

# Catalyst Center APIの理解と使用

## 内容

---

[概要](#)

[背景説明](#)

[前提条件](#)

[要件](#)

[使用するコンポーネント](#)

[背景説明](#)

[HTTPの概要](#)

[HTTP要求](#)

[HTTP応答](#)

[HTTPメソッド \(動詞\)](#)

[HTTPステータスコード](#)

[RESTの概要](#)

[Catalyst CenterおよびREST API](#)

[Brunoを使用した認証トークンの取得](#)

[1. APIエンドポイントの特定](#)

[2. Brunoの認証の設定](#)

[3. Brunoでのグローバルヘッダーの設定](#)

[例1: 外構の床を作成する](#)

[ステップ1: 親IDの取得](#)

[手順2: 床を作成する](#)

[例2: インターフェイス情報の取得](#)

[ステップ1: デバイスIDを特定する](#)

[ステップ2: Brunoでパラメータ化されたリクエストを作成する](#)

[結論](#)

---

## 概要

このドキュメントでは、APIクライアントとプラットフォームの組み込み開発者ツールを使用して、CaTC Northbound (Intent) APIにアクセスし、利用する方法について説明します。

## 背景説明

HTTP/RESTの基本的な概念について説明し、自動化とテストの実例を示します。

# 前提条件

## 要件

次の項目に関する知識が推奨されます。

- HTTP/1.1プロトコルとステータスコード。
- REST建築スタイル。
- Cisco Catalyst Center(CaTC)の基本知識
- Bruno (またはPostmanのような同様のAPIクライアント) の基本的な知識。

## 使用するコンポーネント

このドキュメントの情報は、次のハードウェアとソフトウェアのバージョンに基づいています。

- Cisco Catalyst Centerバージョン2.3.7.xおよび3.x
- Brunoバージョン4.1.0

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されました。このドキュメントで使用するすべてのデバイスは、初期（デフォルト）設定の状態から起動しています。本稼働中のネットワークでは、各コマンドによって起こる可能性がある影響を十分確認してください。

## 背景説明

Catalyst CenterノースバウンドAPI（インテントAPI）を使用すると、グラフィカルユーザーインターフェイスの外部でコントローラと対話できます。これにより、カスタマイズされたワークフローの自動化、サードパーティ製ツールとの統合、および規模に応じた効率的なネットワーク管理が可能になります。

## HTTPの概要

ハイパーテキスト転送プロトコル(HTTP)は、Webのデータ通信の基盤です。これは、クライアントが要求を送信し、サーバが応答を返すクライアント/サーバモデルに従っています。

## HTTP要求

HTTP要求は、次の4つの部分で構成されます。

1. 要求ライン：メソッド ( GETおよびPOSTを含む )、URI、およびプロトコルバージョン ( GET /api/v1/nodes HTTP/1.1など ) を指定します。
2. ヘッダー：認証クレデンシャルやコンテンツタイプなどのメタデータを提供します。
3. 空白行：ヘッダーを本文から分離します。
4. メッセージ本文 ( オプション ) :サーバに送信されるデータが含まれます ( POST/PUT/PATCHで共通 )。

## HTTP応答

HTTP応答は、要求構造を反映しています。

1. ステータスライン：ステータスコード ( 200 OKなど ) が含まれます。
2. ヘッダー：サーバと応答データに関する詳細。
3. メッセージ本文：要求された実際のデータ ( 通常はCatalyst Center用のJSON形式 )。

## HTTPメソッド ( 動詞 )

- GET:リソースからデータを取得します。
- POST:新しいリソースを作成します。
- PUT:既存のリソースを置き換えます。
- PATCH:リソースに部分的な変更を適用します。
- DELETE:リソースを削除します。

## HTTPステータスコード

HTTPステータスコードは、HTTP応答の一部としてサーバによって返される標準化された数値コードです。クライアント要求の一般的な結果を示し、要求が成功したか、追加のアクションが必要か、またはエラーが発生したかを把握するのに役立ちます。

コードは、最初の桁に従って5つの主要なカテゴリにグループ化されます。

- 1xxコードは情報コードであり、要求が受信されて処理が続行されていることを示します。
- 2xxコードは、要求が成功したことを示します。つまり、サーバが要求を正常に受信し、理解し、処理したことを意味します。

- リダイレクトには3xxコードが使用され、要求を完了するために追加のアクションが必要になる可能性があることを示します。
- 4xxコードはクライアントエラーを表します。これは、クライアントによって送信された要求に問題があることを意味します。
- 5xxコードはサーバエラーを表します。これは、サーバが要求を処理しようとしたときに問題が発生したことを意味します。

## RESTの概要

REST(Representational State Transfer)は、ネットワークアプリケーションを設計するためのアーキテクチャスタイルです。RESTful APIはトランスポートメカニズムとしてHTTPを使用し、軽量な性質のために通常はデータ交換にJSONを使用します。

クライアントがREST APIと対話すると、サーバはリソースの現在の状態を表す表現を返します。この表現は、応答の一部としてクライアントに転送され、別の形式で表現できます。

RESTはプロトコルではありません。代わりに、分散システムとAPIを構築するための一連の原則と制約を定義する階層型スタイルです。最新のAPIでは、軽量で処理が簡単なJSONは、最も一般的な形式の1つです。

一般的なRESTインタラクションは、クライアントベースのモデルに準拠しています ( 前述のとおり )。クライアントは、GET、POST、PUT、PATCH、DELETEなどのメソッドを使用してHTTP要求を送信します。サーバは、要求された情報または操作の結果を含むHTTP応答を処理して返します。

RESTful APIでは、リソースに対する操作を実行するための通信メカニズムとして、一般にHTTPが使用されます。このモデルでは、クライアントはリソース/エンドポイントを特定し、HTTPメソッドを使用して、そのリソースに対して実行するアクションを示します。REST APIは、いくつかの主要な要素を使用して記述できます。まず、URIは、クライアントがアクセスするリソースまたはエンドポイントを特定するために使用されます。URIは、API内でそのリソースの場所または識別子を提供します。2つ目は、クライアントがGET、POST、PUT、PATCH、およびDELETEなどの標準HTTPメソッドを使用して要求操作を定義する方法です。たとえば、GETを使用してリソースを取得し、POSTを使用してリソースを作成し、DELETEを使用してリソースを完全に削除することができます。最後に、メディアタイプによって定義された表現を介してリソースが交換されます。この表現は、JSONやXMLなど、クライアントとサーバの間でリソースデータが転送される形式です。

## Catalyst CenterおよびREST API

CaTCは、自動化、設定、ポリシー管理、モニタリング、および分析を一元化することで、キャ

ンパスネットワークの運用を簡素化します。ネットワーク管理者は、デバイスの効率的な導入と管理、一貫したポリシーの維持、ネットワーク動作の監視、全体的な運用可視性の向上を実現できます。これらの機能には、グラフィカルユーザインターフェイス(GUI)のほか、APIや外部統合を通じてアクセスできます。

CaTCプラットフォームは、カスタム統合の構築とネットワーク運用の自動化に使用できるAPIも提供します。CaTC APIを既存のシスコソリューションまたはサードパーティ製アプリケーションと組み合わせることにより、組織はプラットフォーム機能を拡張し、特定の運用またはビジネス要件に合わせてカスタマイズされたワークフローを作成できます。

基本的な概念を説明した後、このドキュメントでは、Cisco Catalyst Center内のノースバウンドAPI ( インテントAPIとも呼ばれる ) に焦点を当てます。これらのAPIは、ビジネスの目的をポリシーベースで抽象化することで、特定のCatalyst Center機能を公開します。これにより、管理者や開発者は、目的の達成に必要な個々の設定手順ではなく、目的の結果に集中することができます。

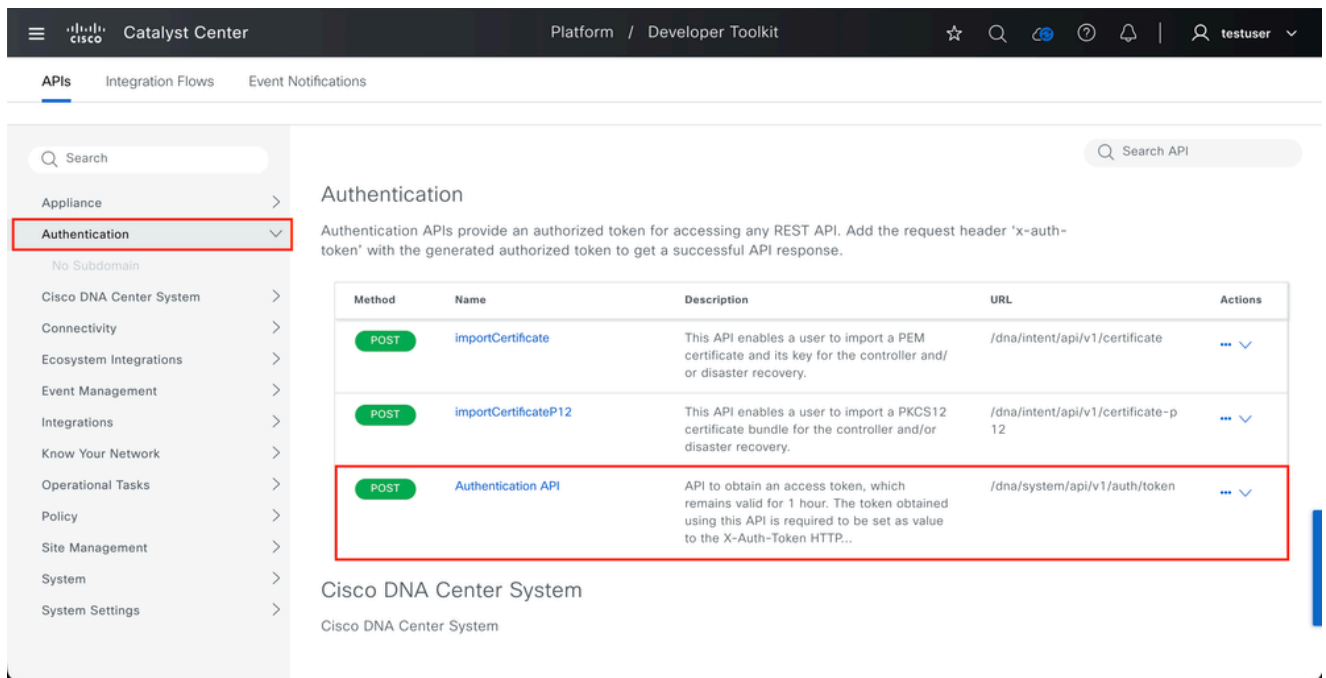
Catalyst Center Intent APIは、JSONデータ形式を使用してプラットフォームと対話するRESTful APIです。ネットワーク情報の検出、運用データの取得、サポートされている制御タスクと自動化タスクの実行にネットワーク全体で使用できます。

## Brunoを使用した認証トークンの取得

API呼び出しを行う前に、トークンを受け取るために認証を行う必要があります。

### 1. APIエンドポイントの特定

1. Catalyst Centerにログインします。
2. Menu > Platform > Developer Toolkitの順に選択します。
3. 左側のメニューで、Authentication > Authentication APIの順に選択します。

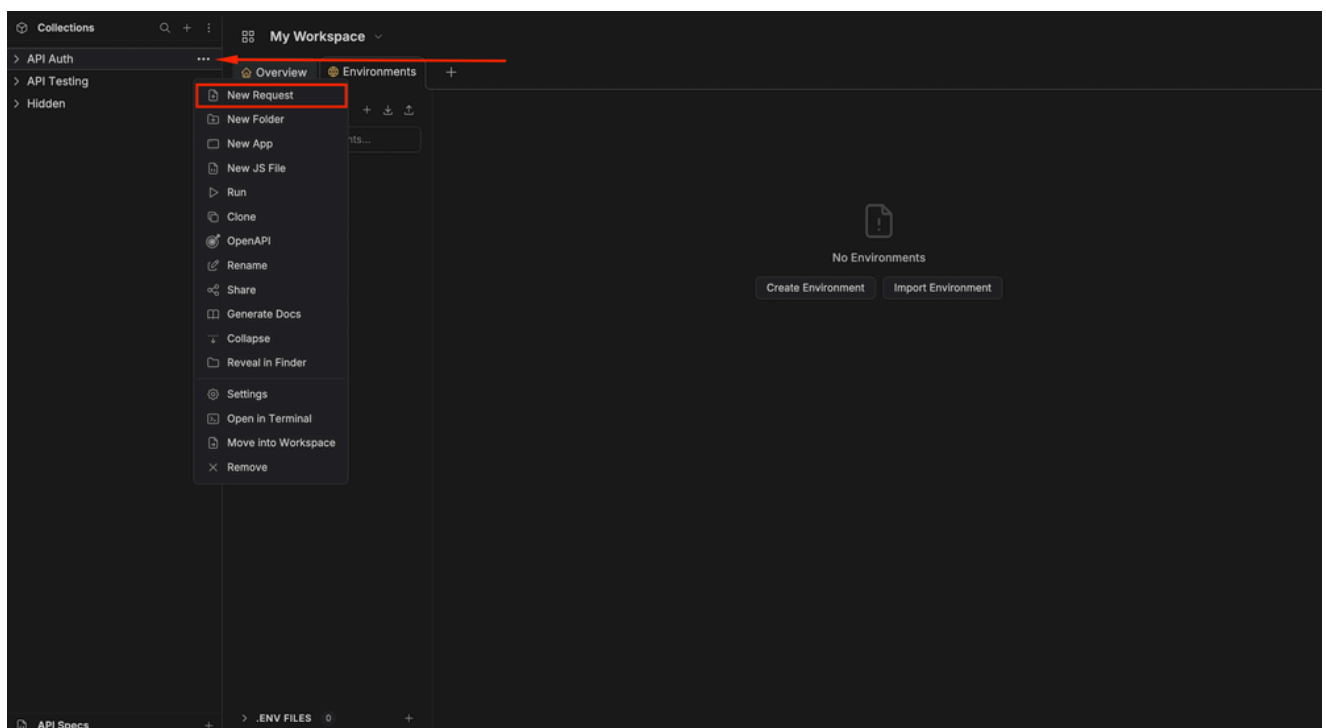


Catalyst Center開発者ツールキット

4. URL ( 通常はPOST /dna/system/api/v1/auth/token ) をコピーします。

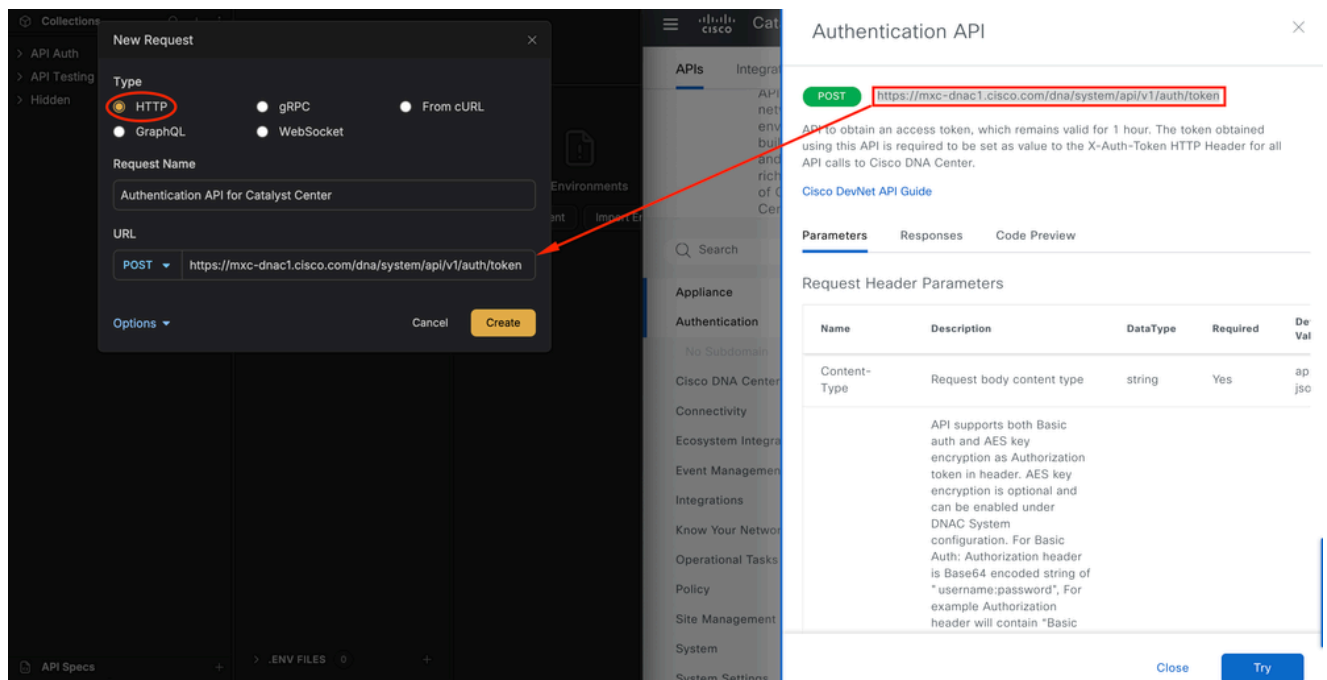
## 2. Brunoの認証の設定

1. Brunoで、新しい要求を作成します。
2. TypeをHTTPに、MethodをPOSTに設定します。



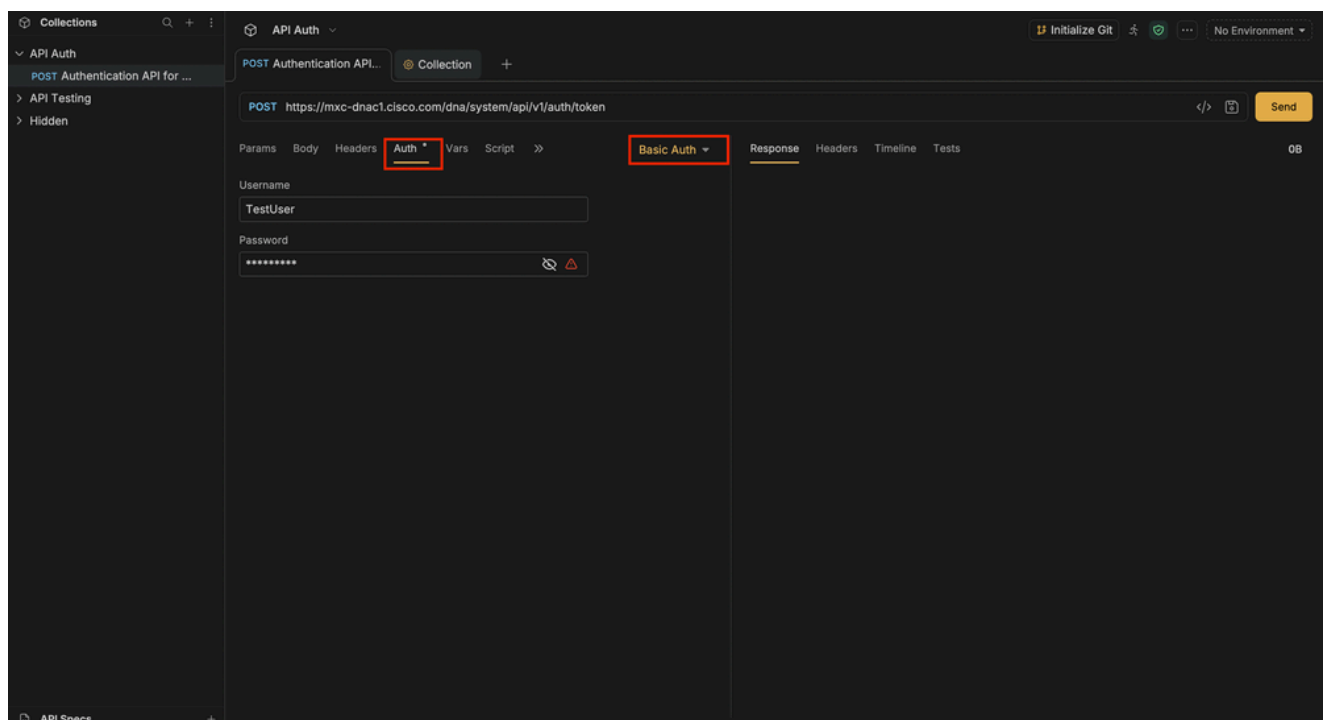
Brunoでの新しいリクエストの作成

### 3. 認証APIのURLを入力します。



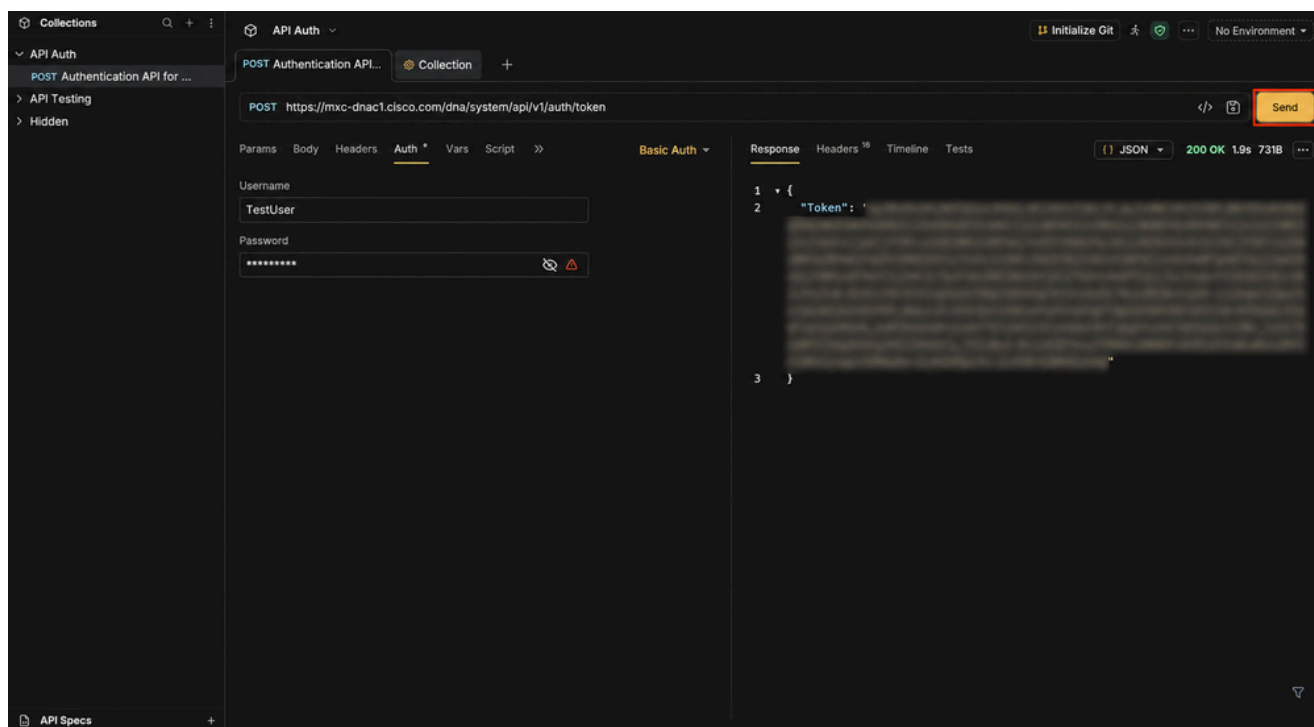
Brunoでの新しいリクエストの作成

### 4. Authタブに移動し、Basic Authを選択します。



認証APIの基本認証

5. Catalyst Center SUPER ADMINクレデンシャルを入力します。
6. [Send] をクリックします。



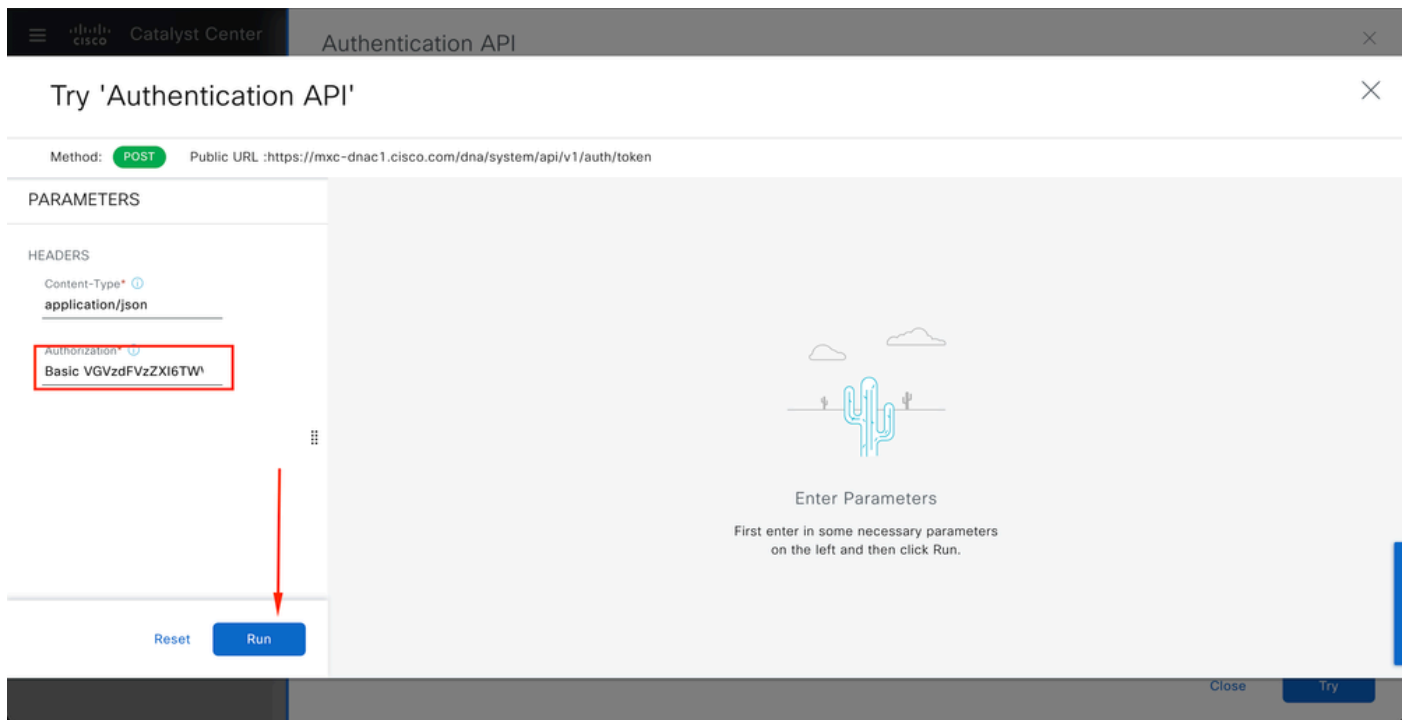
認証APIの実行

サーバは、200 OKステータスと、トークンを含むJSONペイロードを返します。

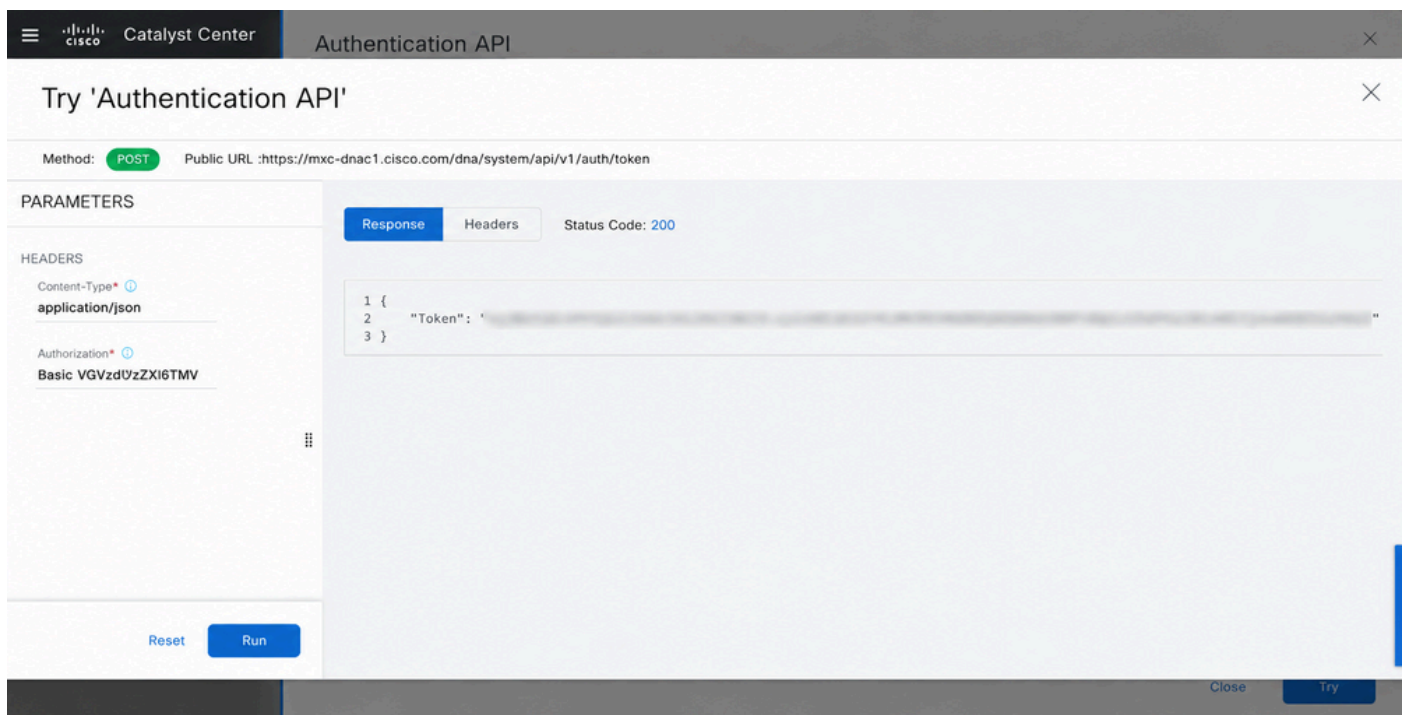
Alternative (Terminal): You can also generate the token by base64-encoding your `username:password` and

```
[redacted] ~ % echo -n 'TestUser:[redacted]' | base64  
VGZzdFVzZXI6VGZzdDEyMyEhIQ==
```

ユーザ名とパスワードのbase64エンコーディングを取得する



エンコードされたbase64ユーザ名とパスワードを使用した基本認証の入力



## Catalyst Center GUIからの認証トークンの取得

この時点で、Brunoの応答またはCatalyst Center Developer Toolkitから実行された応答のいずれかで認証トークンが取得されます。

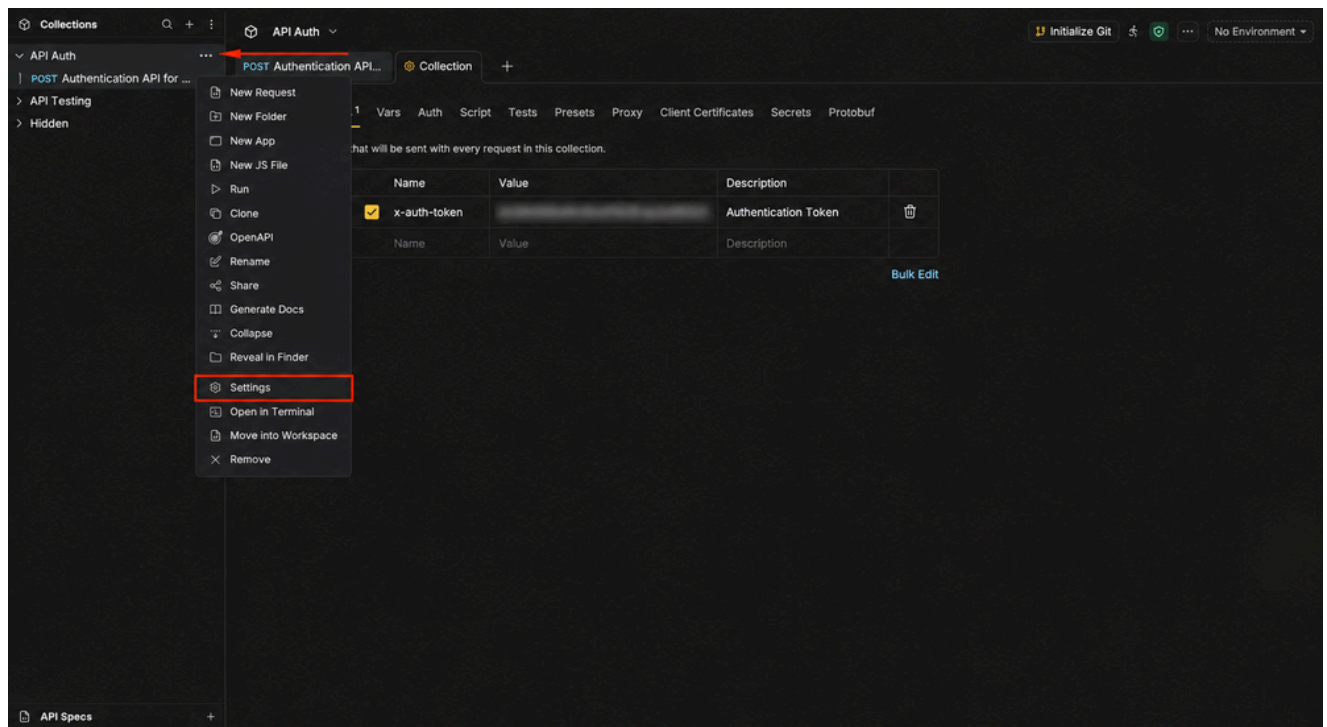
Brunoのコレクション設定でトークンをヘッダーとして追加すると、コレクション内で作成されたその他の要求でトークンが使用されやすくなるため、要求ごとにヘッダーとしてトークンを手

動で入力する必要がなくなります。

### 3. Brunoでのグローバルヘッダーの設定

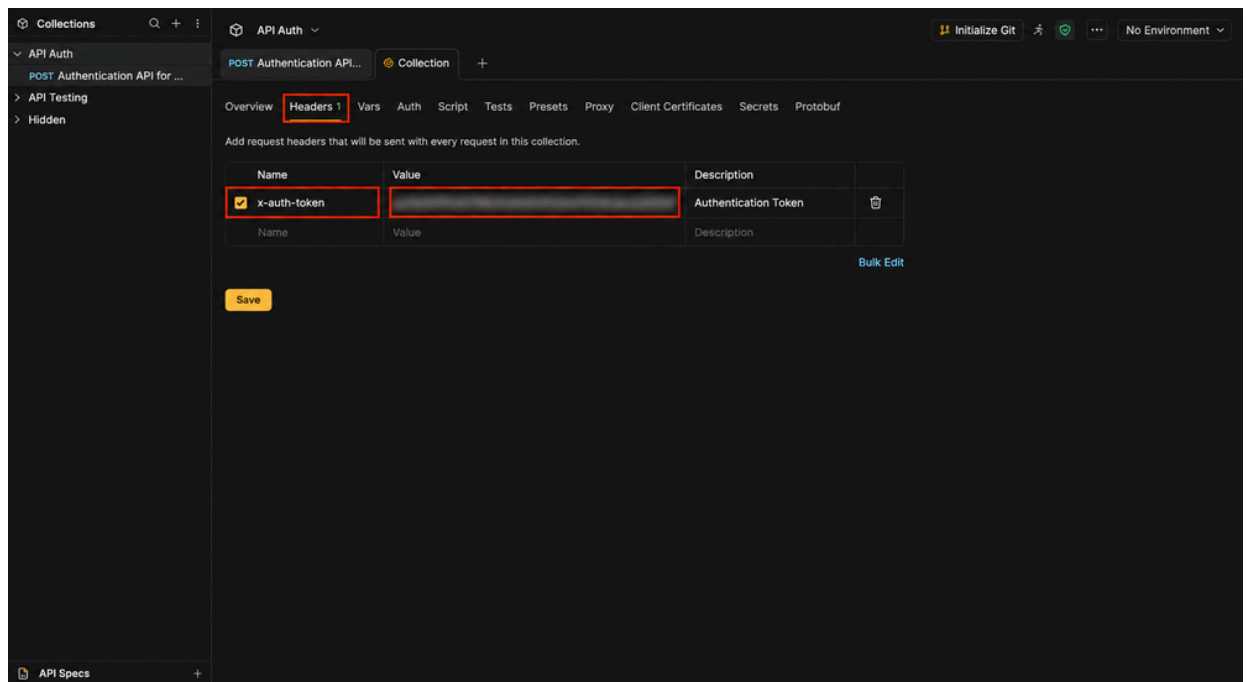
すべての要求に対する再認証を回避するには、トークンをコレクションレベルのヘッダーとして設定します。

1. コレクションを右クリックし、Settingsを選択します。



コレクション設定

2. Headersタブに移動します。
3. 新しいヘッダーを追加する：
  - 名前：x-auth-token
  - [Value]：[ここにトークンを貼り付ける]



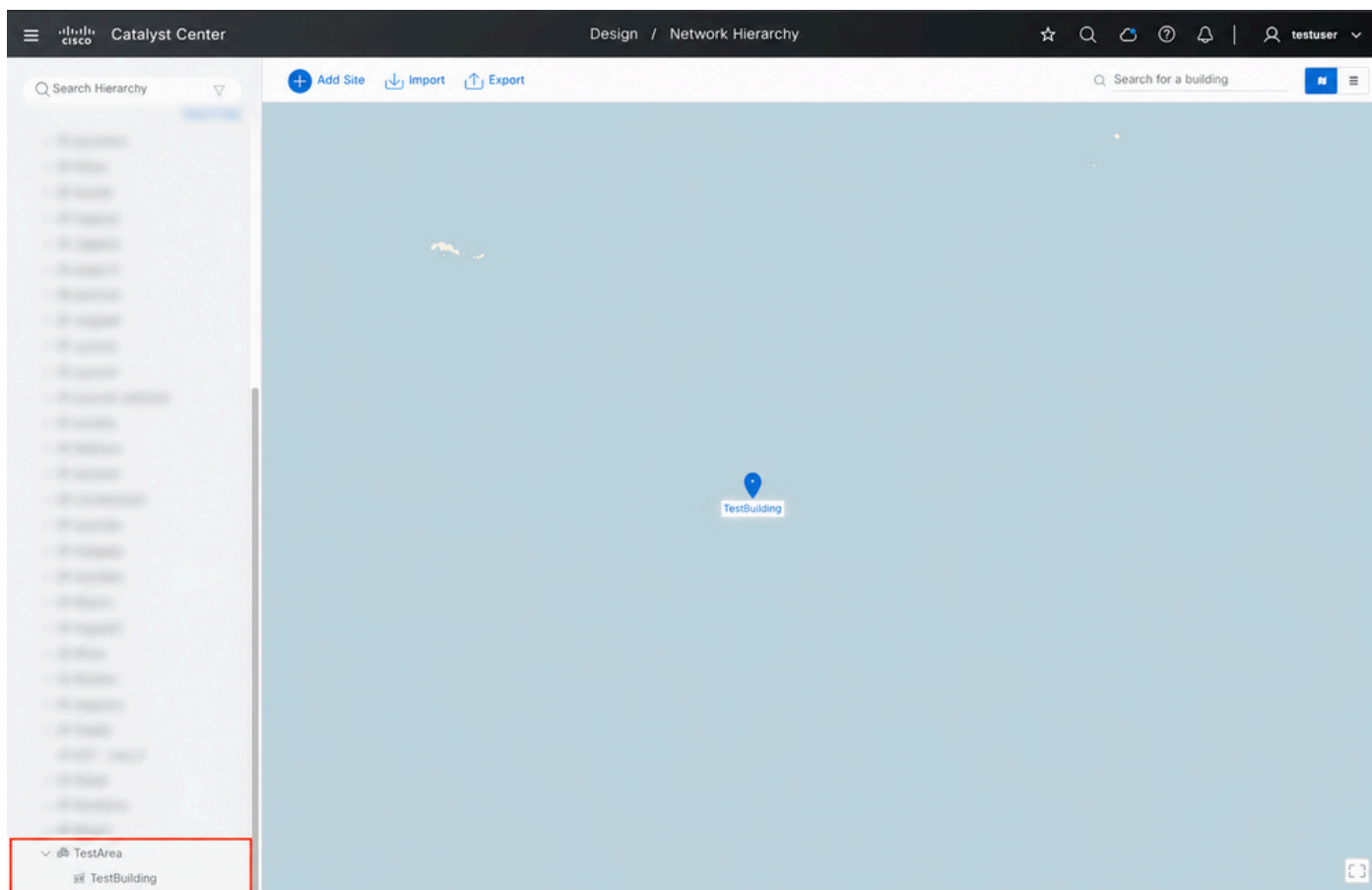
BrunoのAPIコレクションのグローバルヘッダーの入力。

4. [Save] をクリックします。

この時点で、他のCatalyst Center API要求を実行するための認証トークンが生成されています。

## 例1：外構の床を作成する

この例では、ネットワーク階層内にフロアを作成する方法を示します(Global > TestArea > TestBuilding)。

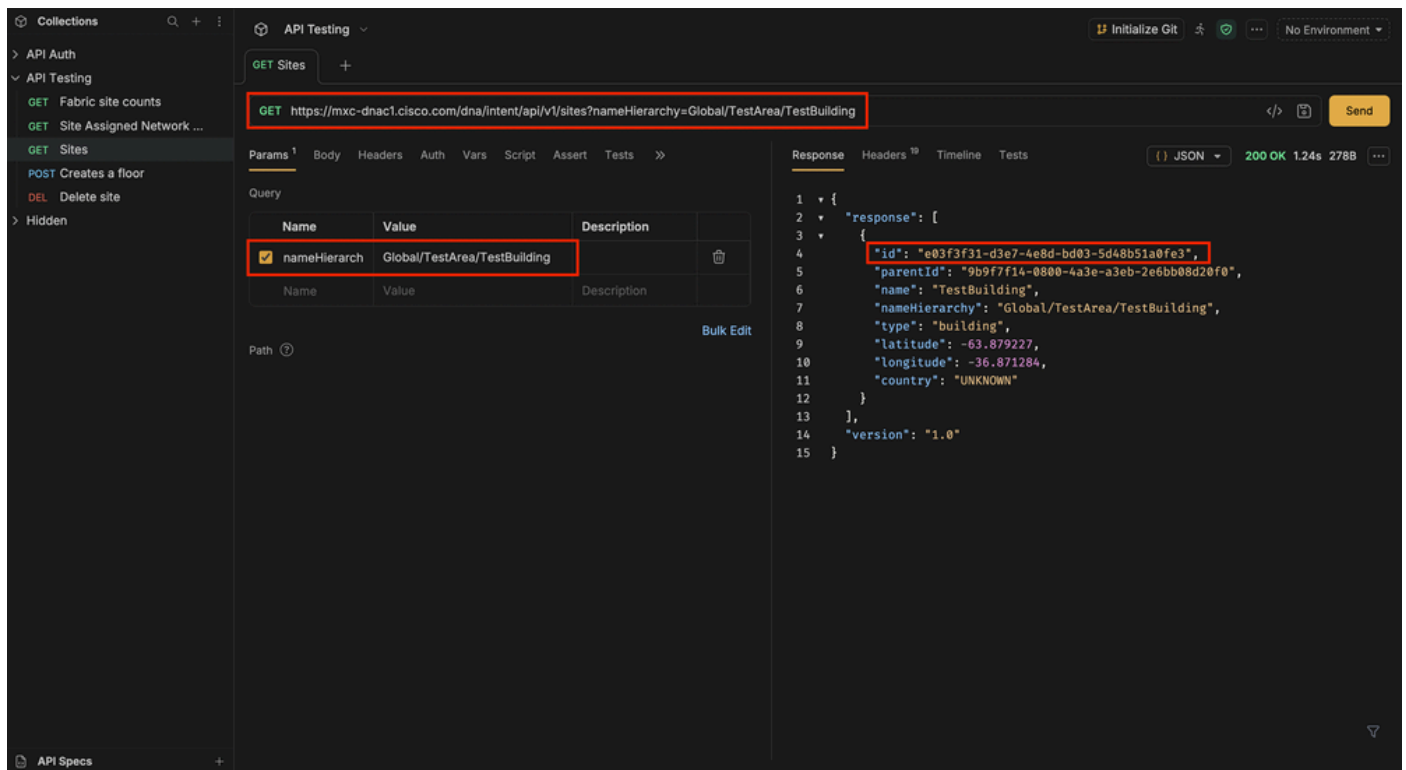


ネットワーク階層

## ステップ1：親IDの取得

GET /dna/intent/api/v1/sites APIを使用してTestBuildingのIDを見つけます。

- クエリパラメータ：nameHierarchy=Global/TestArea/TestBuilding
- 応答からIDをコピーします。これはあなたの親IDです。



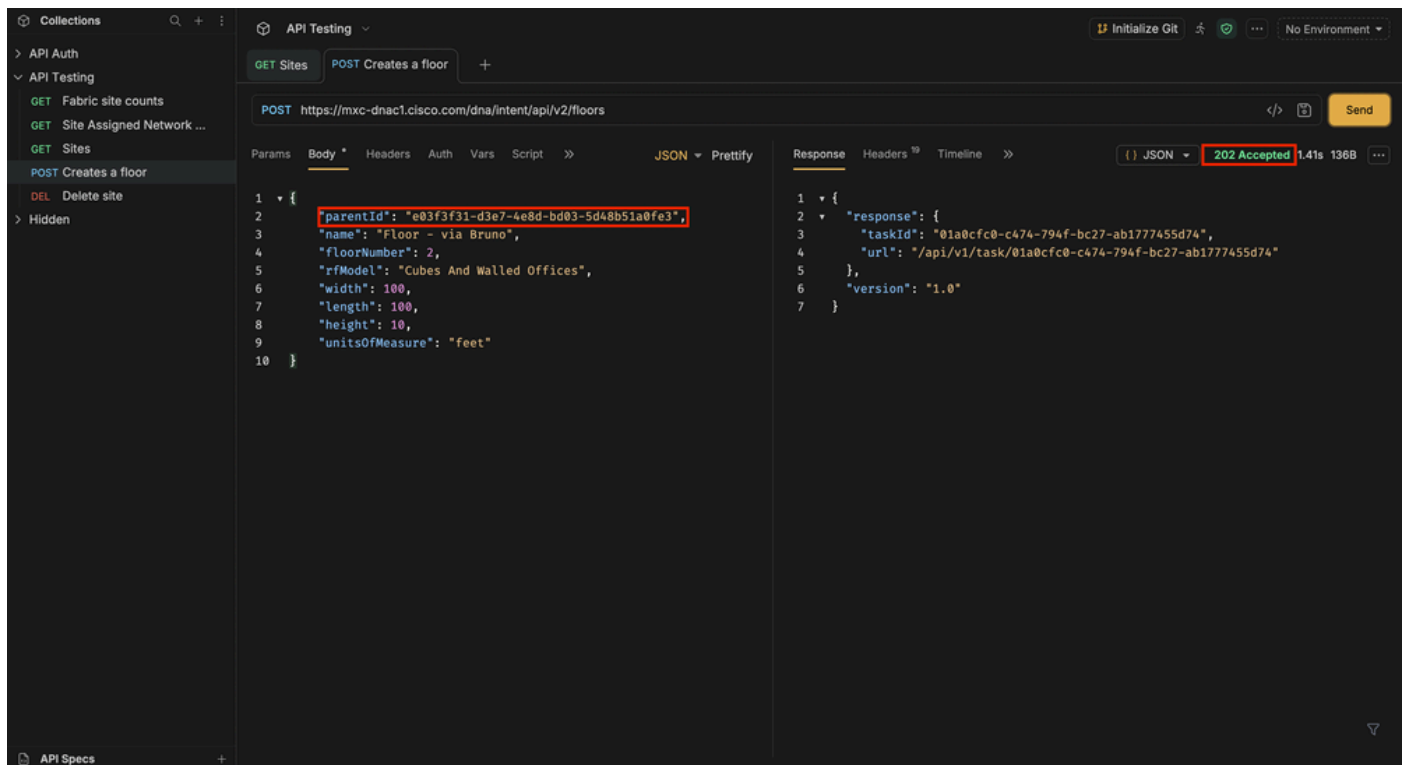
/dna/intent/api/v1/sites?nameHierarchy=Global/TestArea/TestBuildingのGET要求

## 手順2：床を作成する

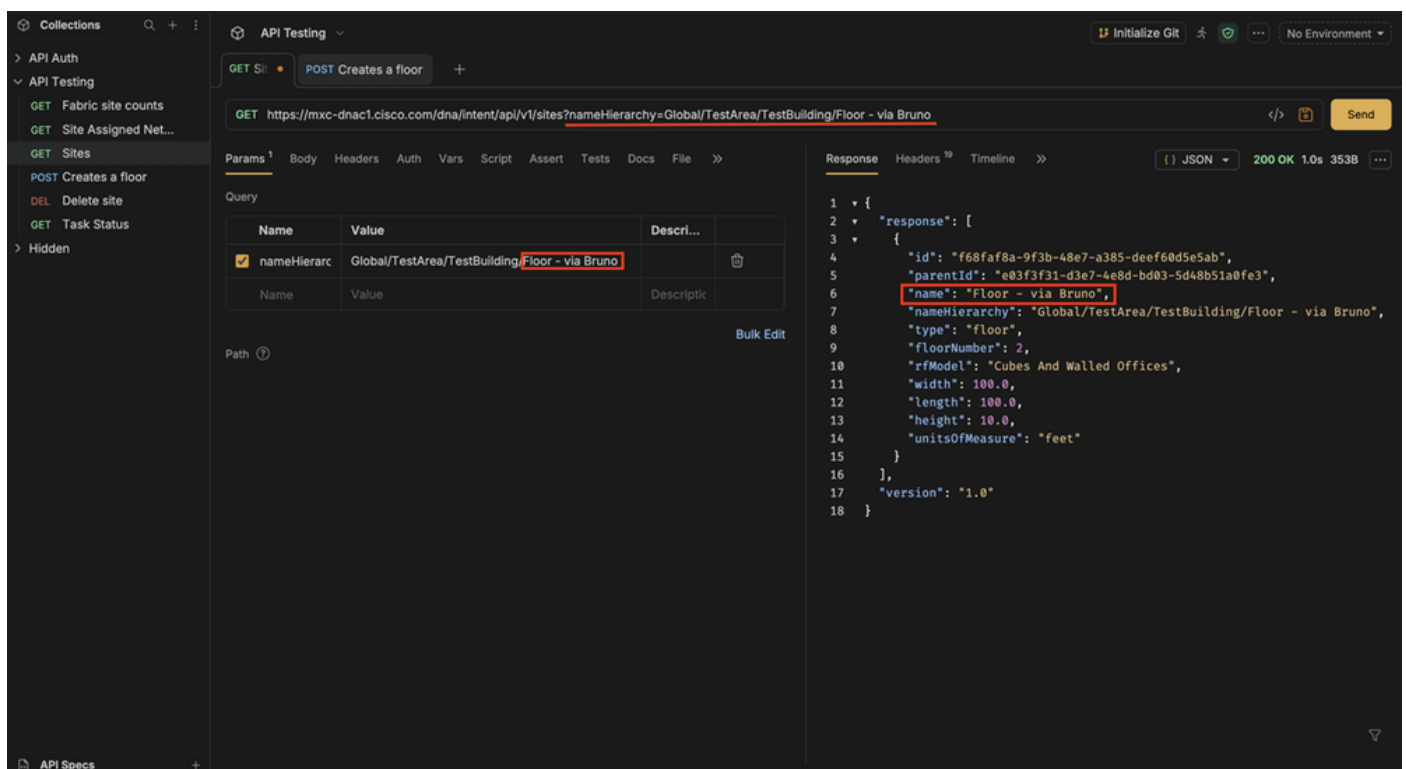
1. /dna/intent/api/v2/floorsに対するPOST要求を作成します。
2. Bodyタブ(JSON)で、必要なパラメータを入力します。

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

3. [Send] をクリックします。応答が成功すると、フロアの作成が確認されます。



POST要求によるフロアの作成



GET APIを使用して、POSTリクエストによってフロアの作成を検証します。

最後に、フロアが正常に作成されたことを検証するために、前のAPIを実行し、上に示すようにクエリを修正して完全な階層パスに一致させることができます。

## 例2：インターフェイス情報の取得

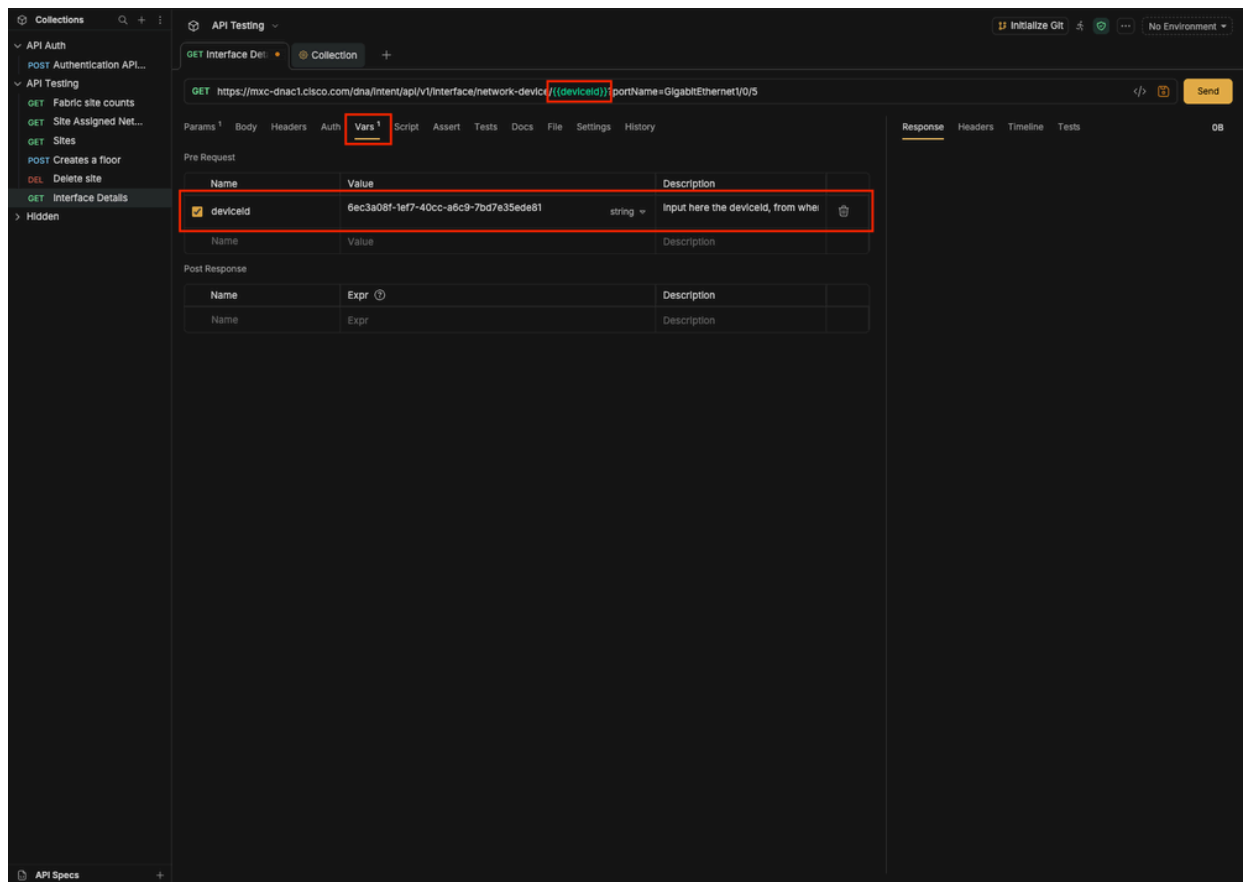
この例では、ネットワークデバイス上の特定のインターフェイスのステータスを取得する方法を示します。

### ステップ1：デバイスIDを特定する

Catalyst CenterのGUIで、Inventoryに移動し、使用しているデバイスをクリックして、ブラウザのURLを確認します。最後のセグメントはデバイスIDです。

### ステップ2: Brunoでパラメータ化されたリクエストを作成する

1. /dna/intent/api/v1/interface/network-device/{{deviceId}}に対するGET要求を作成します。
2. 要求のVarsタブで、deviceIdという名前の変数を追加し、IDを貼り付けます。
3. 特定のポートをフィルタリングするには、クエリーパラメータを追加します。
  - キー：名前
  - 値：GigabitEthernet1/0/5



Brunoでの変数の作成

#### 4. [Send] をクリックします。

The screenshot shows the API Testing interface with a GET request to the Cisco DNA Center API. The request URL is `https://mxc-dnac1.cisco.com/dna/intent/api/v1/interface/network-device/{{deviceId}}?portName=GigabitEthernet1/0/5`. The response is a JSON object containing details about the network interface.

| Name     | Value                | Description |
|----------|----------------------|-------------|
| portName | GigabitEthernet1/0/5 |             |

```
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": " ",
18      "mappedPhysicalInterfaceId": null,
19      "mappedPhysicalInterfaceName": null,
20      "mediaType": null,
21      "mtu": "1500",
22      "nativeVlanId": "1",
23      "ospfSupport": "false",
24      "pid": " ",
25      "portMode": "access",
26      "portName": "GigabitEthernet1/0/5",
27      "portType": "Ethernet Port",
28      "serialNo": " ",
29      "series": " ",
30      "speed": "1000000",
31      "status": "up",
32      "vlanId": "1",
33      "voiceVlan": "",
34      "description": "",
35      "name": null,
36      "instanceUuid": "18208b55-383c-4970-ab8b-db942252974d",
37      "instanceTenantId": "66571d0b1efe061f865c6d5",
38      "id": "18208b55-383c-4970-ab8b-db942252974d"
39    }
40  ],
41   "version": "1.0"
42 }
```

クエリを適用するGET要求

## 結論

Catalyst CenterノースバウンドAPIは、ネットワーク管理者に強力な柔軟性を提供します。GUIベースの管理からAPIベースのワークフローに移行することで、反復的なタスクを自動化し、構成の一貫性を確保し、より広範なITエコシステムにネットワークインテリジェンスを統合できます。

。

## 翻訳について

シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。