

Catalyst Center API 이해 및 사용

목차

[소개](#)

[배경 정보](#)

[사전 요구 사항](#)

[요구 사항](#)

[사용되는 구성 요소](#)

[배경 정보](#)

[HTTP 소개](#)

[HTTP 요청](#)

[HTTP 응답](#)

[HTTP 메서드\(동사\)](#)

[HTTP 상태 코드](#)

[REST 개요](#)

[Catalyst Center 및 REST API](#)

[Bruno를 사용하여 인증 토큰 검색](#)

[1. API 엔드포인트 식별](#)

[2. 인증을 위해 Bruno 구성](#)

[3. Bruno에서 글로벌 헤더 설정](#)

[예 1: 사이트 층 생성](#)

[1단계: 상위 ID 가져오기](#)

[2단계: 층 생성](#)

[예 2: 인터페이스 정보 검색](#)

[1단계: 디바이스 ID 식별](#)

[2단계: Bruno에서 매개 변수가 있는 요청 만들기](#)

[결론](#)

소개

이 문서에서는 API 클라이언트 및 플랫폼의 내장 개발자 도구를 사용하여 CaTC 노스바운드(Intent) API에 액세스하고 이를 활용하는 방법에 대해 설명합니다.

배경 정보

기본적인 HTTP/REST 개념에 대해 설명하고 자동화 및 테스트에 대한 실용적인 예를 제공합니다.

사전 요구 사항

요구 사항

Cisco에서는 다음 항목에 대한 지식을 권장합니다.

- HTTP/1.1 프로토콜 및 상태 코드
- REST 아키텍처 스타일입니다.
- Cisco Catalyst Center(CaTC)에 대한 기본 이해
- Bruno(또는 Postman과 같은 유사한 API 클라이언트)에 대한 기본적인 이해

사용되는 구성 요소

이 문서의 정보는 다음 하드웨어 및 소프트웨어 버전을 기반으로 합니다.

- Cisco Catalyst Center 버전 2.3.7.x 및 3.x
- Bruno 버전 4.1.0

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.

배경 정보

Catalyst Center Northbound API(Intent API)를 사용하면 그래픽 사용자 인터페이스 외부에서 컨트롤러와 상호 작용할 수 있습니다. 이를 통해 맞춤형 워크플로의 자동화, 서드파티 툴과의 통합, 규모에 맞는 효율적인 네트워크 관리가 가능합니다.

HTTP 소개

HTTP(Hypertext Transfer Protocol)는 웹을 위한 데이터 통신의 기반입니다. 클라이언트가 요청을 전송하고 서버가 응답을 반환하는 클라이언트-서버 모델을 준수합니다.

HTTP 요청

HTTP 요청은 네 부분으로 구성됩니다.

1. 요청 라인: 메서드(GET 및 POST 포함), URI 및 프로토콜 버전(예: GET /api/v1/nodes HTTP/1.1)을 지정합니다.
2. 헤더: 인증 자격 증명 및 콘텐츠 유형과 같은 메타데이터를 제공합니다.
3. 빈 줄: 머리글을 본문에서 분리합니다.
4. 메시지 본문(선택 사항): 서버로 전송된 데이터를 포함합니다(POST/PUT/PATCH에서 공통).

HTTP 응답

HTTP 응답은 요청 구조를 미러링합니다.

1. 상태 줄: 상태 코드를 포함합니다(예: 200 OK).
2. 헤더: 서버 및 응답 데이터에 대한 세부 정보.
3. 메시지 본문: 요청된 실제 데이터(일반적으로 Catalyst Center의 경우 JSON 형식).

HTTP 메서드(동사)

- 다운로드: 리소스에서 데이터를 검색합니다.
- POST: 새 리소스를 만듭니다.
- PUT: 기존 리소스를 교체합니다.
- 패치: 리소스에 부분 수정을 적용합니다.
- 삭제: 리소스를 제거합니다.

HTTP 상태 코드

HTTP 상태 코드는 서버가 HTTP 응답의 일부로 반환하는 표준화된 숫자 코드입니다. 클라이언트 요청의 일반적인 결과를 나타내며 요청이 성공적이었는지, 추가 작업이 필요했는지 또는 오류가 발생했는지 여부를 파악하는 데 도움이 됩니다.

코드는 첫 번째 숫자에 따라 5개의 주요 범주로 그룹화됩니다.

- 1xx 코드는 정보를 제공하는 코드이며 요청이 수신되었고 처리가 계속되고 있음을 나타냅니다.
- 2xx 코드는 성공적인 요청을 나타내며, 서버가 성공적으로 요청을 수신, 파악 및 처리했음을 의미합니다.
- 3xx 코드는 리디렉션에 사용되며 요청을 완료하기 위해 추가 작업이 필요할 수 있음을 나타냅니다.
- 4xx 코드는 클라이언트 오류를 나타내며, 클라이언트가 전송한 요청에 문제가 있음을 의미합니다.

니다.

- 5xx 코드는 서버 오류를 나타냅니다. 즉 서버에서 요청을 처리하는 동안 문제가 발생했습니다

REST 개요

REST(Representational State Transfer)는 네트워크 애플리케이션을 설계하기 위한 아키텍처 스타일입니다. RESTful API는 전송 메커니즘으로 HTTP를 사용하며, 가벼운 특성으로 인해 데이터 교환에 JSON을 일반적으로 사용합니다.

클라이언트가 REST API와 상호 작용하면 서버는 리소스의 현재 상태를 표시합니다. 이 표현은 응답의 일부로서 클라이언트에 전송되며 다른 형식으로 표현될 수 있다.

REST는 프로토콜이 아닙니다. 대신, 분산 시스템 및 API를 구축하기 위한 일련의 원칙과 제약 조건을 정의하는 계층적 스타일입니다. 최신 API에서 JSON은 가볍고 쉽게 처리할 수 있으므로 가장 일반적인 형식 중 하나입니다.

일반적인 REST 상호 작용은 클라이언트 기반 모델을 준수합니다(앞서 문서에서 언급됨). 클라이언트는 GET, POST, PUT, PATCH 또는 DELETE와 같은 메서드를 사용하여 HTTP 요청을 보냅니다. 서버 프로세스는 요청된 정보 또는 작업의 결과를 포함하는 HTTP 응답을 반환하고, 반환합니다.

RESTful API는 일반적으로 리소스에 대한 작업을 수행하기 위한 통신 메커니즘으로 HTTP를 사용합니다. 이 모델에서는 클라이언트가 리소스/끝점을 식별한 다음 HTTP 메서드를 사용하여 해당 리소스에 대해 수행할 작업을 나타냅니다. REST API는 몇 가지 기본 요소를 사용하여 설명할 수 있습니다. 먼저 URI는 클라이언트가 액세스하려는 리소스 또는 엔드포인트를 식별하는 데 사용됩니다. URI는 API 내에서 해당 리소스의 위치 또는 식별자를 제공합니다. 둘째, 클라이언트는 GET, POST, PUT, PATCH 및 DELETE와 같은 표준 HTTP 메서드를 사용하여 요청 작업을 정의합니다. 예를 들어 GET을 사용하여 리소스를 검색하고, POST를 사용하여 리소스를 만들고, DELETE를 사용하여 리소스를 완전히 제거할 수 있습니다. 마지막으로 리소스는 해당 미디어 유형에 의해 정의된 표현을 통해 교환됩니다. 이 표현은 클라이언트와 서버 사이에서 리소스 데이터가 전송되는 형식입니다 (예: JSON 또는 XML).

Catalyst Center 및 REST API

CaTC는 자동화, 컨피그레이션, 정책 관리, 모니터링 및 분석을 중앙 집중화하여 간단하게 캠퍼스 네트워크 운영을 지원합니다. 네트워크 관리자가 장치를 더 효율적으로 배포 및 관리하고, 일관된 정책을 유지하고, 네트워크 동작을 모니터링하고, 전반적인 운영 가시성을 향상시킬 수 있도록 지원합니다. 이러한 기능은 GUI(그래픽 사용자 인터페이스)는 물론 API 및 외부 통합을 통해서도 액세스할 수 있습니다.

CaTC 플랫폼은 맞춤형 통합을 구축하고 네트워크 운영을 자동화하는 데 사용할 수 있는 API도 제공합니다. CaTC API를 기존 Cisco 솔루션 또는 타사 애플리케이션과 결합함으로써 조직은 플랫폼 기능을 확장하고 특정 운영 또는 비즈니스 요구 사항에 맞는 맞춤형 워크플로를 만들 수 있습니다.

기본 개념을 다룬 후 이 문서에서는 Cisco Catalyst Center 내의 노스바운드(Northbound) API(Intent API라고도 함)를 중점적으로 살펴봅니다. 이러한 API는 정책 기반의 비즈니스 목적 추상화를 통해 특정 Catalyst Center 기능을 공개하므로 관리자와 개발자는 이를 달성하는 데 필요한 개별 구성 단계 대신 원하는 결과에 집중할 수 있습니다.

Catalyst Center Intent API는 JSON 데이터 형식을 사용하여 플랫폼과 상호 작용하는 RESTful API입니다. 네트워크 정보를 검색하고 운영 데이터를 검색하며 네트워크 전체에서 지원되는 제어 및 자동화 작업을 수행하는 데 사용할 수 있습니다.

Bruno를 사용하여 인증 토큰 검색

API 호출을 수행하기 전에 토큰을 받으려면 인증해야 합니다.

1. API 엔드포인트 식별

1. Catalyst Center에 로그인합니다.
2. Menu(메뉴) > Platform(플랫폼) > Developer Toolkit(개발자 툴킷)으로 이동합니다.
3. 왼쪽 메뉴에서 Authentication(인증) > Authentication API(인증 API)를 선택합니다.

The screenshot shows the Cisco Catalyst Center web interface. The top navigation bar includes the Cisco logo, 'Catalyst Center', and a breadcrumb trail 'Platform / Developer Toolkit'. Below this, there are tabs for 'APIs', 'Integration Flows', and 'Event Notifications'. The left sidebar contains a search bar and a list of categories: 'Appliance', 'Authentication' (highlighted with a red box), 'No Subdomain', 'Cisco DNA Center System', 'Connectivity', 'Ecosystem Integrations', 'Event Management', 'Integrations', 'Know Your Network', 'Operational Tasks', 'Policy', 'Site Management', 'System', and 'System Settings'. The main content area is titled 'Authentication' and includes a search bar. Below the title, there is a description: 'Authentication APIs provide an authorized token for accessing any REST API. Add the request header 'x-auth-token' with the generated authorized token to get a successful API response.' A table lists three APIs:

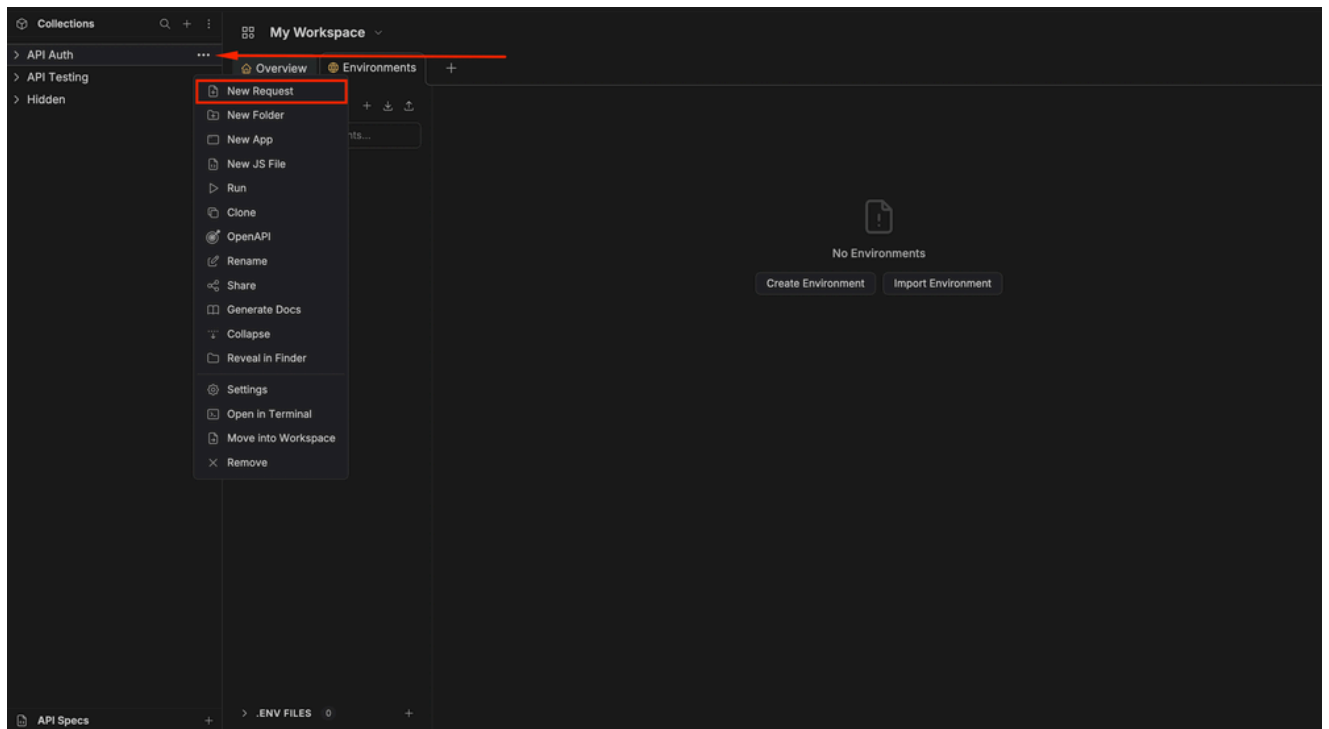
Method	Name	Description	URL	Actions
POST	importCertificate	This API enables a user to import a PEM certificate and its key for the controller and/or disaster recovery.	/dna/intent/api/v1/certificate	...
POST	importCertificateP12	This API enables a user to import a PKCS12 certificate bundle for the controller and/or disaster recovery.	/dna/intent/api/v1/certificate-p12	...
POST	Authentication API	API to obtain an access token, which remains valid for 1 hour. The token obtained using this API is required to be set as value to the X-Auth-Token HTTP...	/dna/system/api/v1/auth/token	...

The 'Authentication API' row is highlighted with a red box. Below the table, the text 'Cisco DNA Center System' is displayed.

4. URL을 복사합니다(일반적으로 POST /dna/system/api/v1/auth/token).

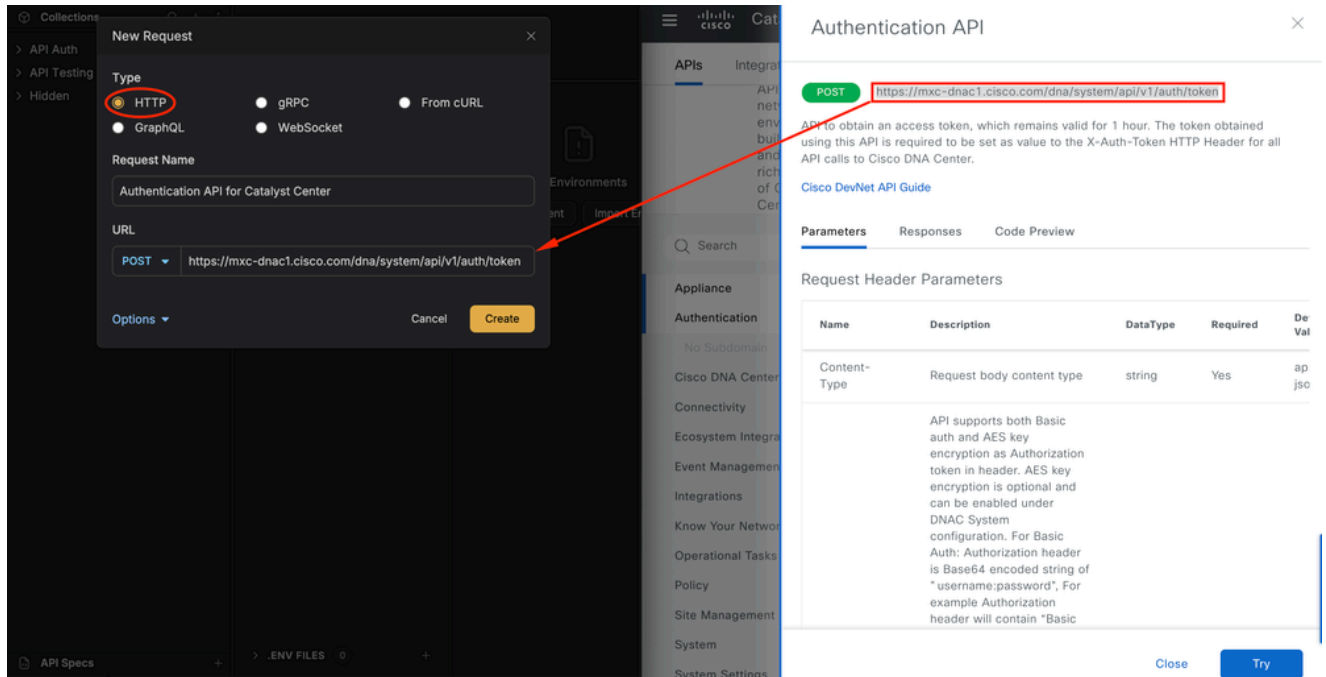
2. 인증을 위해 Bruno 구성

1. Bruno에서 새 요청을 생성합니다.
2. Type을 HTTP로 설정하고 Method를 POST로 설정합니다.



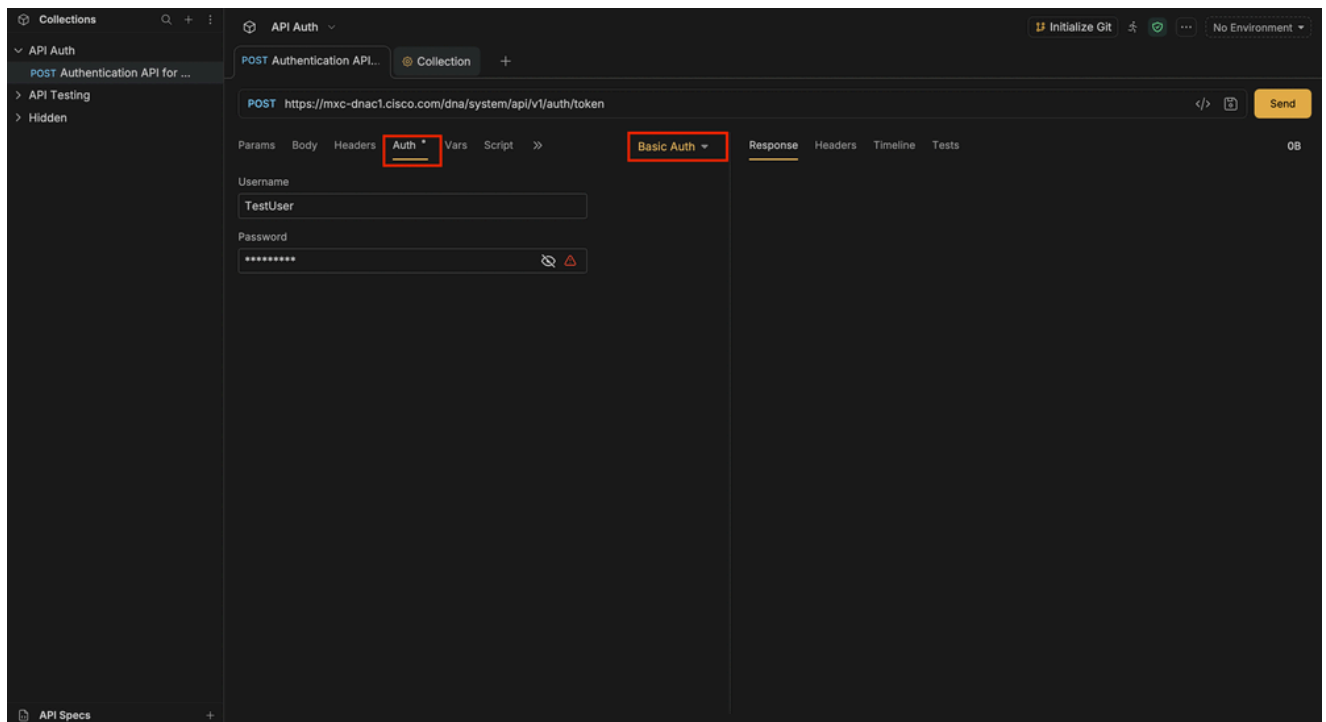
Bruno에서 새 요청 생성

3. 인증 API의 URL을 입력합니다.



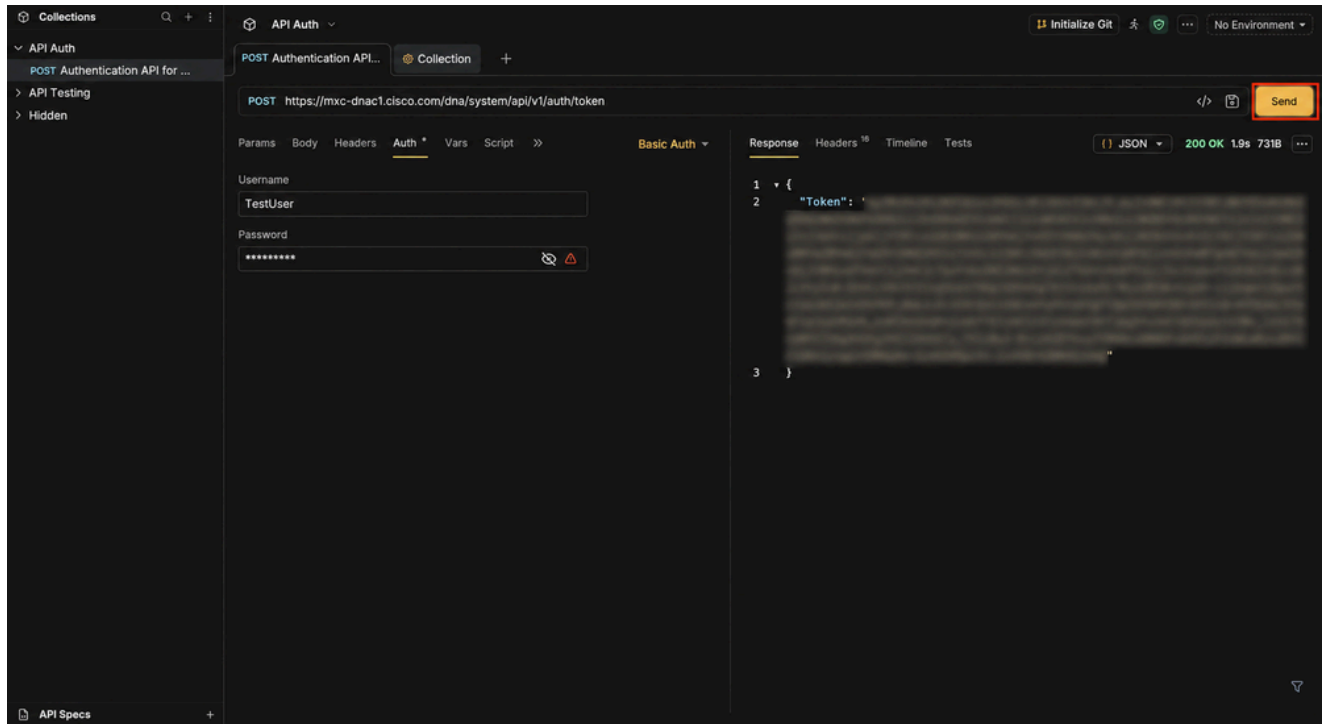
Bruno에서 새 요청 생성

4. Auth(인증) 탭으로 이동하고 Basic Auth(기본 인증)를 선택합니다.



인증 API에 대한 기본 인증

5. Catalyst Center SUPER ADMIN 자격 증명을 입력합니다.
6. Send(보내기)를 클릭합니다.



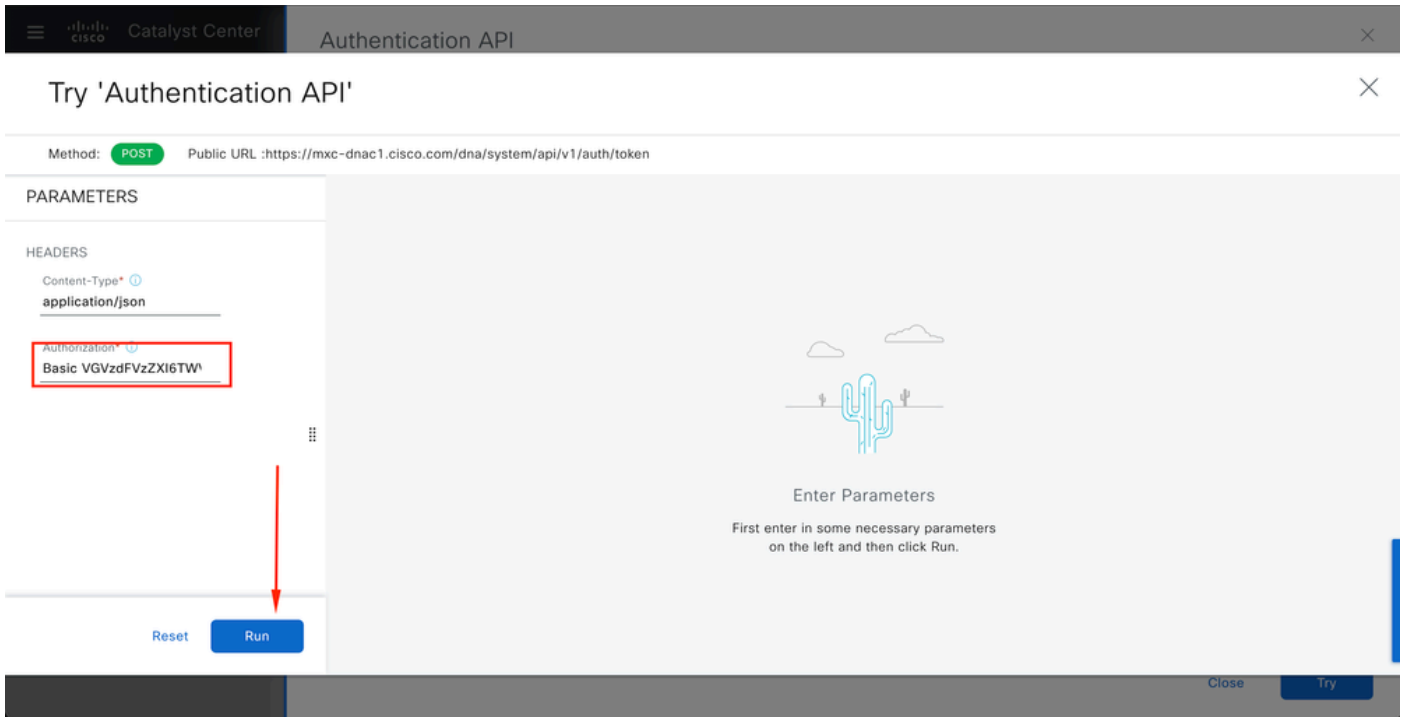
인증 API 실행

서버는 200 OK 상태 및 Token을 포함하는 JSON 페이로드를 반환합니다.

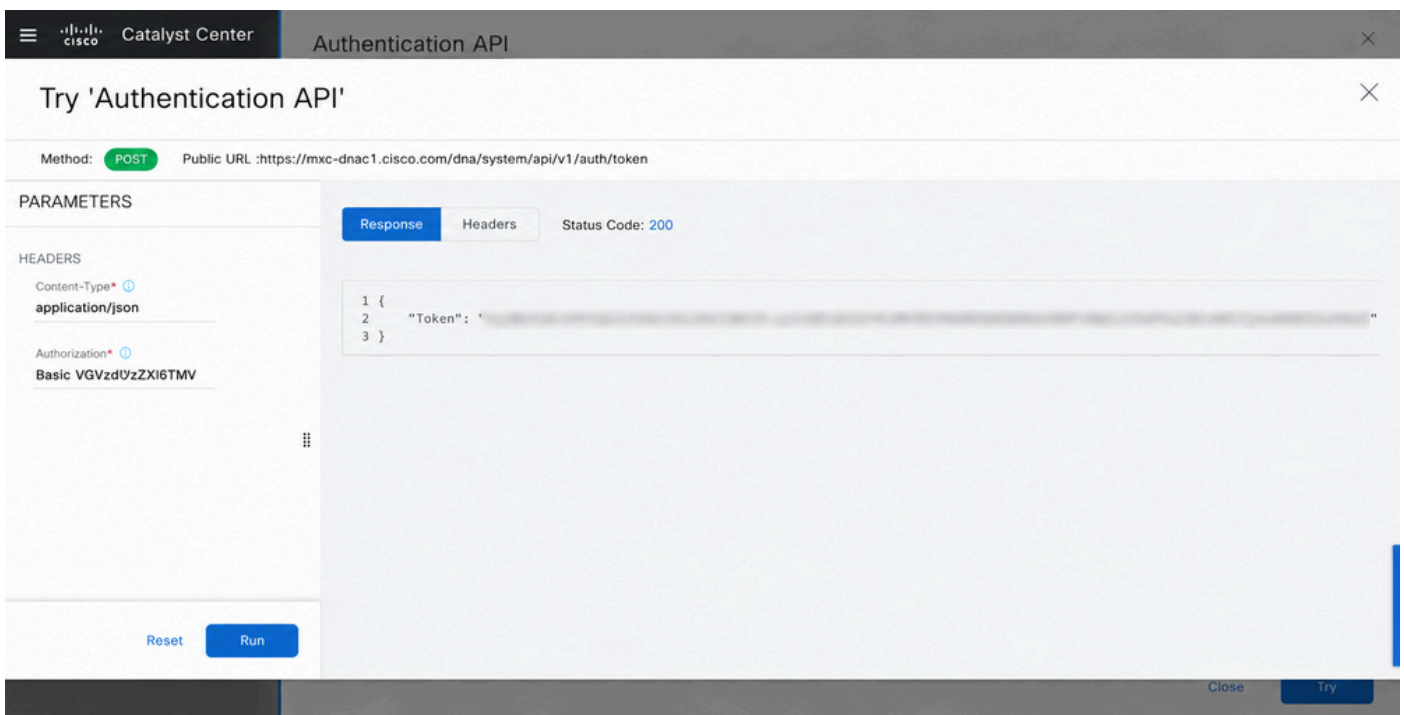
Alternative (Terminal): You can also generate the token by base64-encoding your `username:password` and

```
[~] % echo -n 'TestUser:VGvzdFVzZXI6VGvzdDEyMyEhIQ==' | base64
```

사용자 이름 및 비밀번호에 대한 base64 인코딩 가져오기



인코딩된 base64 사용자 이름 및 비밀번호로 기본 인증 입력



Catalyst Center GUI에서 인증 토큰을 가져옵니다.

이때 Bruno 응답 또는 Catalyst Center Developer Toolkit에서 수행한 응답에 대한 인증 토큰이 있습니다.

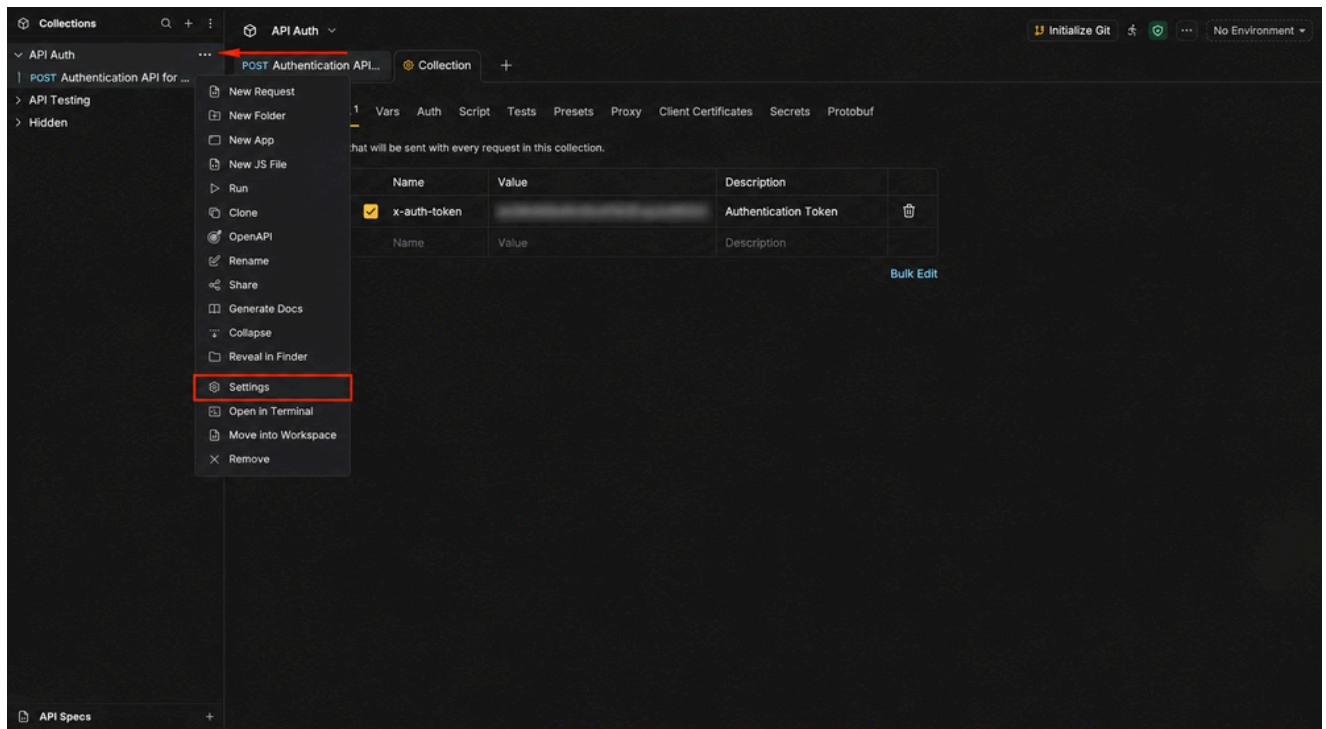
이제 토큰을 Bruno의 컬렉션 설정에 Header로 추가합니다. 이렇게 하면 컬렉션 내에서 생성된 다른 모든 요청이 토큰을 사용할 수 있으므로 각 요청에 대한 헤더로 토큰을 수동으로 입력할 필요가 없

습니다.

3. Bruno에서 글로벌 헤더 설정

모든 요청에 대해 재인증을 방지하려면 토큰을 컬렉션 수준 헤더로 설정합니다.

1. 컬렉션을 마우스 오른쪽 단추로 클릭하고 설정을 선택합니다.

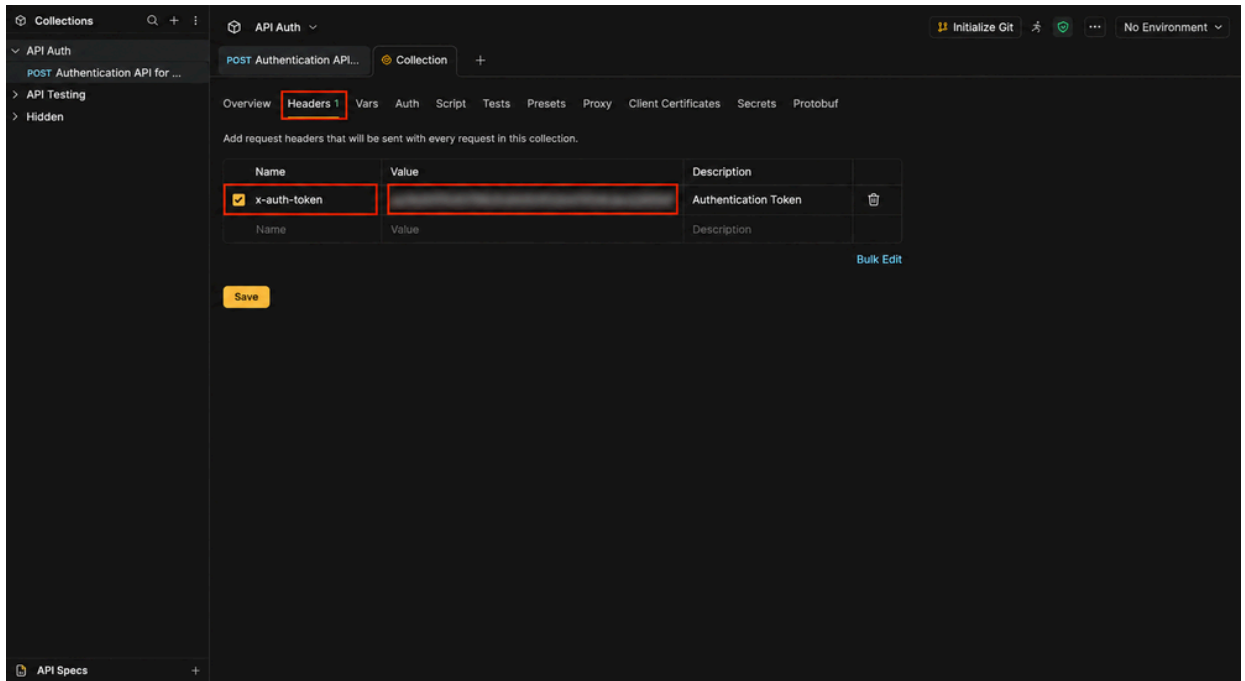


컬렉션 설정

2. Headers(헤더) 탭으로 이동합니다.

3. 새 헤더 추가:

- 이름: x 인증 토큰
- 가치: [여기에 토큰 붙여넣기]



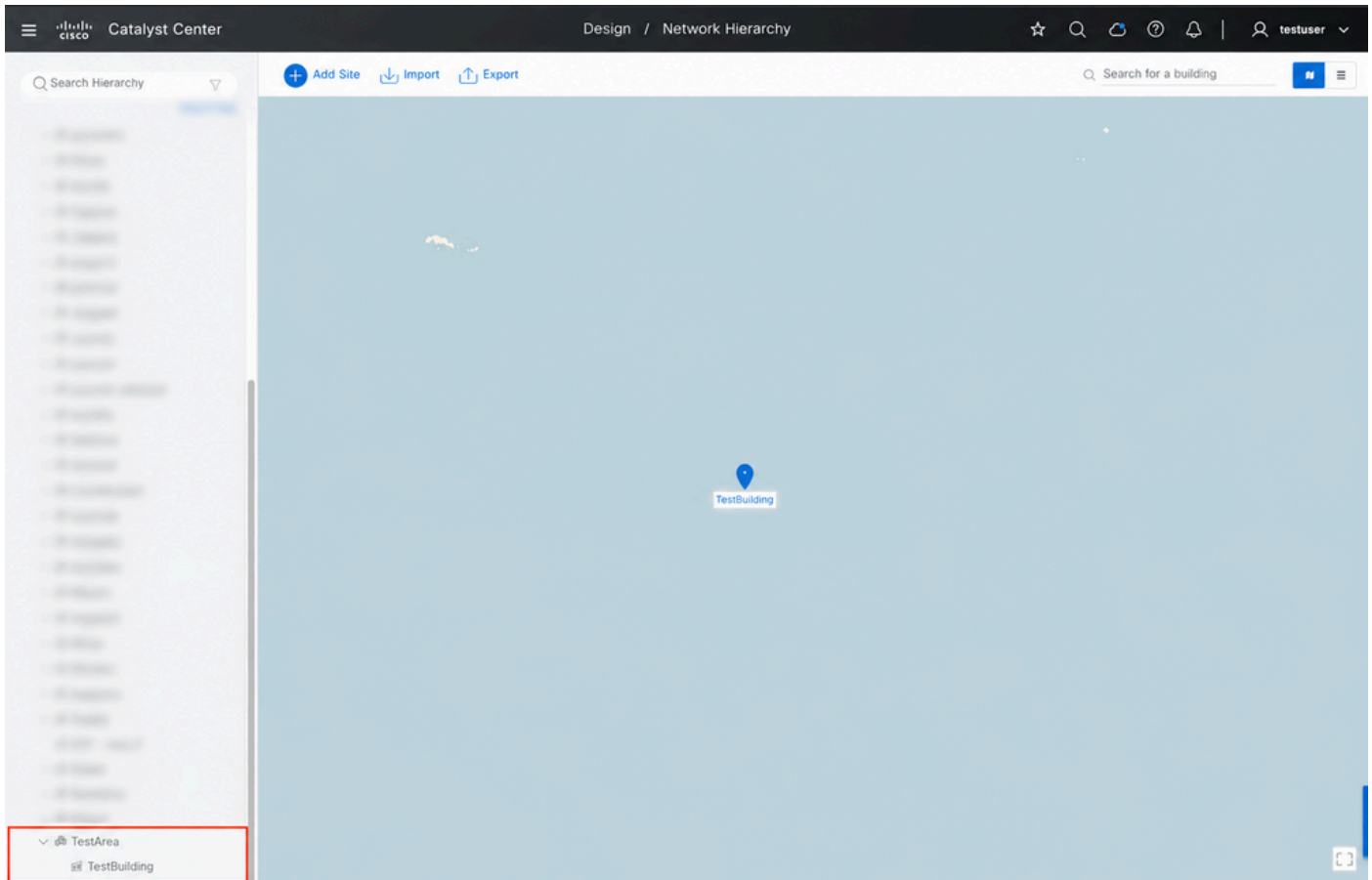
Bruno에서 API 컬렉션에 대한 전역 헤더를 입력하는 중입니다.

4. 저장을 클릭합니다.

이 시점에서 다른 Catalyst Center API 요청을 수행하기 위해 인증 토큰을 생성했습니다.

예 1: 사이트 층 생성

이 예에서는 네트워크 계층 구조(Global > TestArea > TestBuilding) 내에 층을 만드는 방법을 보여줍니다.

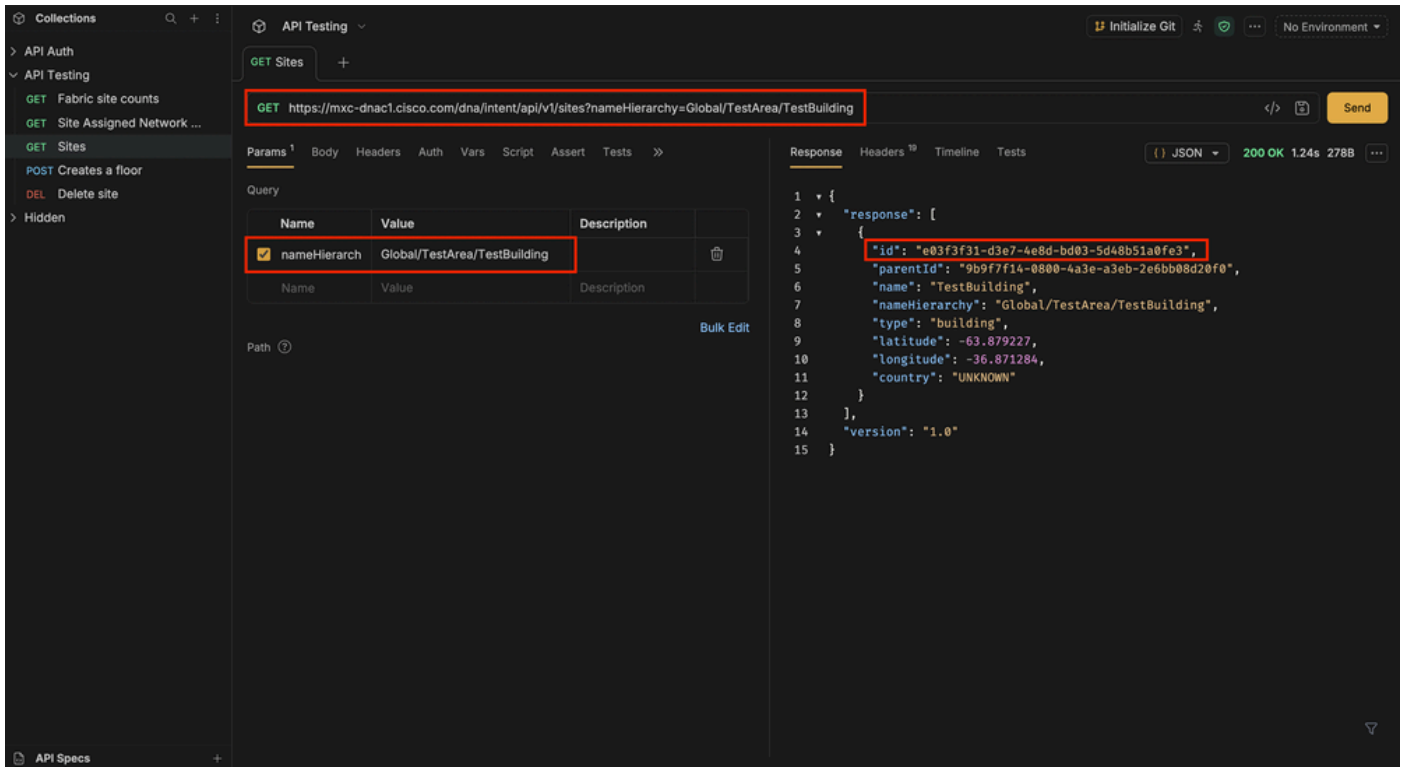


네트워크 계층

1단계: 상위 ID 가져오기

GET /dna/intent/api/v1/sites API를 사용하여 TestBuilding의 ID를 찾습니다.

- 쿼리 매개 변수: nameHierarchy=전역/테스트 영역/테스트 빌딩
- 응답에서 ID를 복사합니다. 이것은 귀하의 parentId입니다.



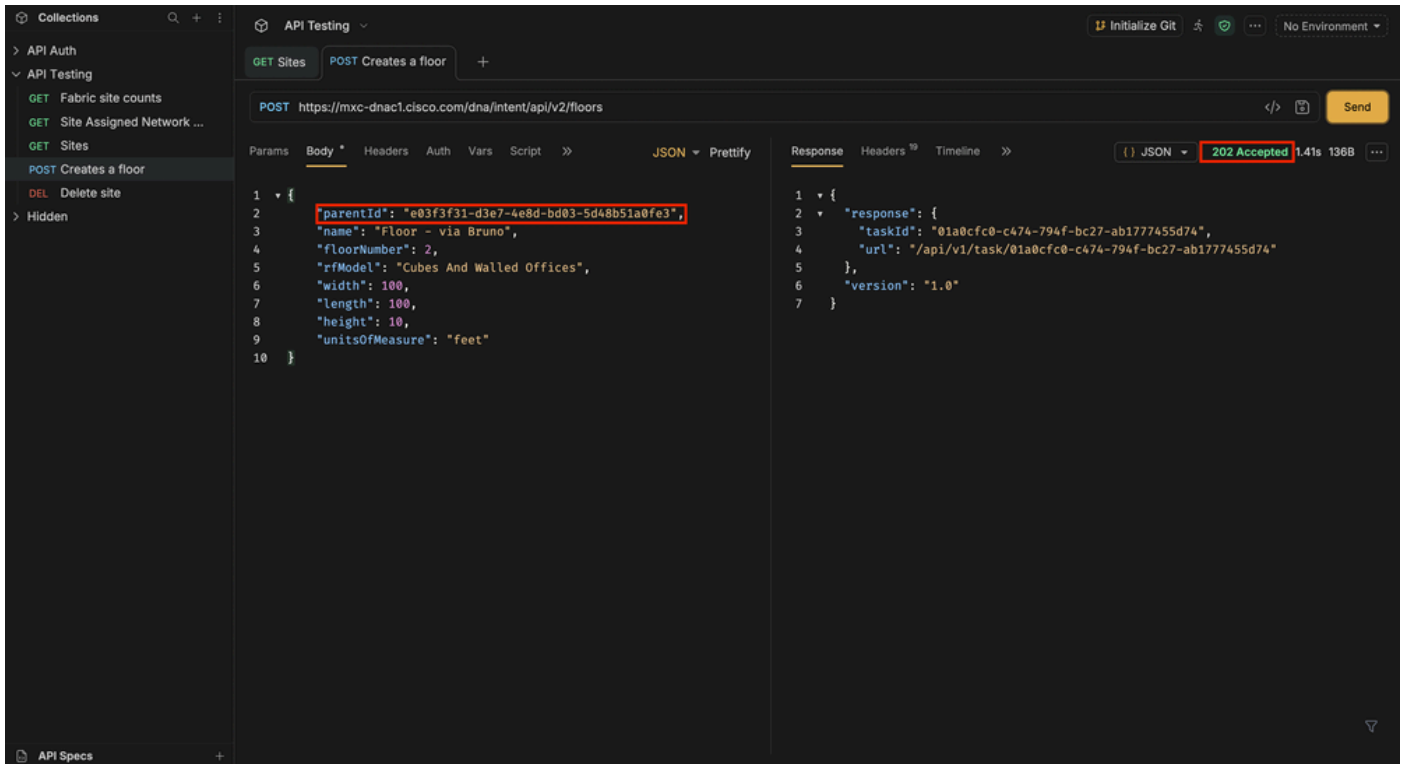
/dna/intent/api/v1/sites?nameHierarchy=Global/TestArea/TestBuilding에 대한 GET 요청

2단계: 층 생성

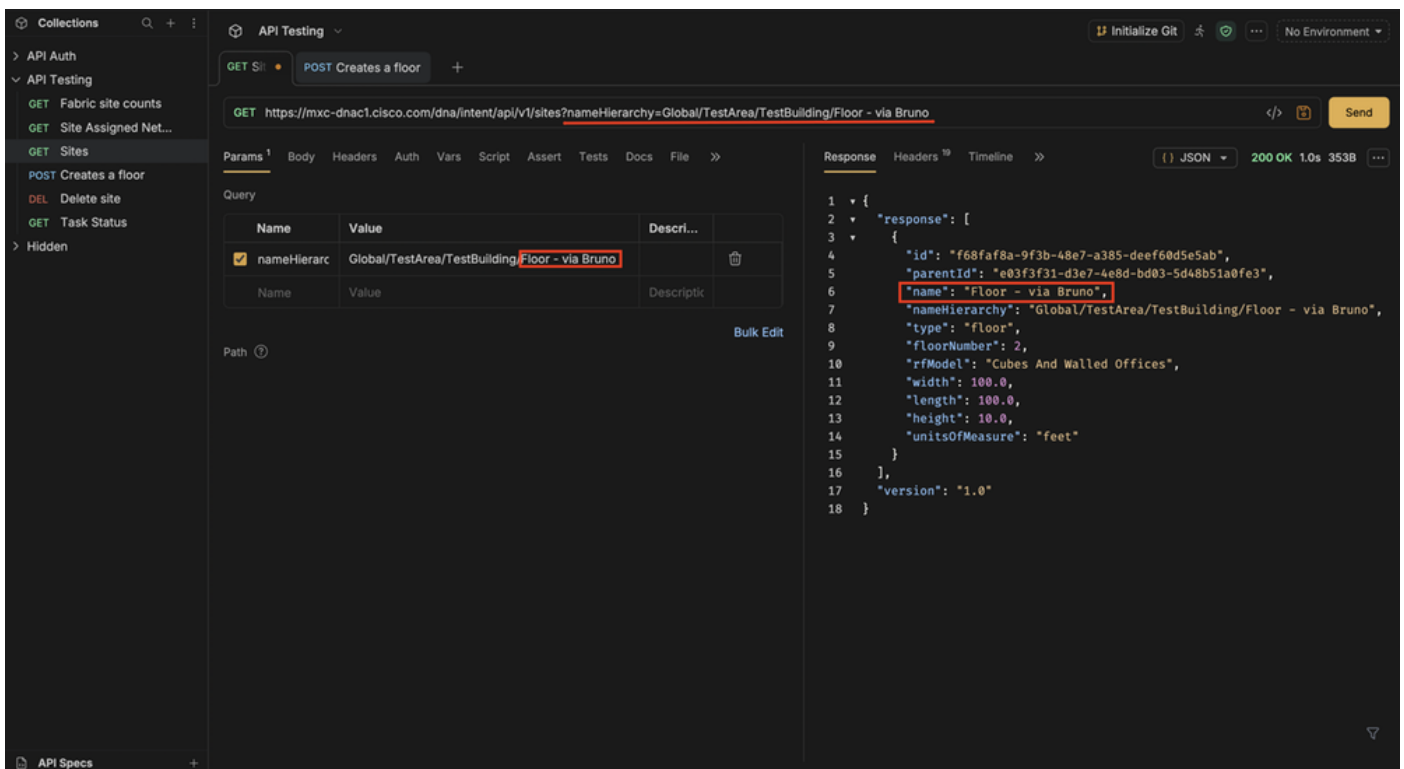
1. /dna/intent/api/v2/floor에 대한 POST 요청을 생성합니다.
2. JSON(본문) 탭에서 필요한 매개변수를 입력합니다.

```
{
  "name": "Floor 1",
  "parentId": "your-building-id-here",
  "floorNumber": 1,
  "rfModel": "Cubic-Office",
  "width": 100,
  "length": 100,
  "height": 10,
  "unitsOfMeasure": "feet"
}
```

3. Send(보내기)를 클릭합니다. 응답이 성공하면 바닥 생성이 확인됩니다.



POST 요청 - 층을 생성합니다.



POST 요청을 통해 층 생성을 검증하는 API를 가져옵니다.

마지막으로, 플로어가 성공적으로 생성되었는지 검증하기 위해 이전 API를 실행하고 위에 표시된 전체 계층 경로와 일치하도록 쿼리를 수정할 수 있습니다.

예 2: 인터페이스 정보 검색

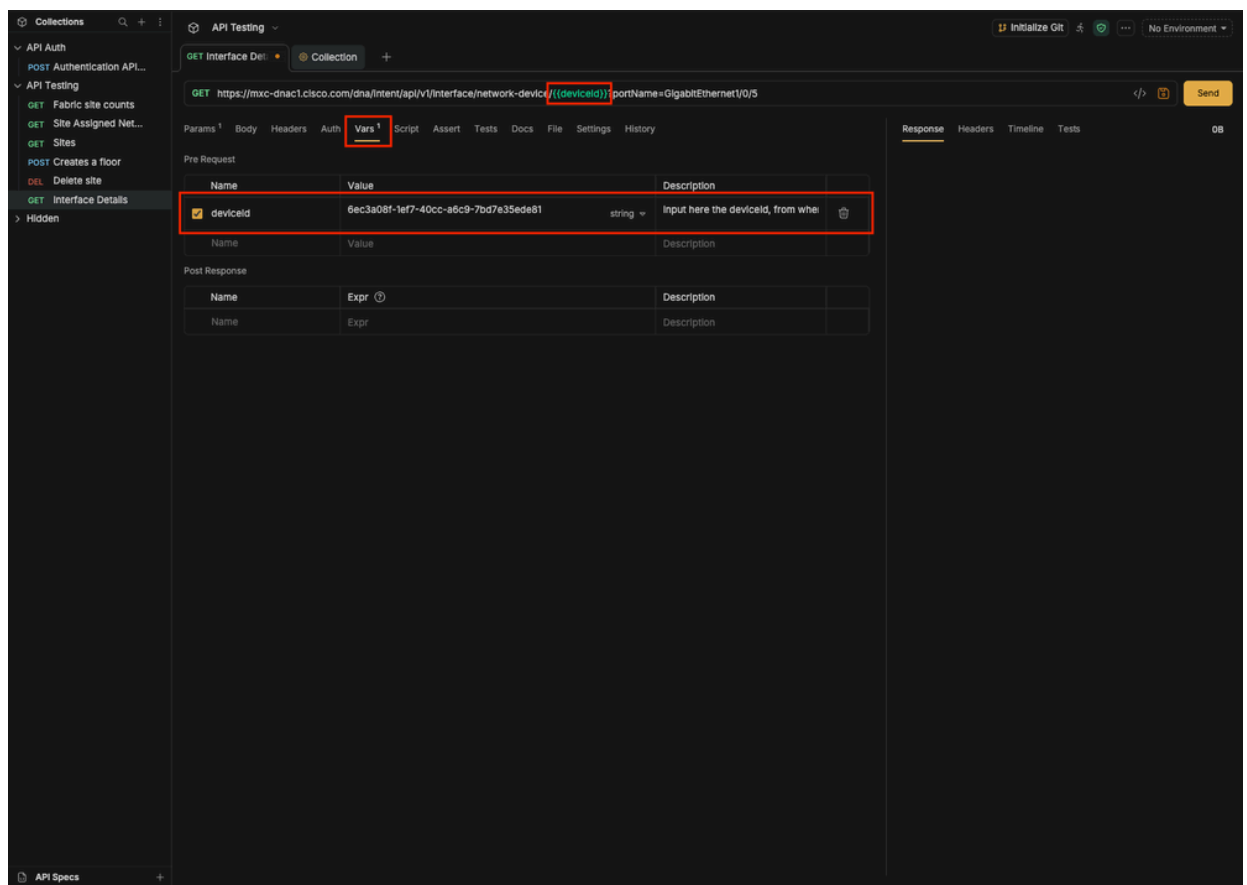
이 예에서는 네트워크 디바이스에서 특정 인터페이스의 상태를 가져오는 방법을 보여 줍니다.

1단계: 디바이스 ID 식별

Catalyst Center GUI에서 Inventory(인벤토리)로 이동하여 디바이스를 클릭하고 브라우저의 URL을 확인합니다. 마지막 세그먼트는 Device ID입니다.

2단계: Bruno에서 매개 변수가 있는 요청 만들기

1. `/dna/intent/api/v1/interface/network-device/{{deviceid}}`에 대한 GET 요청을 만듭니다.
2. 요청의 Vars 탭에서 deviceid라는 변수를 추가하고 ID를 붙여넣습니다.
3. 특정 포트에 대해 필터링하려면 쿼리 매개변수를 추가합니다.
 - 키: 이름
 - 가치: GigabitEthernet1/0/5



Bruno에서 변수 만들기

4. Send(보내기)를 클릭합니다.

The screenshot shows the API Testing interface with a GET request to `https://mxc-dnac1.cisco.com/dna/intent/api/v1/interface/network-device/{{deviceId}}?portName=GigabitEthernet1/0/5`. The request parameters are defined in a table:

Name	Value	Description
portName	GigabitEthernet1/0/5	

The response is a JSON object showing details for the specified interface:

```
1 {
2   "response": [
3     {
4       "addresses": [],
5       "adminStatus": "UP",
6       "className": null,
7       "deviceId": "6ce3a08f-1ef7-40cc-a6c9-7bd7e358de81",
8       "duplex": "FullDuplex",
9       "ifIndex": "12",
10      "interfaceType": "Physical",
11      "ipv4Address": null,
12      "ipv4Mask": null,
13      "isisSupport": "false",
14      "lastIncomingPacketTime": 1786685566000,
15      "lastOutgoingPacketTime": 1790265622000,
16      "lastUpdated": null,
17      "macAddress": "000000000000",
18      "mappedPhysicalInterfaceId": null,
19      "mappedPhysicalInterfaceName": null,
20      "mediaType": null,
21      "mtu": "1500",
22      "nativeVlanId": "1",
23      "ospfSupport": "false",
24      "pid": "20",
25      "portMode": "access",
26      "portName": "GigabitEthernet1/0/5",
27      "portType": "Ethernet Port",
28      "serialNo": "000000000000",
29      "series": "000000000000",
30      "speed": "1000000",
31      "status": "up",
32      "vlanId": "1",
33      "voiceVlan": "",
34      "description": "",
35      "name": null,
36      "instanceBuid": "18208b55-383c-4970-ab8b-db942252974d",
37      "instanceTenantId": "66571d0b1efef061f865c6d5",
38      "id": "18208b55-383c-4970-ab8b-db942252974d"
39    }
40  ],
41   "version": "1.0"
42 }
```

쿼리를 적용하는 GET 요청

결론

Catalyst Center 노스바운드 API는 네트워크 관리자에게 강력한 유연성을 제공합니다. GUI 기반 관리에서 API 기반 워크플로로 전환함으로써 반복적인 작업을 자동화하고, 구성 일관성을 보장하며, 더 광범위한 IT 에코시스템에 네트워크 인텔리전스를 통합할 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.