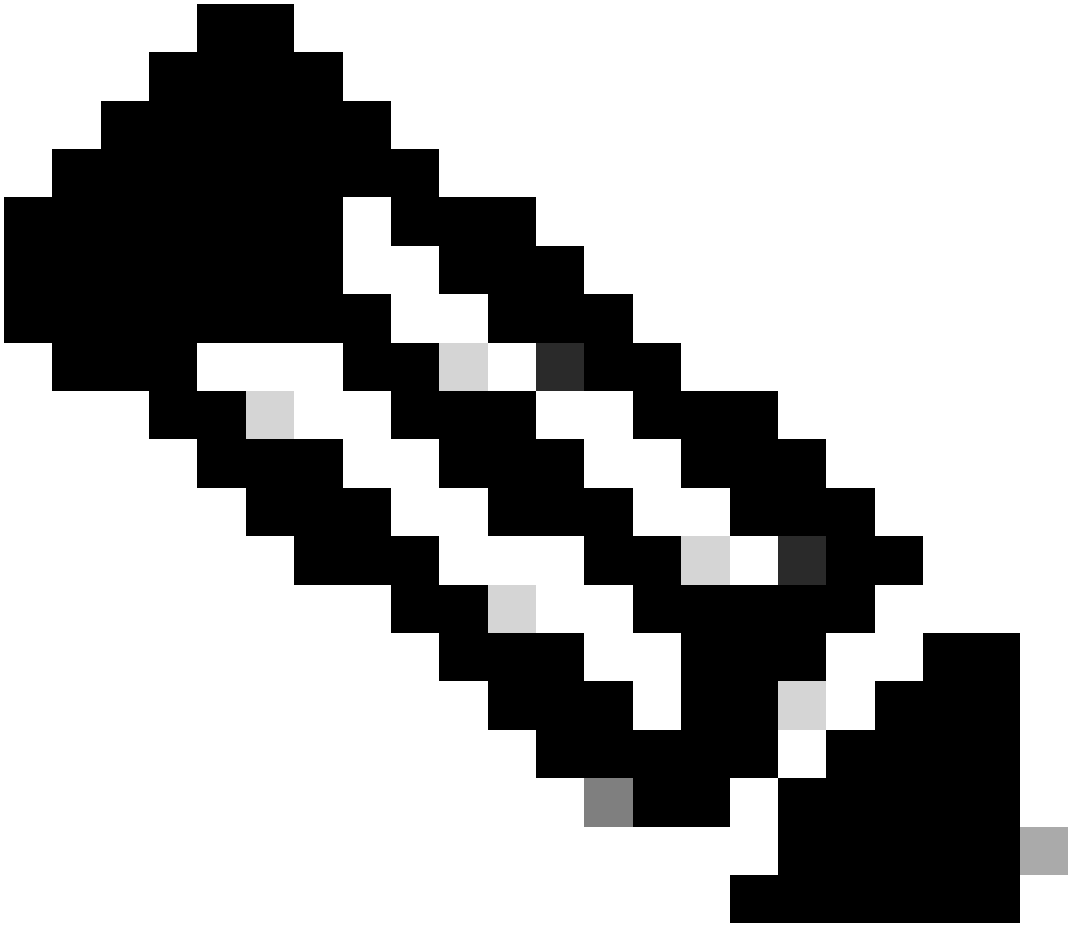


Catalyst Center API 페이지

모든 API는 Catalyst Center에서 수행해야 하는 작업 또는 정보에 따라 고유한 목적을 갖습니다. API가 작동하려면 사전 요구 사항으로 토큰을 사용하여 Catalyst Center를 올바르게 인증하고 성공적인 API 응답을 받아야 합니다. 그에 따라 토큰은 REST 호출자에 대한 권한을 식별합니다.

또한 다음과 같은 API를 구성하는 구성 요소를 식별하는 것이 중요합니다.

- URL: 특정 리소스에 대한 액세스를 제공하는 엔드포인트입니다.
- 메서드: 모든 API는 메서드를 포함해야 합니다. 클라이언트가 특정 엔드포인트에 대해 수행하고자 하는 작업/작업을 정의합니다. 예: 게시물, 가져오기, 넣기, 삭제
- 헤더: 요청에 대한 추가 정보를 키-값 쌍 형식으로 제공합니다. 예를 들어, 권한 부여 헤더는 자격 증명을 사용하는 인증 방법을 제공합니다.
- 매개변수: API를 사용하여 엔드포인트에 특정 명령을 제공하는 변수입니다. 매개변수는 엔드포인트의 URL에 속할 수 있습니다.
- 페이로드: API 호출 중에 엔드포인트로 전송해야 하는 데이터입니다.

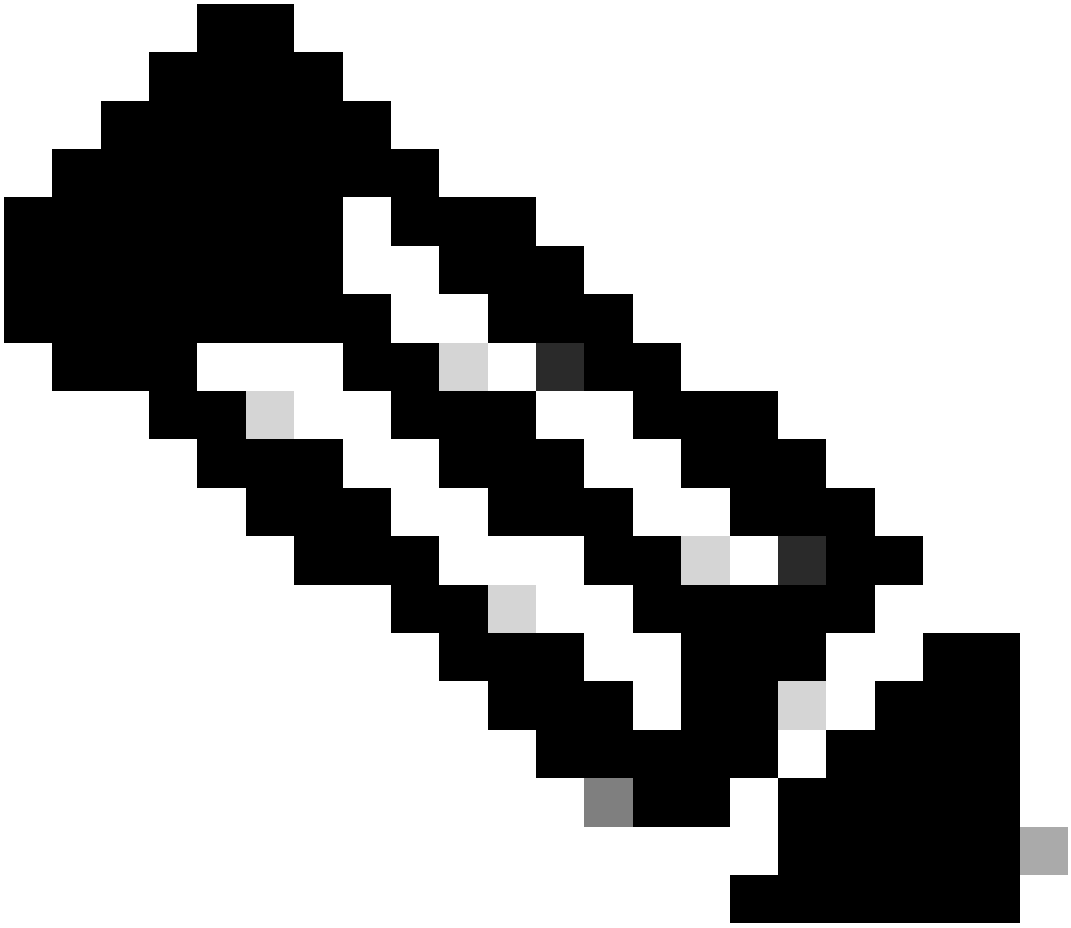


참고: Catalyst Center에서 사용할 수 있는 각 API에 대한 자세한 내용은 API 참조 설명서를 참조하십시오.

모듈

사용된 파이썬 모듈:

- 요청:이 모듈은 HTTP/1.1 요청을 특정 URL에 보낼 수 있습니다. 모듈에 대한 자세한 내용은 요청 모듈 가이드를 참조하십시오.
- base64: 인코딩 및 디코딩 기능을 제공합니다. 모듈에 대한 자세한 내용은 base64 모듈 가이드를 참조하십시오.
- json: 이 모듈에서는 API 응답에서 특정 데이터를 가져올 수 있습니다. 모듈에 대한 자세한 내용은 json module guide를 참조하십시오.



참고: Python 모듈을 설치하는 방법에 대한 자세한 내용은 Python 모듈 설치 설명서를 참조하십시오.

토큰 생성

새 토큰을 생성하려면 Authentication API라는 API를 사용해야 합니다.

인증 API:

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

생성된 토큰은 1시간 동안 유효하다는 것을 언급하는 것이 중요하다. 1시간 후에는 위에서 언급한 동일한 API를 사용하여 새 토큰을 생성해야 합니다.

새 Python 파일에서 모듈(요청, base64 및 json)을 가져온 다음 4개의 변수를 생성합니다.

```
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"
```

인증 API는 헤더에서 권한 부여 토큰으로 기본 인증을 지원합니다. 기본 인증은 엔드포인트를 인증하는 데 사용할 수 있는 방법으로, 사용자 이름과 비밀번호를 콜론으로 구분하여 제공합니다(사용자 이름:비밀번호). 두 값은 모두 base64로 인코딩되며 엔드포인트는 로그인 자격 증명을 디코딩하고 사용자가 액세스할 수 있는지 여부를 확인합니다.

사용자 이름 및 비밀번호에 대한 Base64 인코딩 문자열을 생성하려면 base64 모듈이 사용됩니다. 이를 위해 b64encode 함수를 사용합니다.

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

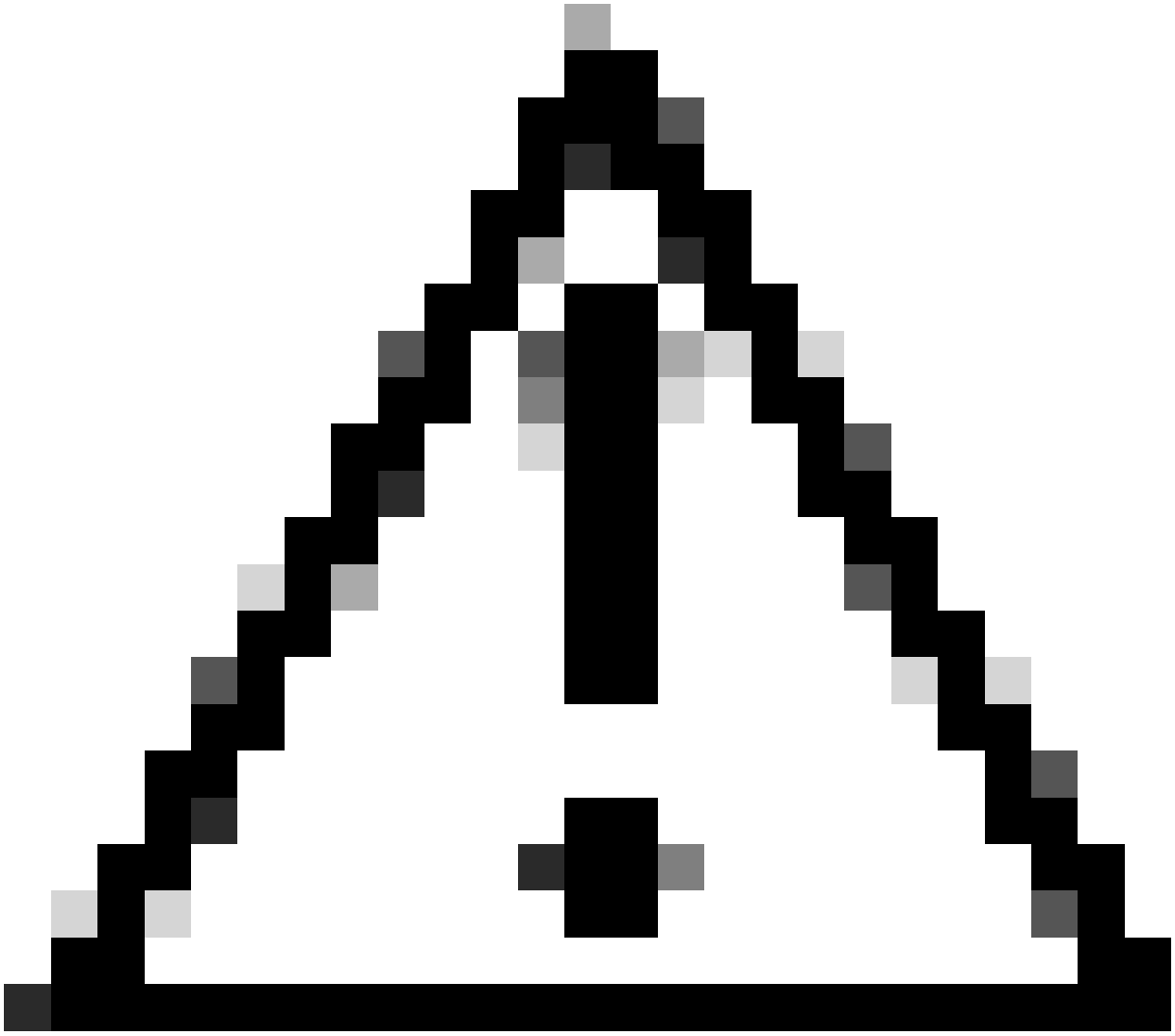
위의 코드에서 byte_string 변수는 '.encode("ascii")' 함수를 사용하여 생성되었습니다. 이는 base64.b64encode 함수에 bytes 형식의 개체가 필요하기 때문입니다. 또한 사용자 및 비밀번호 변수를 사용하여 문자열 형식 'user:password'를 유지했습니다. 마지막으로 사용자 및 비밀번호로 base64 인코딩 바이트 문자열을 만들었습니다. 'decode()' 메서드를 사용하여 값을 str 객체로 변환했습니다.

이를 확인하려면 authorizationBase64 변수의 값을 인쇄할 수 있습니다.

```
print(authorizationBase64)
```

출력 예:

```
am9yZ2QhbDI6Sm9yZ2VhbDXxXxXx
```



주의: base64는 암호화 알고리즘이 아닙니다. 보안 목적으로 사용해서는 안 됩니다. 인증 API는 또한 AES 키 암호화를 Authorization 토큰으로 헤더에서 지원하여 보안을 강화합니다.

사용자 및 비밀번호를 사용하여 base64 인코딩 문자열을 생성한 Catalyst Center를 인증하면 모듈 요청을 사용하여 API 인증 API 호출을 진행합니다. 또한, request라는 기능을 통해, 요청의 텍스트가 포함된 응답 객체를 가져올 수 있습니다.

메서드의 구문:

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

**kwargs는 쿠키, 사용자 에이전트, 페이로드, 헤더 등 요청에 전달된 모든 매개변수를 의미합니다.

Authentication API는 메서드가 POST, URL이"/dna/system/api/v1/auth/token"이고 기본 인증을

Header에서 지정해야 함을지정합니다.

이러한 변수는 request() 함수에 사용하기 위해 생성됩니다.

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

headers 변수에는 두 가지를 지정하였다. 첫 번째 유형은 content-type으로, 엔드포인트로 전송된 리소스의 미디어 유형을 지정합니다. 이는 엔드포인트가 데이터를 정확하게 구문 분석하고 처리하는 데 도움이 됩니다. 두 번째는 Authorization입니다. 이 경우 변수 authorizationBase64(base64 인코딩 문자열 저장)가 Catalyst Center에 인증하기 위한 매개변수로 전송됩니다.

이제 request() 함수를 사용하여 API 호출을 수행합니다. 다음 코드는 함수의 구문을 보여줍니다.

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

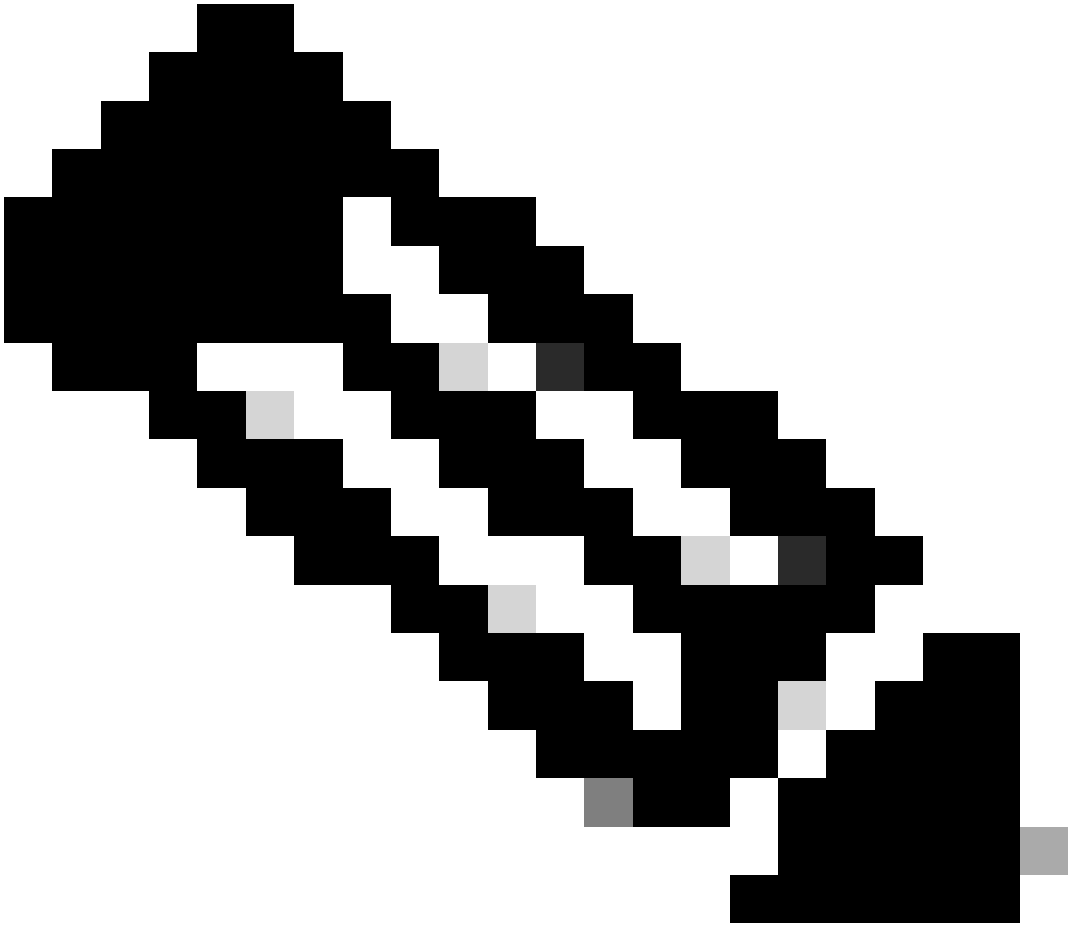
응답 변수는 API 호출의 데이터를 저장하기 위해 생성되었습니다.

얻은 응답을 인쇄하려면 응답 변수의 text() 메서드와 함께 print 함수를 사용합니다. text() 메서드는 Catalyst Center에서 받은 응답으로 str 객체를 생성합니다.

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

출력 예:

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



참고: Catalyst Center에서 자체 서명 인증서를 사용하는 경우 API 요청이 실패하고 다음 오류가 발생할 수 있습니다.

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

이 문제를 해결하려면 요청 함수에 verify 매개 변수를 False로 추가해야 합니다. 이는 엔드포인트(Catalyst Center)에서 SSL 인증서를 확인하는 것을 무시합니다.

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

API 인증 호출에서 받은 응답에서는 구조가 Python의 사전과 유사하지만 str 객체입니다.

객체의 유형을 검증하려면 type() 함수를 사용합니다.

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

다음 출력을 반환합니다.

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

다른 Catalyst Center API를 사용하려면 토큰만 매개변수로 전달되어야 하므로 실용적인 목적상 전체 문자열이 아닌 API에서 받은 응답에서 토큰 값만 추출하면 됩니다.

API 호출로부터 수신된 응답은 Python의 사전과 유사한 구조를 가지지만 객체 유형은 str이므로, json 모듈을 이용하여 상기 객체를 사전으로 변환해야 한다. 이렇게 하면 API에서 받은 전체 문자열에서 토큰 값이 추출됩니다.

이를 위해 json.loads() 함수는 문자열을 사전으로 변환하여 나중에 토큰 값만 추출하고 토큰 변수에 직접 할당합니다.

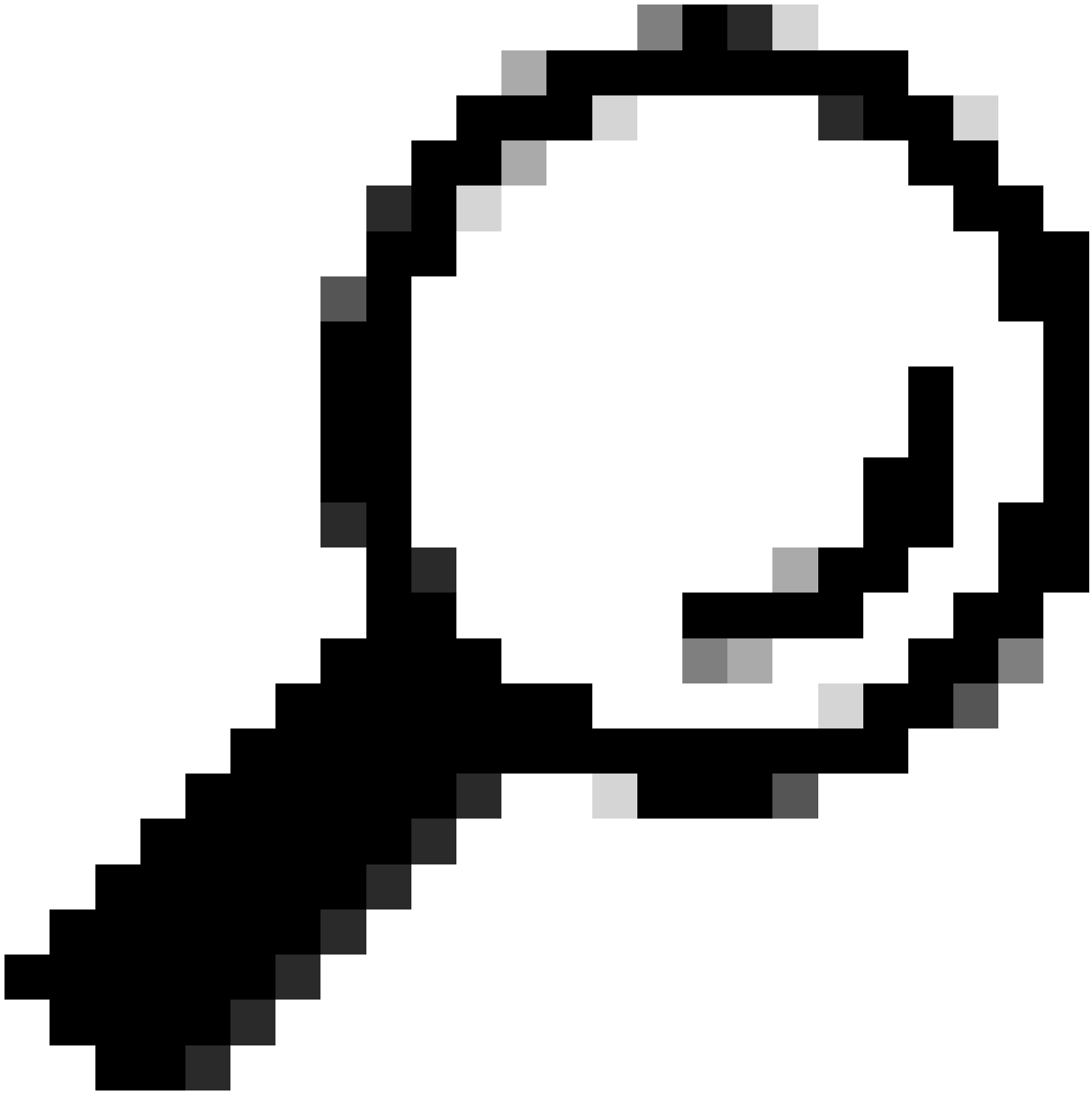
```
token = json.loads(response.text)  # Converting the response.text string value into a dictionary (It is
token = (token["Token"])  # Extracting just the token value by specifying the key as a parameter.
```

토큰 변수의 값이 토큰뿐인지 확인하려면 계속 인쇄합니다.

```
print(token)
```

출력 예:

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



팁: 생성된 각 토큰은 기본적으로 1시간 후에 만료되므로 토큰을 생성하기 위한 코드를 포함하는 Python 메서드를 만들고, 생성된 메서드를 호출하여 전체 프로그램을 실행하지 않고도 토큰이 만료될 때마다 호출할 수 있습니다.

API 테스트

토큰이 토큰 변수에 성공적으로 할당되었으므로 사용 가능한 Catalyst Center API를 사용할 수 있습니다.

이 경우 Cisco DNA Center Nodes Configuration Summary API가 테스트됩니다.

Cisco DNA Center Nodes 컨피그레이션 요약

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

이 API는 NTP 서버 구성, 노드 이름, 클러스터 내 링크, LACP 모드 등 Catalyst Center의 현재 구성에 대한 세부 정보를 제공합니다.

Cisco DNA Center Nodes Configuration Summary API는 이 경우 사용되는 방법이 GET이고, URL은 "/dna/intent/api/v1/nodes-config"이며, 토큰 문자열이 추출되어 토큰 변수에 할당되었으므로, 이번에는 토큰이 API 호출의 헤더에서 변수로 'X-Auth-Token'으로 전달됩니다:XXX 토큰이 뒤에 옵니다.

이렇게 하면 수행되는 모든 API 호출에 대해 Catalyst Center에 대한 요청이 인증됩니다. 각 토큰은 1시간 동안 유효합니다. 1시간 후 Catalyst Center에 대한 API 호출을 계속 수행하려면 새 토큰을 생성해야 합니다.

계속해서 API를 테스트할 변수를 생성합니다.

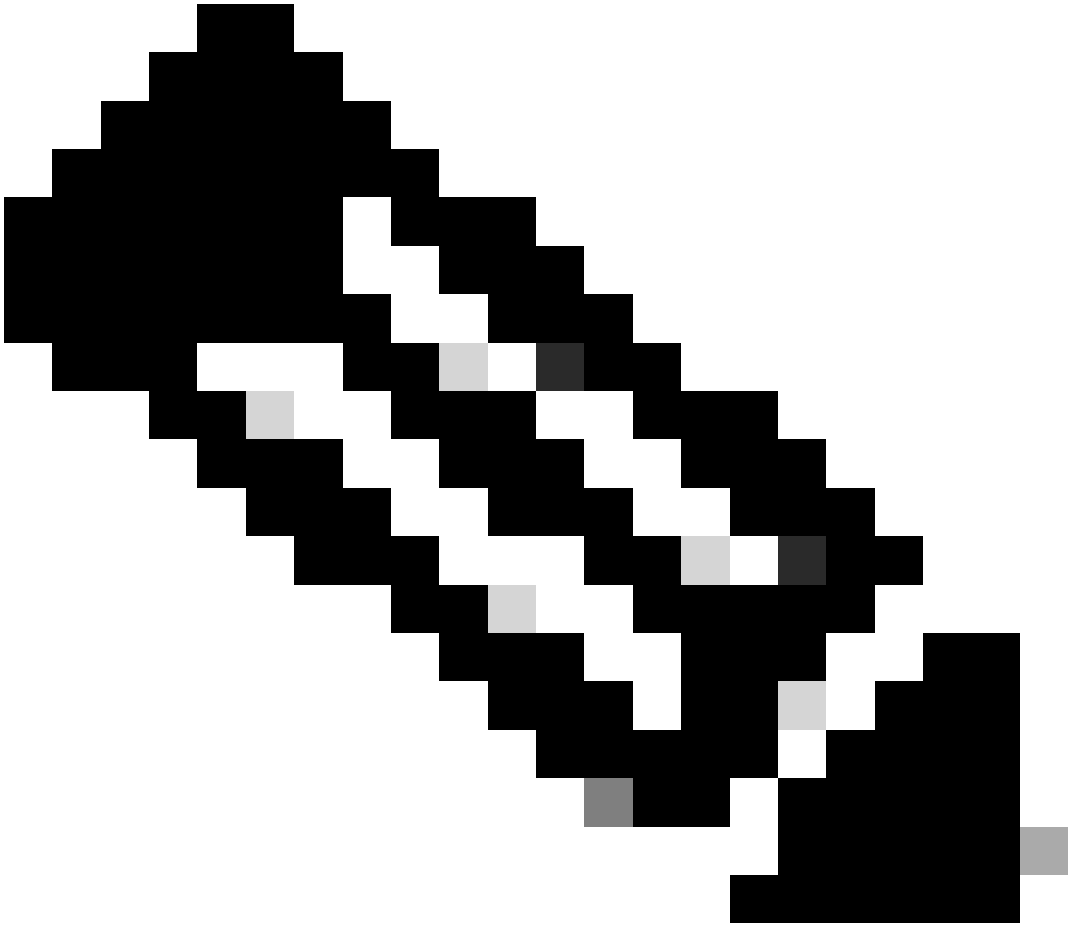
```
nodeInfo_url = "https://<CatalystCenterIP>/dna/intent/api/v1/nodes-config"
nodeInfo_headers = {
    'X-Auth-Token': token
}

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

nodeInfo_url 변수는 API의 URL을 저장하기 위해 만들어졌습니다. nodeInfo_headers 변수는 API의 헤더를 저장합니다. 이 경우 'X-Auth-Token:' 및 토큰 변수가 Catalyst Center에 요청을 인증하기 위한 매개변수로 전달되었습니다. 마지막으로 nodeInfoResponse 변수는 API의 응답을 저장합니다.

수신된 응답을 검증하려면 print() 함수를 사용할 수 있습니다.

출력 예:

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



참고: 자체 서명 인증서를 Catalyst Center에서 사용 중인 경우 API 요청이 실패하고 다음 오류가 발생할 수 있습니다.

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

이 문제를 해결하려면 요청에 verify 매개 변수를 False로 추가해야 합니다. 이렇게 하면 엔드포인트(Catalyst Center)에서 SSL 인증서를 확인하지 않습니다.

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

API로부터 수신된 응답은 판독하기 어려울 수 있다. json() 모듈을 사용하면 응답을 읽기 쉬운 문자열로 인쇄할 수 있습니다. 먼저 json.loads() 함수 다음에 json.dumps() 함수를 사용하여 API 응답을

JSON 객체로 로드해야 합니다.

```
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: 이 함수는 JSON 형식의 문자열에서 매개 변수로 가져온 JSON 개체를 반환합니다.

indent: 이 매개 변수는 JSON 형식의 문자열에 대한 들여쓰기 수준을 정의합니다.

출력 예:

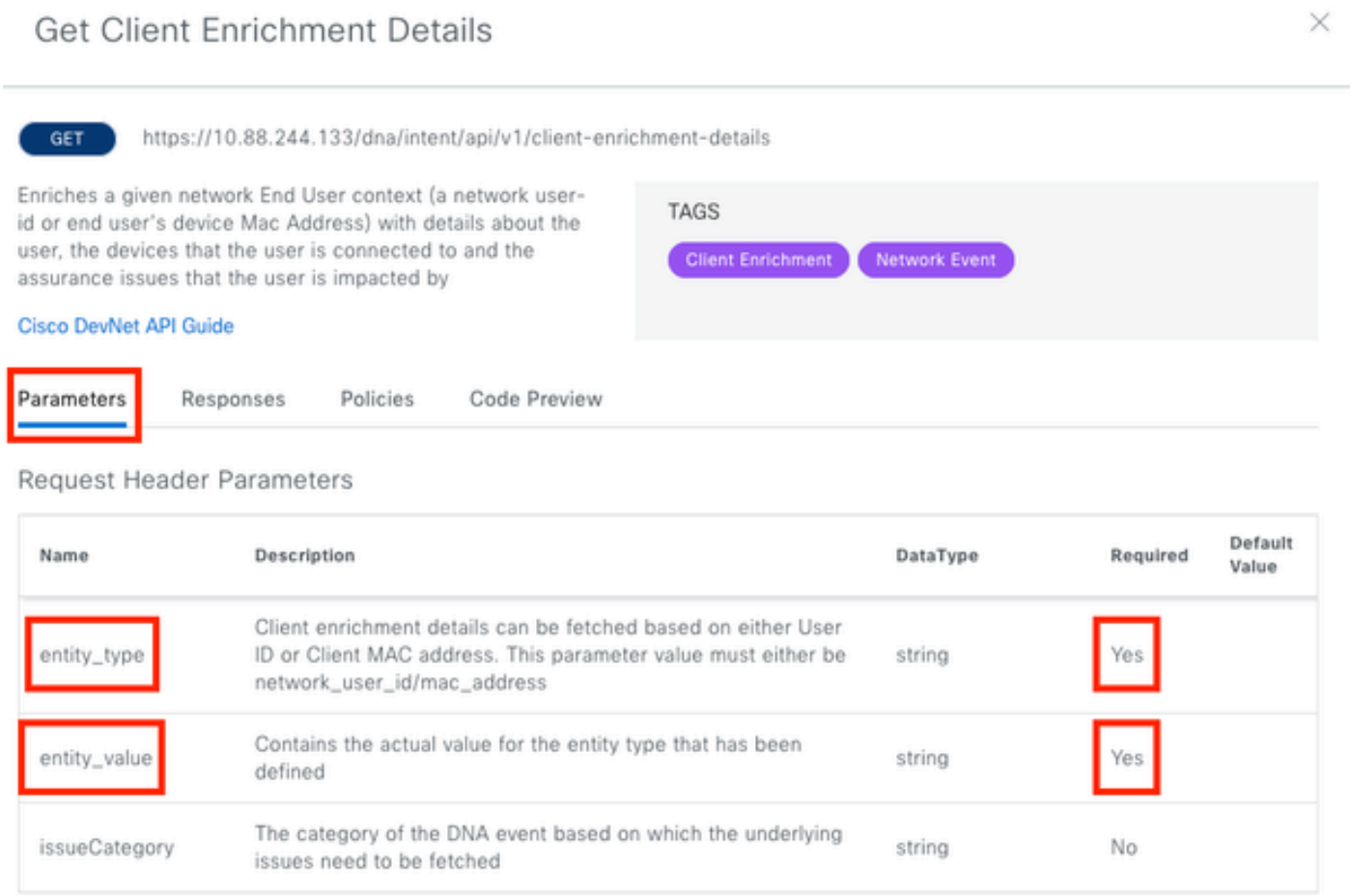
```
{
  "response": {
    "nodes": [
      {
        "name": "X.X.X.X",
        "id": "ea5dbec1-fbb6-4339-9242-7694eb1xXxX",
        "network": [
          {
            "slave": [
              "enp9s0"
            ],
            "lacp_supported": true,
            "intra_cluster_link": false,
!---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

헤더 매개변수를 사용하는 API

일부 API는 헤더에서 일부 매개변수를 보내 예상대로 작동하도록 해야 합니다. 이 경우 Get Client Enrichment Details API가 테스트됩니다.

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

API가 예상대로 작동하는 데 필요한 헤더 매개변수를 확인하려면 Platform(플랫폼) > Developer Toolkit(개발자 툴킷) > APIs(API) > Get Client Enrichment Details(클라이언트 보강 세부사항 가져오기)로 이동하고 API의 이름을 클릭합니다. 새 창이 열리고 Parameters 옵션 아래에 API가 작동하는 데 필요한 Headers Parameters가 표시됩니다.



이 경우 entity_type 매개 변수의 설명에 따라 값은 network_user_id 또는 mac_address일 수 있으며 entity_value 매개 변수에는 정의된 엔터티 유형의 값이 포함되어야 합니다.

계속 진행하기 위해 두 개의 새 변수, 즉 entity_type과 entity_value가 해당 값과 함께 정의됩니다.

```
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
```

API 호출을 수행하기 위한 새로운 변수도 생성됩니다. API 호출의 URL은 userEnrichment_url 변수에 저장됩니다. 헤더는 userEnrichmentHeaders 변수에 저장됩니다. 받은 응답은 userEnrichmentResponse 변수에 저장됩니다.

```
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)
```

보다시피 userEnrichmentHeaders에서 entity_type 및 entity_value 변수는 토큰 변수와 함께 API 호출의 헤더 매개변수로 전달되었습니다.

수신된 응답을 검증하려면 print() 함수를 사용합니다.

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

출력 예:

```
[ {
  "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
```

쿼리 매개 변수가 있는 API

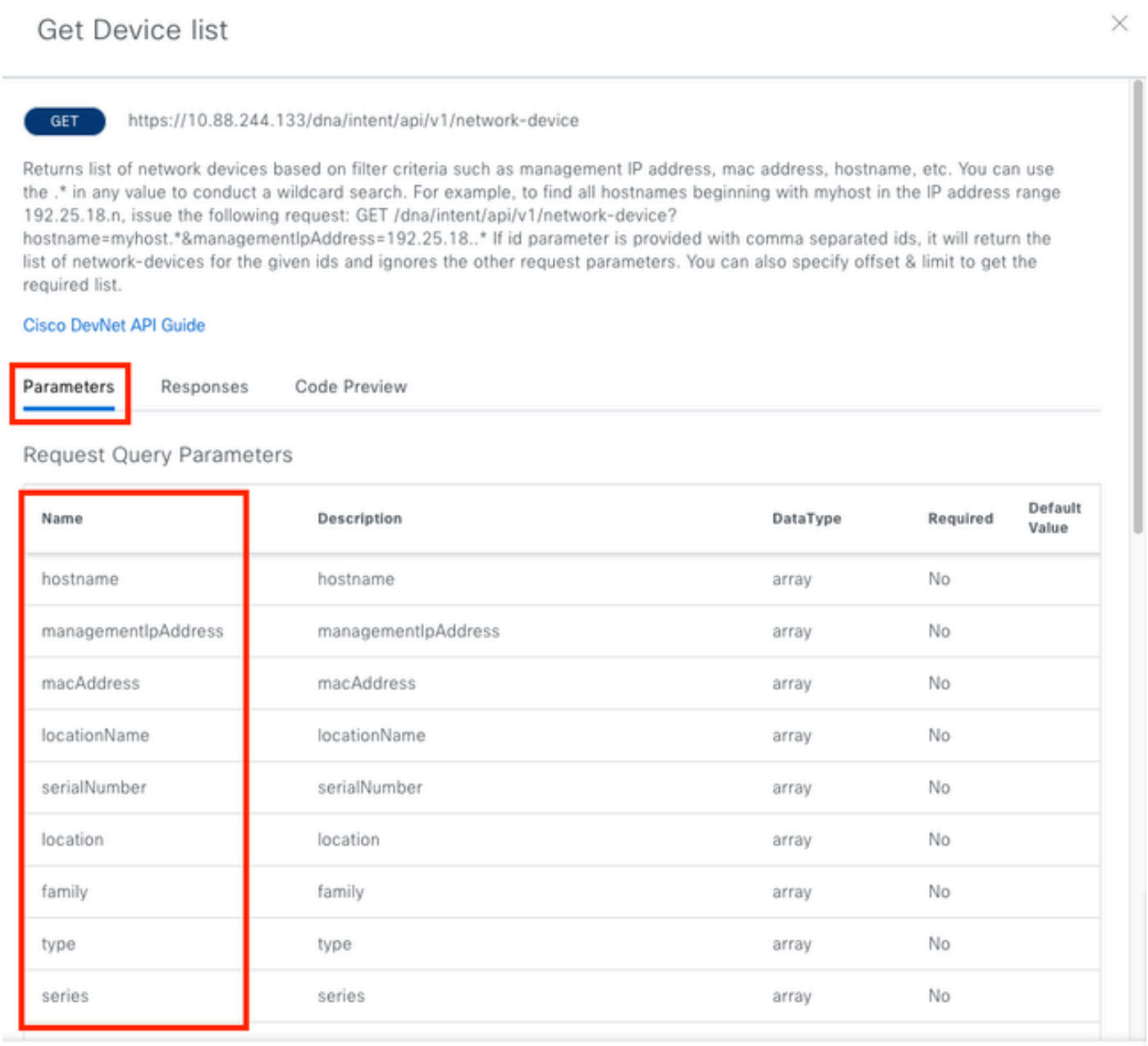
쿼리 매개 변수를 사용하여 API에서 반환되는 특정 개수의 결과를 필터링할 수 있습니다. 이러한 매개변수는 API의 URL에 추가됩니다.

Get Device List API 호출이 테스트됩니다.

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

Get Device List API는 Catalyst Center에 추가된 모든 디바이스의 목록을 반환합니다. 특정 장치에 대한 세부사항이 요청되면 쿼리 매개변수를 사용하여 특정 정보를 필터링할 수 있습니다.

API에 사용할 수 있는 쿼리 매개변수를 확인하려면 Platform(플랫폼) > Developer Toolkit(개발자 툴킷) > APIs(API) > Get Device List(디바이스 목록 가져오기)로 이동하고 API의 이름을 클릭합니다. 새 창이 열리고 Parameters 옵션 아래에 API에 사용할 수 있는 쿼리 매개변수가 표시됩니다.



이 예에서는 managementIpAddress 및 serialNumber 쿼리 매개 변수가 사용됩니다(API 호출에 모든 쿼리 매개 변수를 사용할 필요는 없다는 점을 고려함). 계속해서 두 쿼리 매개변수에 해당하는 값을 생성하고 할당합니다.

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

위에서 언급했듯이 쿼리 매개변수는 API의 URL에 추가되며, 특히 API의 끝에 '?'를 사용하고 그 뒤에 쿼리 매개변수가 추가됩니다.

여러 쿼리 매개 변수를 사용할 경우 &'' 기호가 쿼리 문자열 사이에 배치됩니다.

다음 예에서는 API 호출의 URL을 저장하는 deviceListUrl 변수에 쿼리 매개변수를 추가하는 방법을 보여줍니다.

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

이전에 생성한 변수가 URL 문자열에 추가되었습니다. 즉, URL의 전체 문자열은 다음과 같습니다.

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

API 호출을 계속 진행하면, deviceListHeaders 변수가 생성되어 API 헤더를 매개변수로 전달된 토큰 변수와 함께 저장하고 deviceListResponse 변수는 API 응답을 저장합니다.

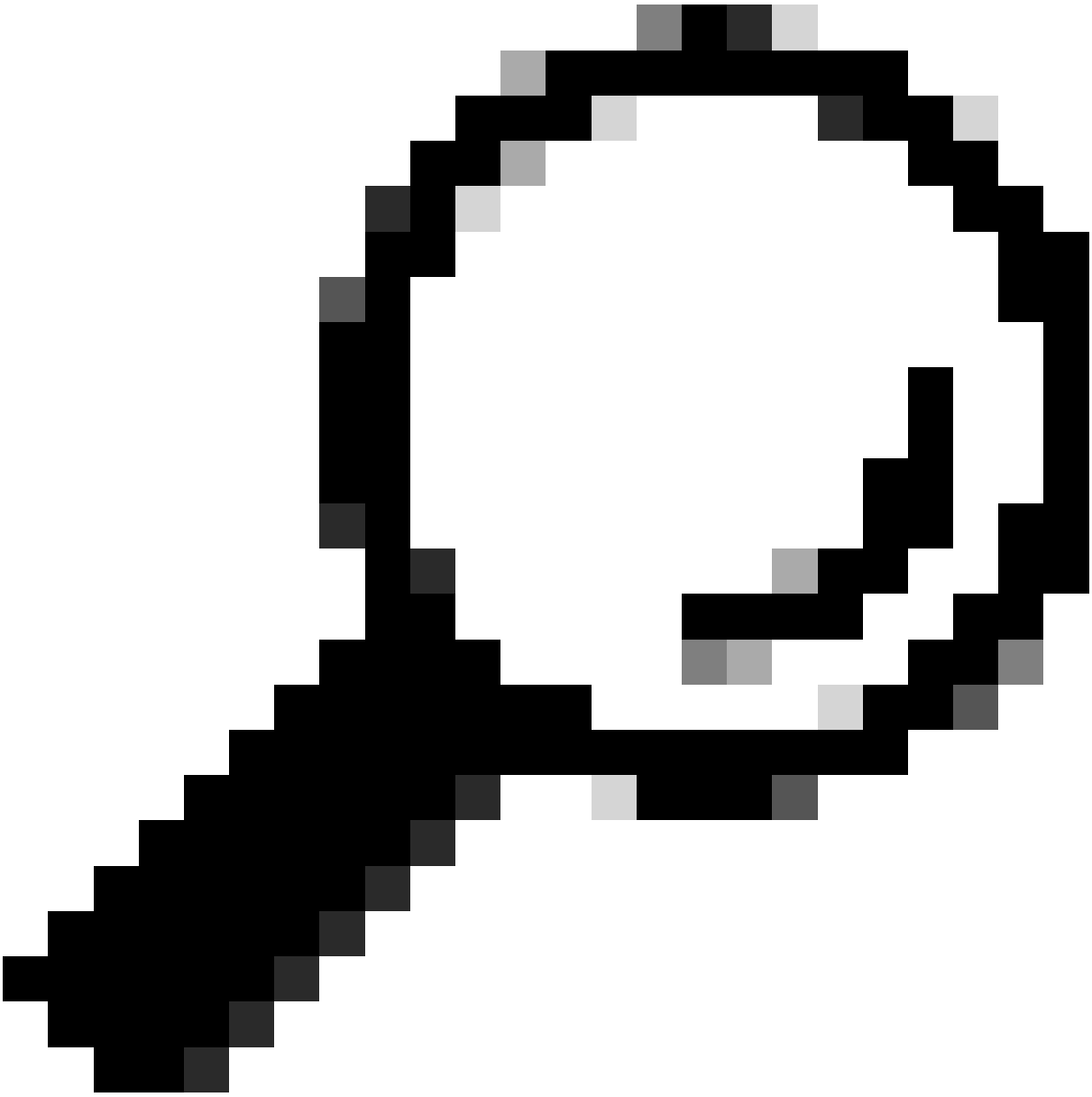
```
deviceListHeaders = {
            'X-Auth-Token': token,
        }

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

수신된 응답을 검증하려면 print() 함수를 사용할 수 있습니다.

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

출력 예:

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



팁: 응답을 좀 더 읽기 쉬운 방법으로 인쇄하려면 API 테스트 섹션에 설명된 json.loads() 및 json.dumps() 함수를 사용할 수 있습니다.

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.