

Comprensión y uso de las API de Catalyst Center

Contenido

[Introducción](#)

[Antecedentes](#)

[Prerequisites](#)

[Requirements](#)

[Componentes Utilizados](#)

[Antecedentes](#)

[Introducción a HTTP](#)

[La solicitud HTTP](#)

[La respuesta HTTP](#)

[Métodos HTTP \(verbos\)](#)

[Códigos de estado HTTP](#)

[Descripción general de REST](#)

[Catalyst Center y API REST](#)

[Recuperar un token de autenticación mediante Bruno](#)

[1. Identificar el terminal API](#)

[2. Configure Bruno para la autenticación](#)

[3. Establecer un encabezado global en Bruno](#)

[Ejemplo 1: Crear una planta de sitio](#)

[Paso 1: Obtener la ID principal](#)

[Paso 2: Crear la planta](#)

[Ejemplo 2: Recuperar información de interfaz](#)

[Paso 1: Identificar la ID del dispositivo](#)

[Paso 2: Crear una solicitud parametrizada en Bruno](#)

[Conclusión](#)

Introducción

Este documento describe cómo acceder y utilizar las API CaTC Northbound (Intent) mediante un cliente de API y las herramientas de desarrollo integradas de la plataforma.

Antecedentes

Abarca los conceptos básicos de HTTP/REST y proporciona ejemplos prácticos de automatización y pruebas.

Prerequisites

Requirements

Cisco recomienda conocer estos temas:

- Protocolos HTTP/1.1 y códigos de estado.
- Estilo arquitectónico REST.
- Conocimientos básicos de Cisco Catalyst Center (CaTC)
- Comprensión básica de Bruno (o clientes API similares como Postman).

Componentes Utilizados

La información que contiene este documento se basa en estas versiones de software y hardware.

- Cisco Catalyst Center versiones 2.3.7.x y 3.x
- Bruno versión 4.1.0

La información que contiene este documento se creó a partir de los dispositivos en un ambiente de laboratorio específico. Todos los dispositivos que se utilizan en este documento se pusieron en funcionamiento con una configuración verificada (predeterminada). Si tiene una red en vivo, asegúrese de entender el posible impacto de cualquier comando.

Antecedentes

Las API ascendentes (API por intención) de Catalyst Center permiten a los usuarios interactuar con el controlador fuera de la interfaz gráfica de usuario. Esto permite la automatización de flujos de trabajo a medida, la integración con herramientas de terceros y la gestión eficaz de la red a escala.

Introducción a HTTP

El protocolo de transferencia de hipertexto (HTTP) es la base de la comunicación de datos en la Web. Se adhiere a un modelo cliente-servidor donde el cliente envía una solicitud y el servidor devuelve una respuesta.

La solicitud HTTP

Una solicitud HTTP consta de cuatro partes:

1. Línea de solicitud: Especifica el método (incluidos GET y POST), el URI y la versión del protocolo (como GET /api/v1/nodes HTTP/1.1).
2. Encabezados: Proporcione metadatos como credenciales de autenticación y tipo de contenido.
3. Línea vacía: Separa los encabezados del cuerpo.
4. Cuerpo del mensaje (opcional): Contiene datos enviados al servidor (común en POST/PUT/PATCH).

La respuesta HTTP

Una respuesta HTTP refleja la estructura de solicitudes:

1. Línea de estado: Incluye el código de estado (como 200 OK).
2. Encabezados: Detalles sobre el servidor y los datos de respuesta.
3. Cuerpo del mensaje: Los datos reales solicitados (normalmente en formato JSON para Catalyst Center).

Métodos HTTP (verbos)

- GET: Recupera datos de un recurso.
- POST: Crea un nuevo recurso.
- PUT: Reemplaza un recurso existente.
- PARCHE: Aplica modificaciones parciales a un recurso.
- ELIMINAR: Quita un recurso.

Códigos de estado HTTP

Los códigos de estado HTTP son códigos numéricos estandarizados devueltos por un servidor como parte de una respuesta HTTP. Indican el resultado general de una solicitud de cliente y ayudan a comprender si la solicitud se realizó correctamente, si se requiere una acción adicional

o si se detectó un error.

Los códigos se agrupan en cinco categorías principales según su primer dígito.

- Los códigos 1xx son informativos e indican que la solicitud se ha recibido y que el procesamiento continúa.
- Los códigos 2xx indican una solicitud correcta, lo que significa que el servidor ha recibido, entendido y procesado correctamente la solicitud.
- Los códigos 3xx se utilizan para el redireccionamiento e indican que se puede requerir una acción adicional para completar la solicitud.
- Los códigos 4xx representan errores de cliente, lo que significa que hay un problema con la solicitud enviada por el cliente.
- Los códigos 5xx representan errores de servidor, lo que significa que el servidor encontró un problema al intentar procesar una solicitud.

Descripción general de REST

REST (Representational State Transfer) es un estilo arquitectónico para diseñar aplicaciones en red. Las API RESTful utilizan HTTP como mecanismo de transporte y normalmente utilizan JSON para el intercambio de datos debido a su naturaleza ligera.

Cuando un cliente interactúa con una API REST, el servidor devuelve una representación del estado actual de un recurso. Esta representación se transfiere al cliente como parte de la respuesta y puede expresarse en un formato diferente.

REST no es un protocolo. En su lugar, se trata de un estilo jerárquico que define un conjunto de principios y restricciones para crear sistemas distribuidos y API. En las API modernas, JSON es uno de los formatos más comunes porque es ligero y fácil de procesar.

Una interacción típica de REST se adhiere a un modelo basado en el cliente (esto se mencionó anteriormente en el artículo). El cliente envía una solicitud HTTP mediante métodos como GET, POST, PUT, PATCH o DELETE. El servidor procesa y devuelve una respuesta HTTP que contiene la información solicitada o el resultado de la operación.

Las API RESTful suelen utilizar HTTP como mecanismo de comunicación para realizar operaciones en los recursos. En este modelo, el cliente identifica un recurso o extremo y, a continuación, utiliza el método HTTP para indicar la acción que se debe realizar en ese recurso. Una API REST se puede describir utilizando varios elementos principales. En primer lugar, URI se utiliza para identificar el recurso o extremo al que el cliente desea acceder. El URI proporciona la ubicación o el identificador de ese recurso dentro de la API. En segundo lugar, el cliente utiliza métodos HTTP estándar como GET, POST, PUT, PATCH y DELETE para definir la operación de

solicitud. Por ejemplo, GET se puede utilizar para recuperar un recurso, POST para crear uno y DELETE para eliminar uno por completo. Finalmente, el recurso se intercambia a través de una representación, que se define por su tipo de medio. Esta representación es la forma en que se transfieren los datos de recursos entre el cliente y el servidor, como JSON o XML.

Catalyst Center y API REST

CaTC ayuda a simplificar las operaciones de red de las instalaciones mediante la centralización de la automatización, la configuración, la gestión de políticas, la supervisión y los análisis. Permite a los administradores de red implementar y gestionar dispositivos de forma más eficaz, mantener políticas uniformes, supervisar el comportamiento de la red y mejorar la visibilidad operativa general. Se puede acceder a estas funciones a través de la interfaz gráfica de usuario (GUI), así como mediante API e integraciones externas.

La plataforma CaTC también proporciona API que se pueden utilizar para crear integraciones personalizadas y automatizar las operaciones de red. Al combinar las API de CaTC con las soluciones de Cisco o aplicaciones de terceros existentes, las organizaciones pueden ampliar la funcionalidad de la plataforma y crear flujos de trabajo personalizados para operaciones o requisitos empresariales específicos.

Después de abordar los conceptos básicos, este documento se centra en las API ascendentes, también denominadas API por intención, dentro de Cisco Catalyst Center. Estas API exponen funciones específicas de Catalyst Center mediante una abstracción basada en políticas de la intención empresarial, lo que permite a los administradores y desarrolladores centrarse en el resultado deseado en lugar de en los pasos de configuración individuales necesarios para conseguirlo.

Las API por intención de Catalyst Center son API RESTful que utilizan el formato de datos JSON para interactuar con la plataforma. Se pueden utilizar para descubrir información de red, recuperar datos operativos y realizar tareas de control y automatización compatibles en toda la red.

Recuperar un token de autenticación mediante Bruno

Antes de realizar llamadas API, debe autenticarse para recibir un token.

1. Identificar el terminal API

1. Inicie sesión en Catalyst Center.
2. Vaya a Menú > Plataforma > Kit de herramientas del desarrollador.

3. En el menú de la izquierda, seleccione Authentication > Authentication API.

The screenshot shows the Cisco Catalyst Center Developer Toolkit interface. On the left, a sidebar menu lists various categories, with 'Authentication' highlighted and expanded. The main content area is titled 'Authentication' and contains a table of API endpoints. The 'Authentication API' entry is highlighted with a red box. Below the table, the 'Cisco DNA Center System' section is visible.

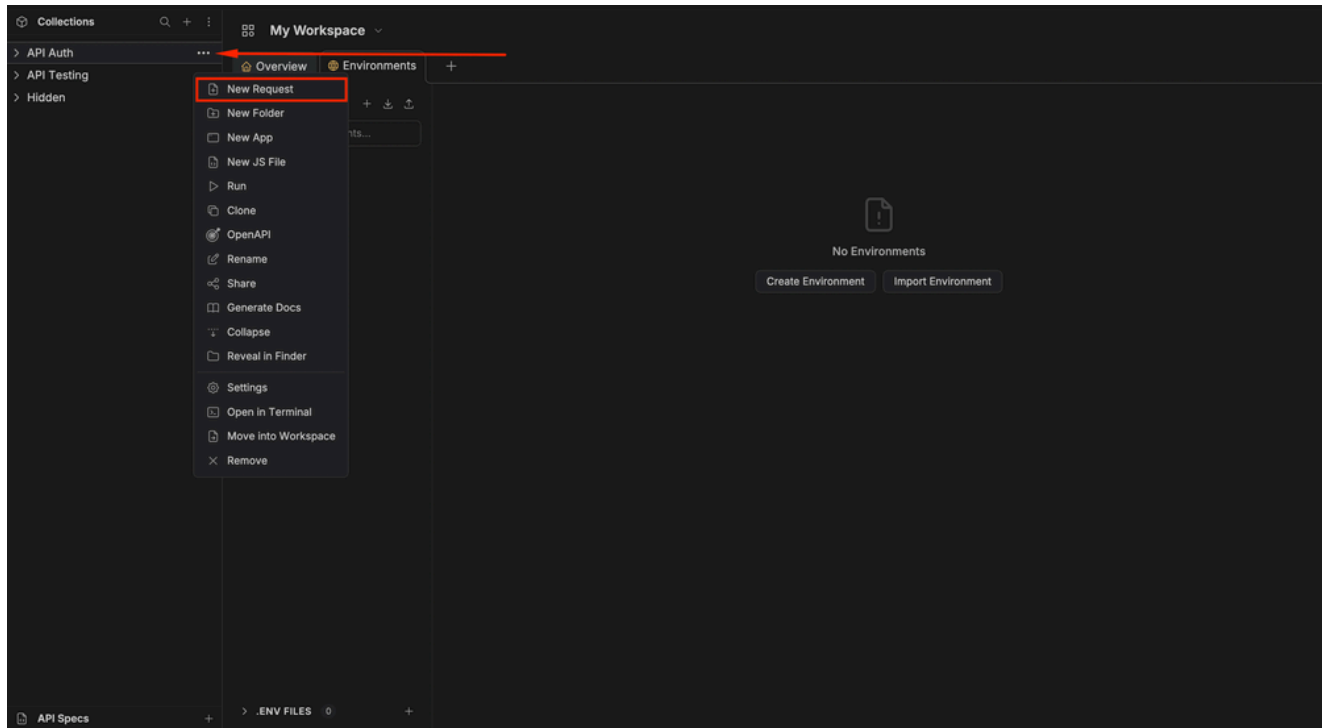
Method	Name	Description	URL	Actions
POST	importCertificate	This API enables a user to import a PEM certificate and its key for the controller and/or disaster recovery.	/dna/intent/api/v1/certificate	...
POST	importCertificateP12	This API enables a user to import a PKCS12 certificate bundle for the controller 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	...

Kit de herramientas para desarrolladores de Catalyst Center

4. Copie la URL (normalmente POST /dna/system/api/v1/auth/token).

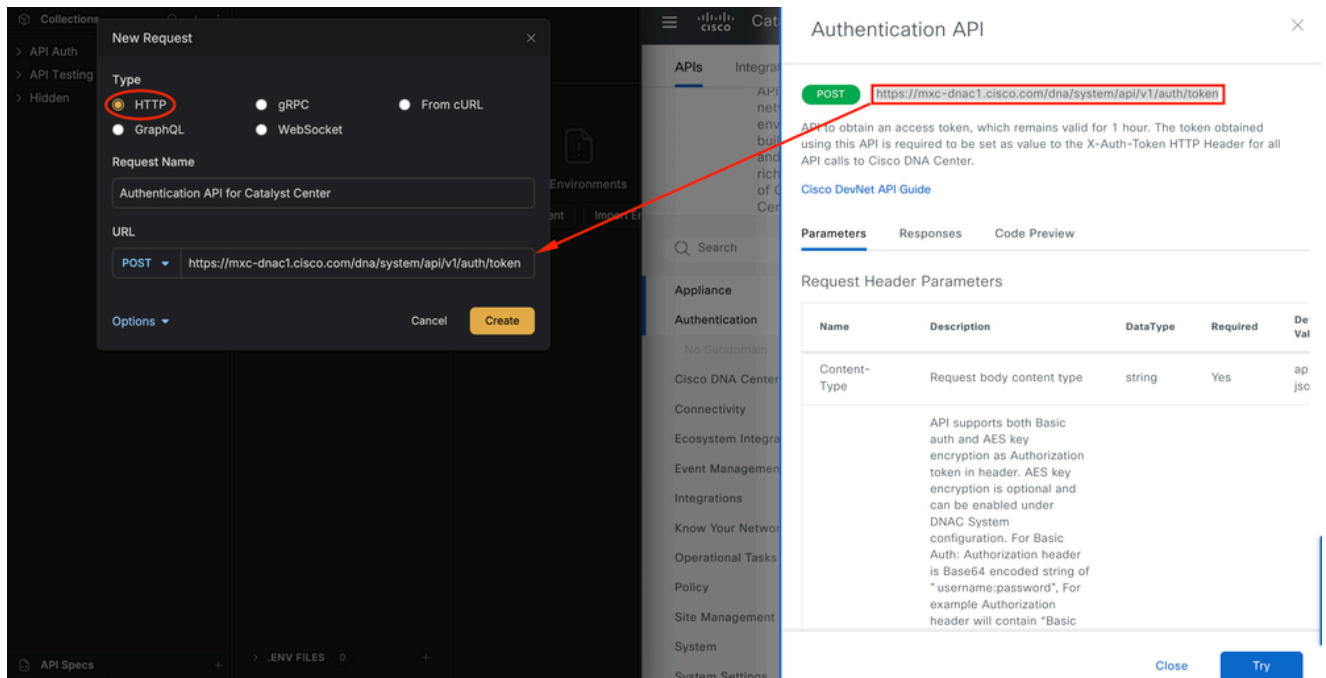
2. Configure Bruno para la autenticación

1. En Bruno, cree una nueva solicitud.
2. Establezca el Tipo en HTTP y el Método en POST..



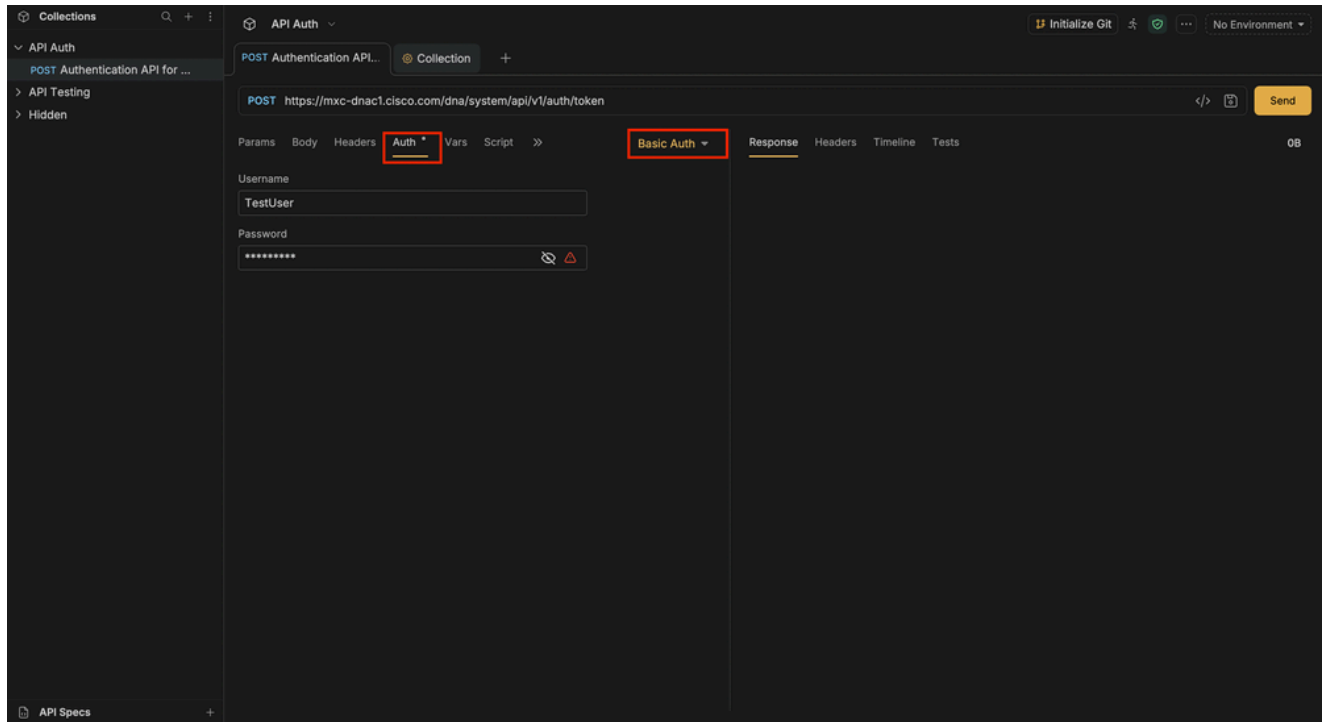
Creación de una nueva solicitud en Bruno

3. Introduzca la URL de la API de autenticación.



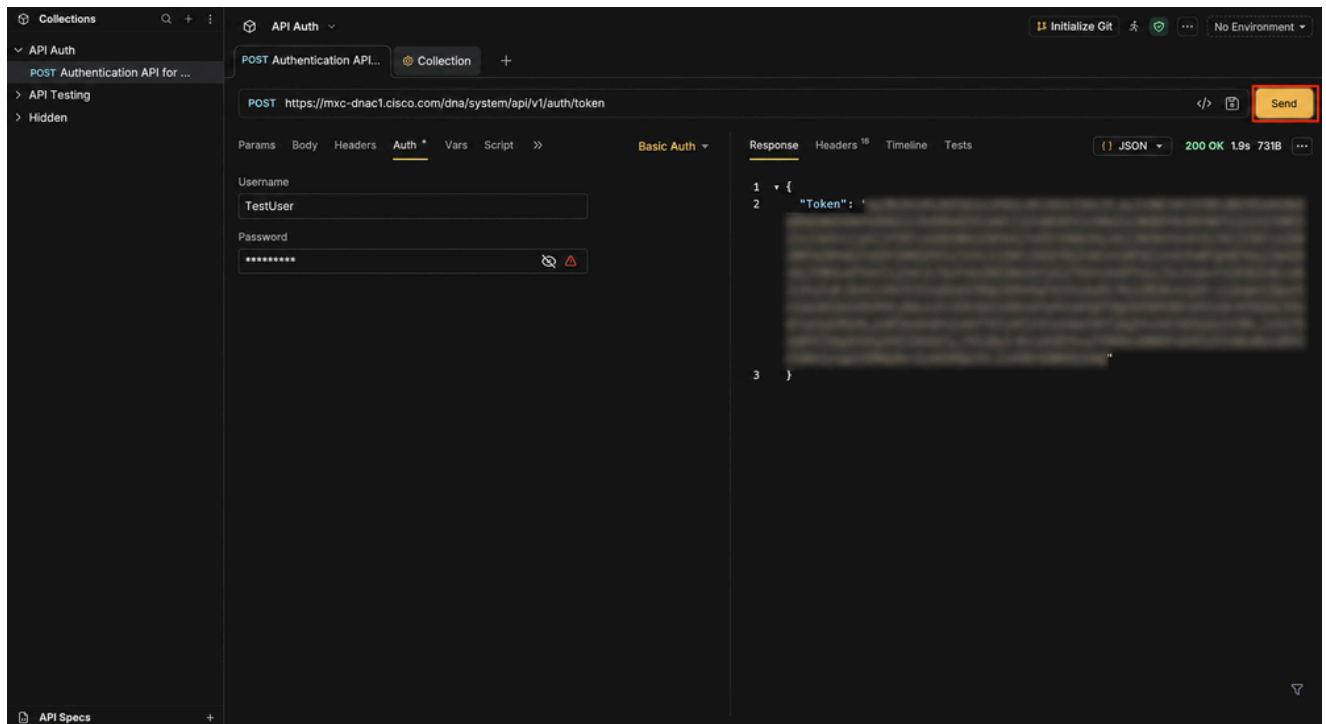
Creación de una nueva solicitud en Bruno

4. Vaya a la pestaña Auth y seleccione Basic Auth.



Autenticación básica para la API de autenticación

5. Introduzca sus credenciales de Catalyst Center SUPER ADMIN.
6. Haga clic en Send.



Ejecución de la API de autenticación

El servidor devuelve un estado 200 OK y una carga JSON que contiene el Token.

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

```
[root@localhost ~]# echo -n 'TestUser:Test123456' | base64
VGZzdFVzZXI6VGZzdDEyMyEhIQ==
```

Obtención de la codificación base64 para el nombre de usuario y la contraseña

Catalyst Center Authentication API

Try 'Authentication API'

Method: POST Public URL :https://mxc-dnac1.cisco.com/dna/system/api/v1/auth/token

PARAMETERS

HEADERS

Content-Type* application/json

Authorization* Basic VGZzdFVzZXI6VGZzdDEyMyEhIQ==

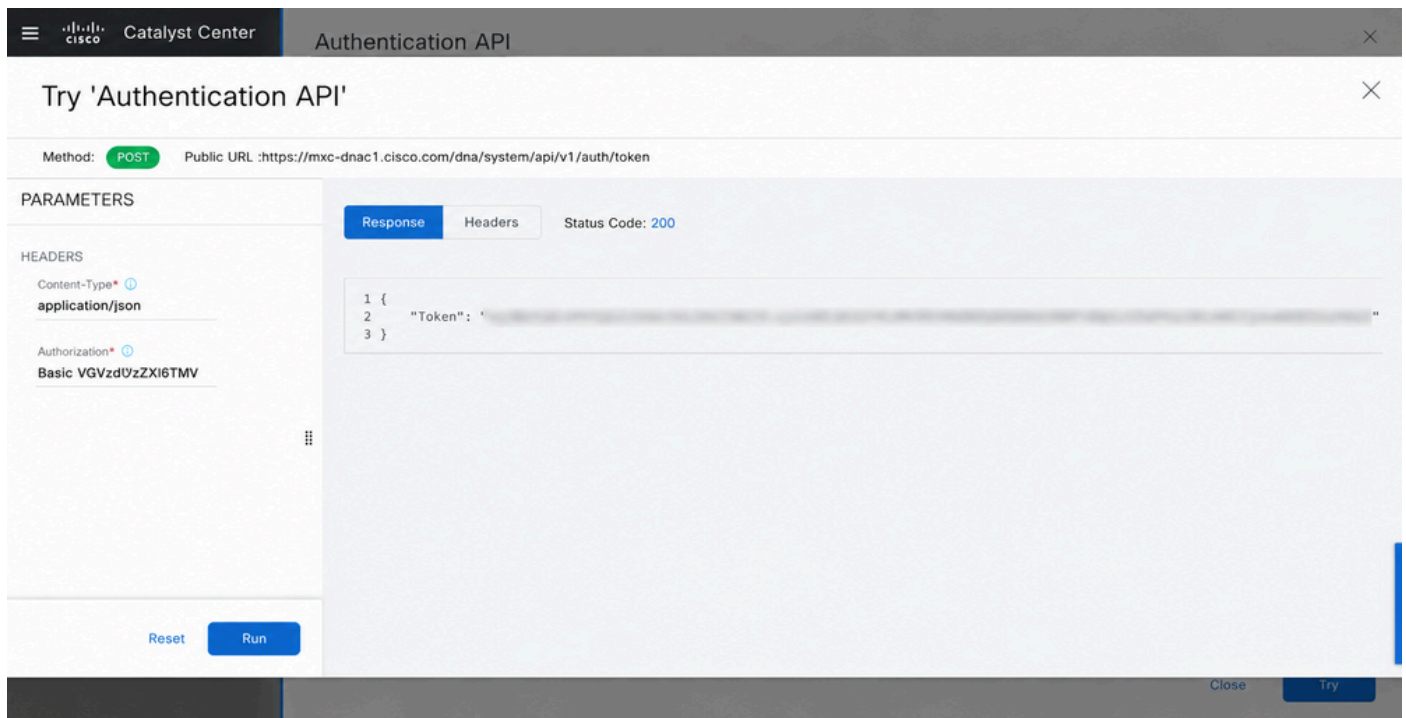
Reset Run

Enter Parameters

First enter in some necessary parameters on the left and then click Run.

Close Try

Introducción de una autenticación básica con nombre de usuario y contraseña base64 codificados



Obteniendo el token de autenticación de la GUI de Catalyst Center.

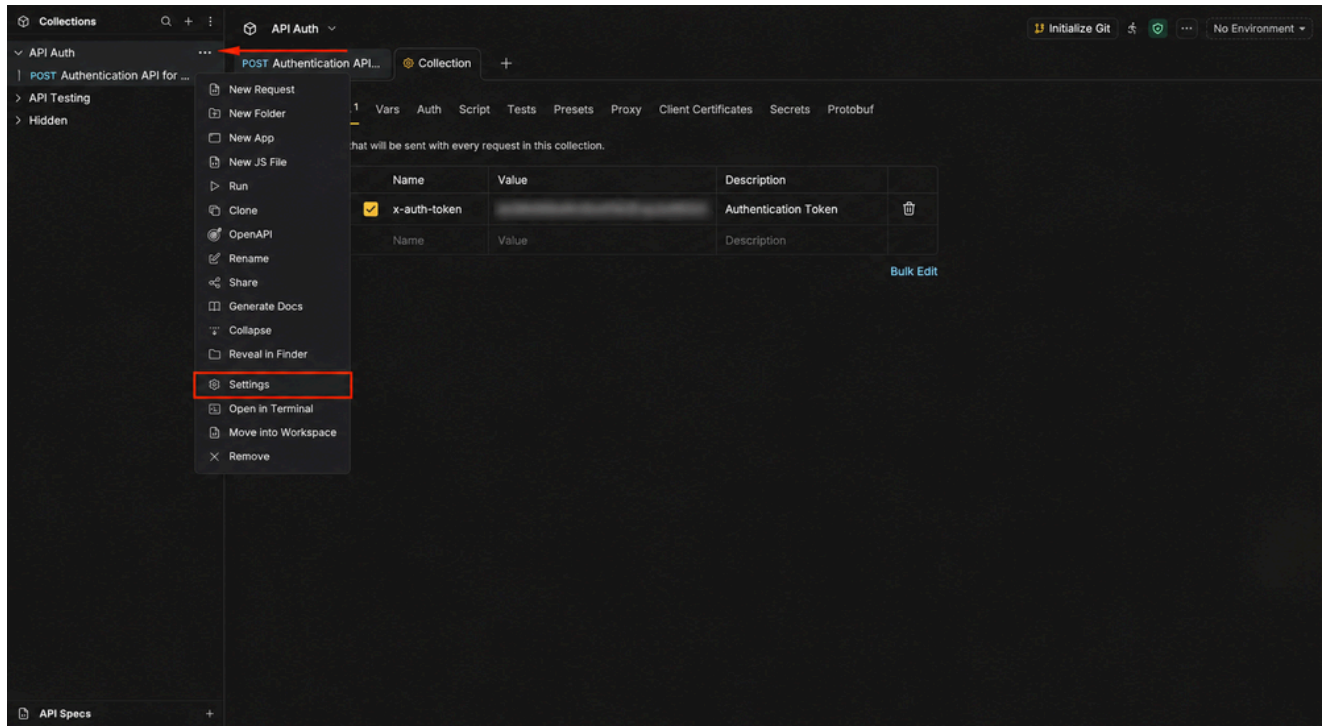
En este punto, tiene el token de autenticación, ya sea en la respuesta de Bruno o en la respuesta realizada desde el kit de herramientas para desarrolladores de Catalyst Center.

Ahora agregue el token como Header en la configuración de recolección en Bruno, esto facilita cualquier otra solicitud creada dentro de la recolección para utilizar el token, de modo que no tenga necesidad de ingresar el token manualmente como encabezado para cada solicitud.

3. Establecer un encabezado global en Bruno

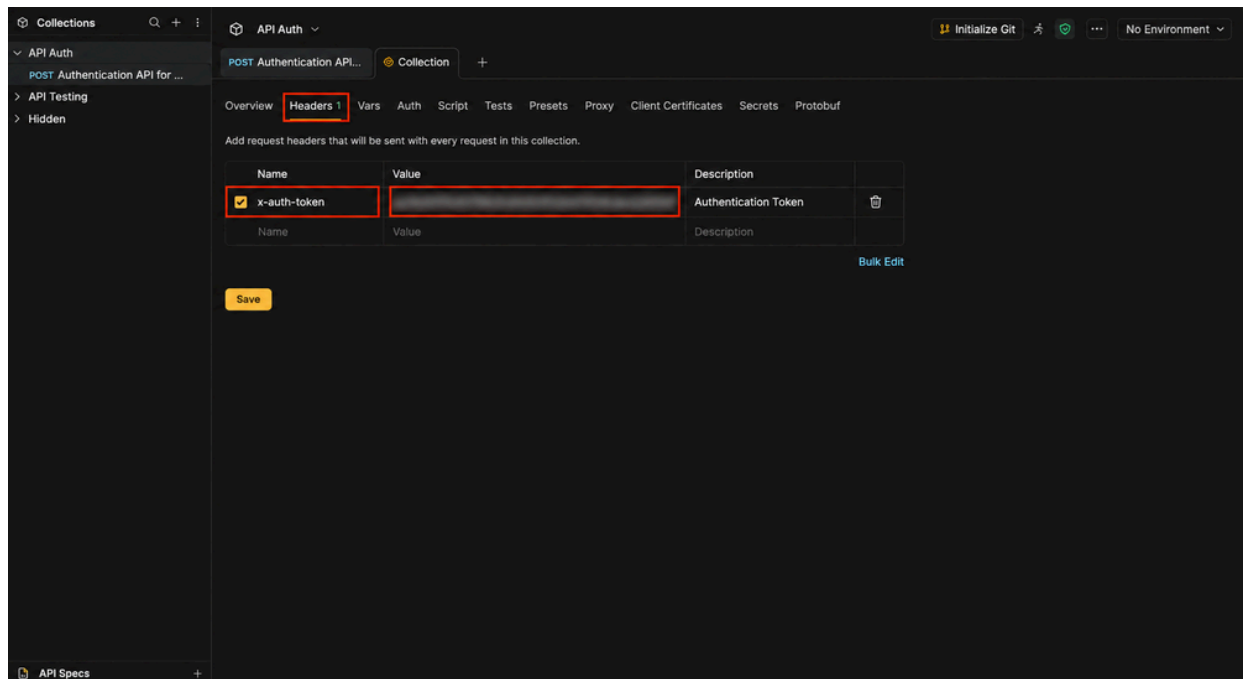
Para evitar la reautenticación para cada solicitud, establezca el token como encabezado de nivel de recolección:

1. Haga clic con el botón derecho del ratón en su colección y seleccione Configuración.



Configuración de colección

2. Vaya a la ficha Encabezados.
3. Agregar un nuevo encabezado:
 - Nombre: x-auth-token
 - Valor: [Pegue aquí su token]



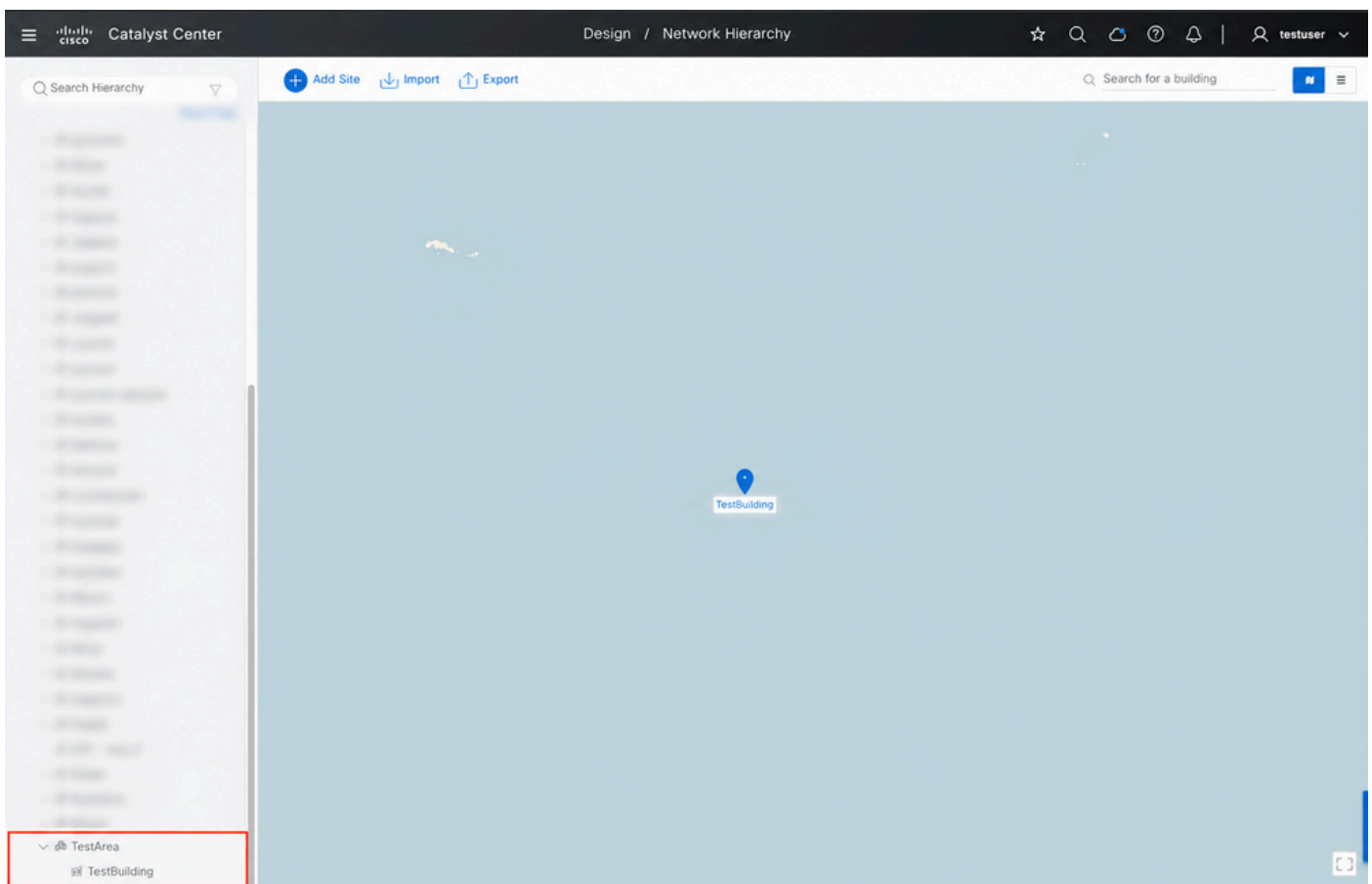
Introducir encabezados globales para la colección API en Bruno.

4. Click Save.

En este momento, ha generado el token de autenticación para realizar cualquier otra solicitud de la API de Catalyst Center.

Ejemplo 1: Crear una planta de sitio

En este ejemplo se muestra cómo crear una planta dentro de la jerarquía de red (Global > TestArea > TestBuilding).

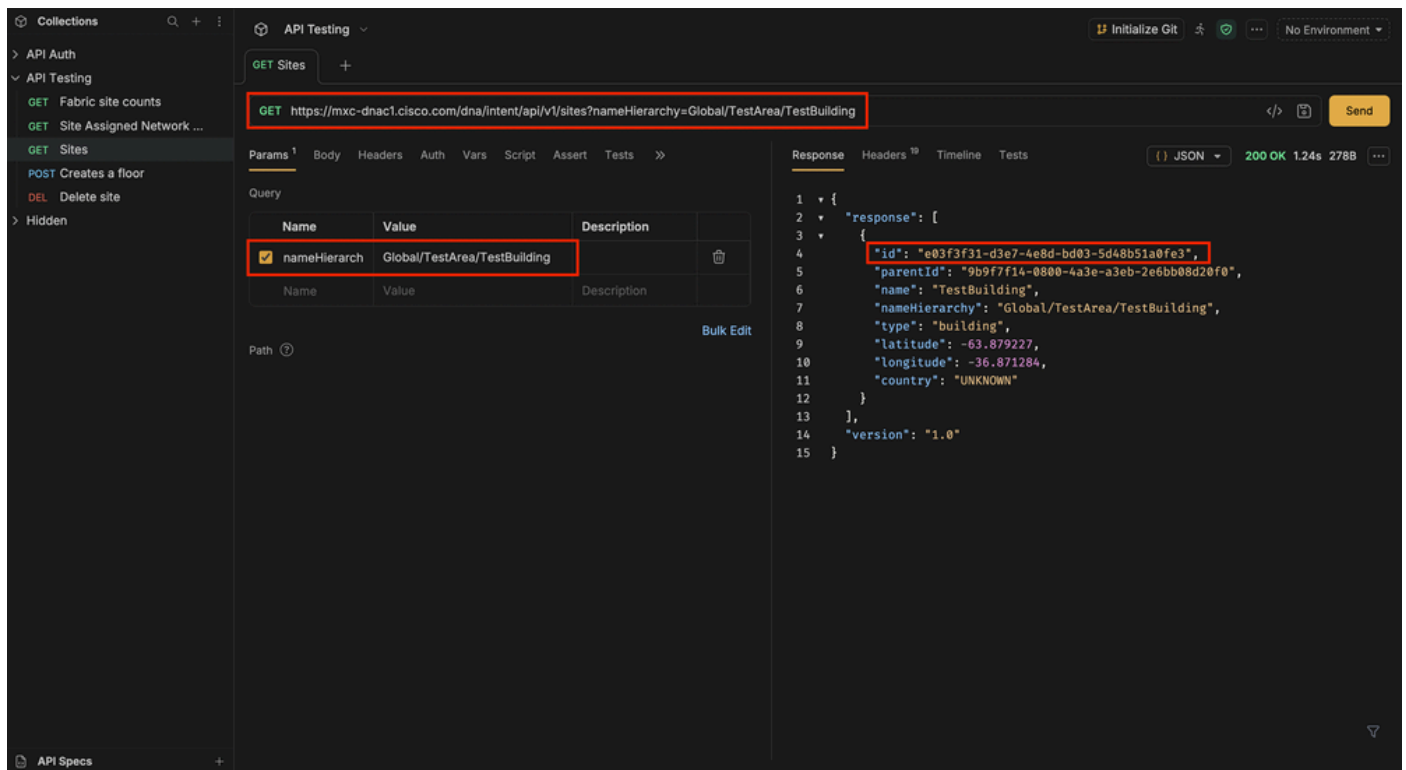


Jerarquía de red

Paso 1: Obtener la ID principal

Utilice la API GET `/dna/intent/api/v1/sites` para encontrar el ID de TestBuilding.

- Parámetro de consulta: `nameHierarchy=Global/TestArea/TestBuilding`
- Copie el ID de la respuesta; éste es su `parentId`.



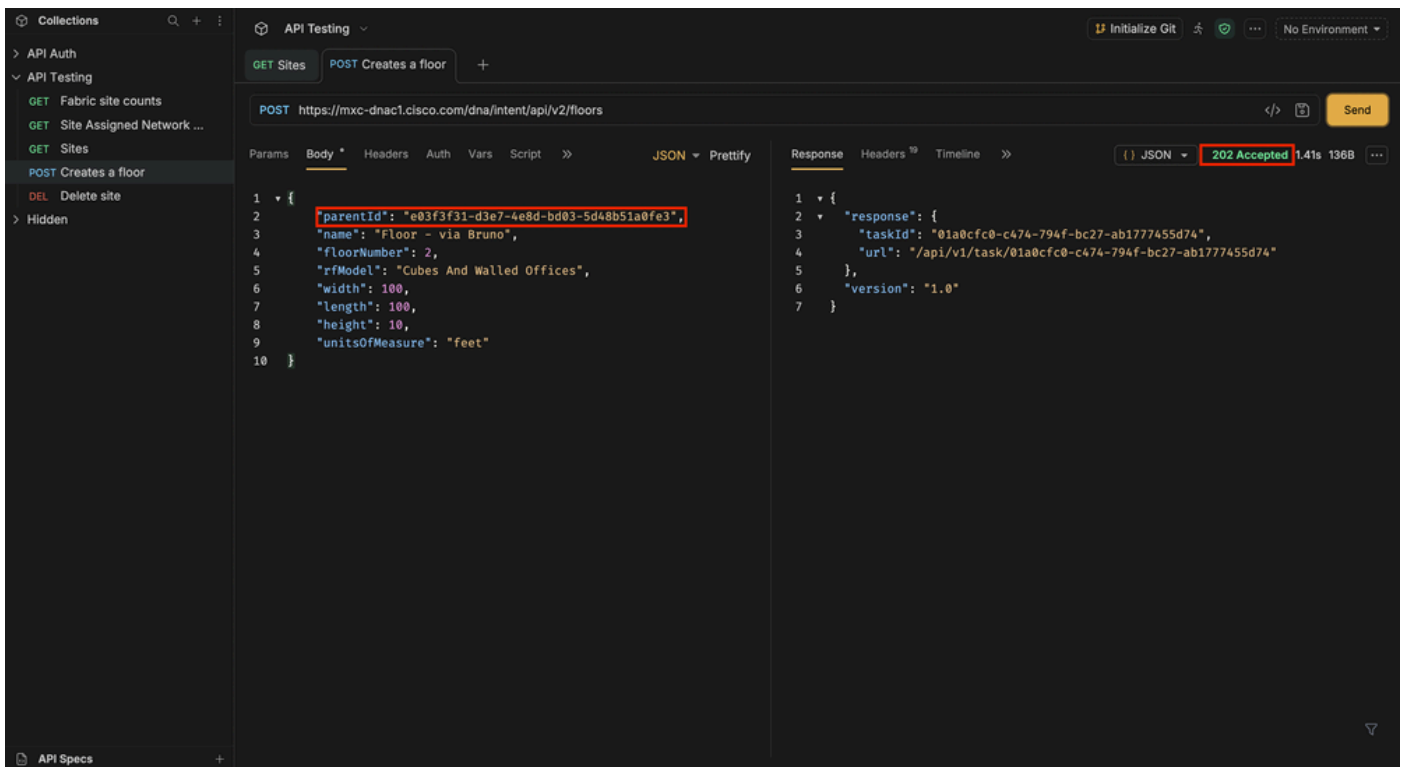
GET request for `/dna/intent/api/v1/sites?nameHierarchy=Global/TestArea/TestBuilding`

Paso 2: Crear la planta

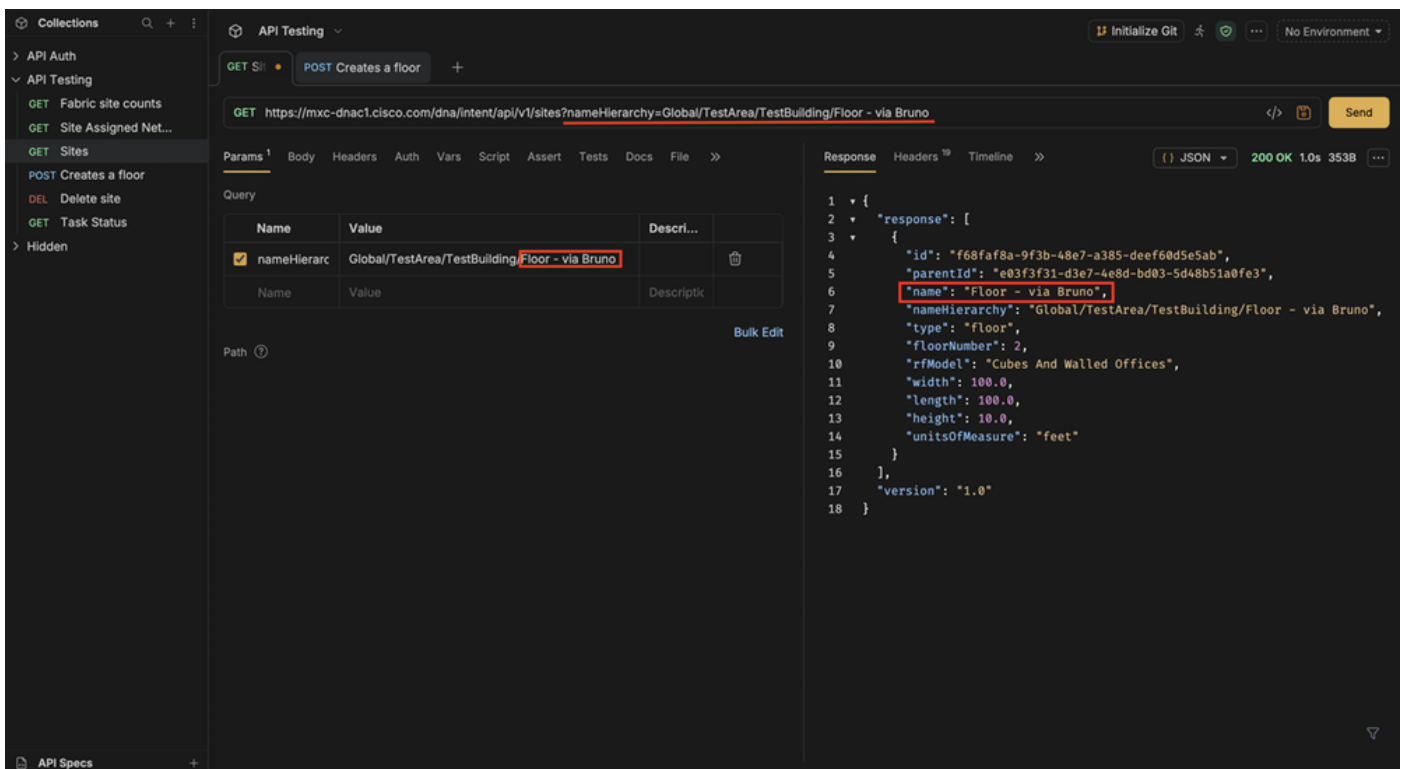
1. Cree una solicitud POST para `/dna/intent/api/v2/floor`.
2. En la ficha Body (Cuerpo) (JSON), introduzca los parámetros necesarios:

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

3. Haga clic en Send. Una respuesta exitosa confirma la creación del piso.



Solicitud POST para crear la planta



OBTENGA API para validar la creación del suelo a través de una solicitud POST.

Por último, para validar que la planta se creó correctamente, puede ejecutar la API anterior y modificar la consulta para que coincida con la ruta jerárquica completa, como se muestra anteriormente.

Ejemplo 2: Recuperar información de interfaz

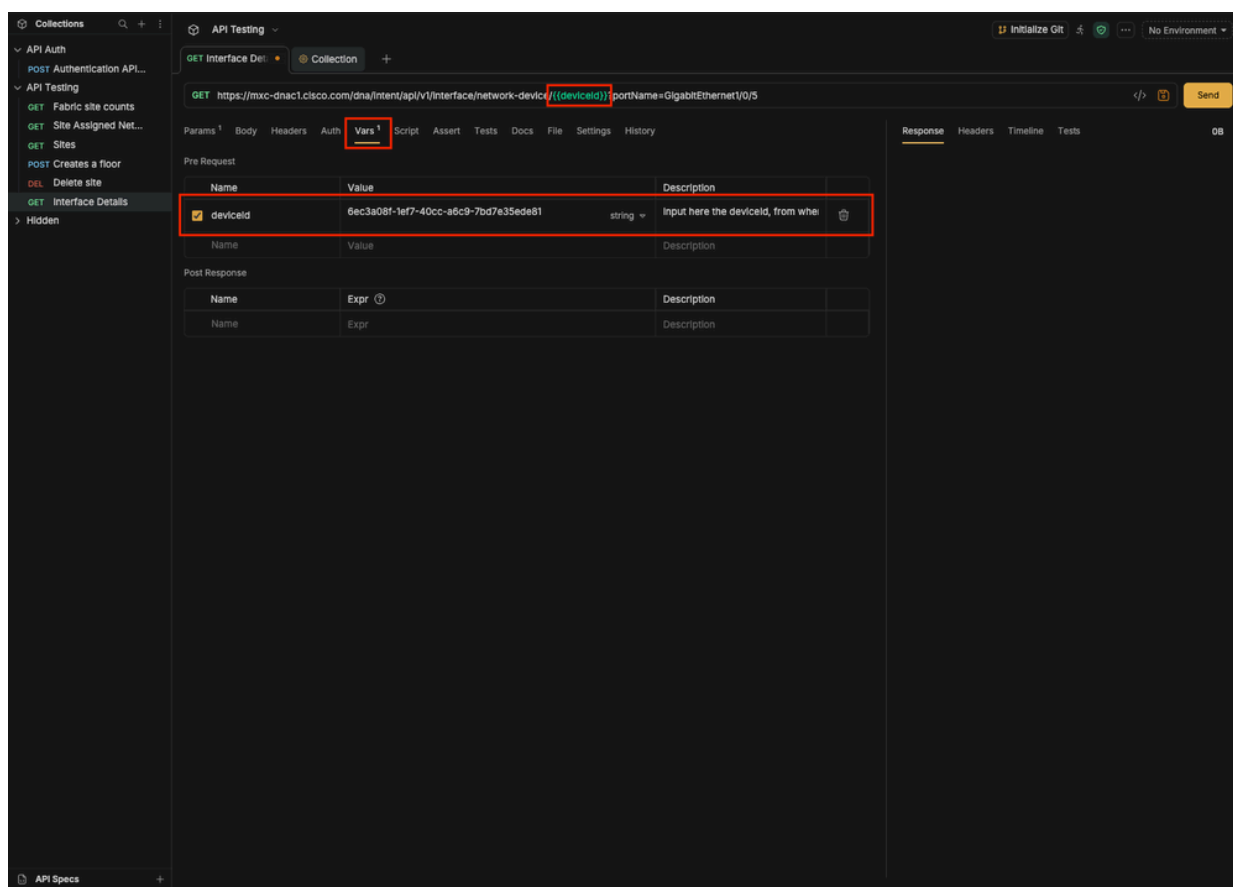
Este ejemplo muestra cómo obtener el estado de una interfaz específica en un dispositivo de red.

Paso 1: Identificar la ID del dispositivo

En la GUI de Catalyst Center, vaya a Inventory, haga clic en su dispositivo y observe la URL en su navegador. El último segmento es el Device ID.

Paso 2: Crear una solicitud parametrizada en Bruno

1. Cree una solicitud GET para `/dna/intent/api/v1/interface/network-device/{{deviceId}}`.
2. En la pestaña Vars de la solicitud, agregue una variable llamada `deviceId` y pegue su ID.
3. Para filtrar por un puerto específico, agregue un parámetro de consulta:
 - Clave: `nombre`
 - Valor: `GigabitEthernet1/0/5`



Creación de variables en Bruno

4. Haga clic en Send.

The screenshot displays the API Testing tool interface. On the left, a sidebar shows a collection of API endpoints under 'API Testing', with 'GET Interface Details' selected. The main panel shows a GET request to the URL: `https://mxc-dnac1.cisco.com/dna/intent/api/v1/interface/network-device/{{deviceid}}?portName=GigabitEthernet1/0/5`. The 'Params' tab is active, showing a table with one parameter:

Name	Value	Description
portName	GigabitEthernet1/0/5	

Below the table is a 'Bulk Edit' button. The 'Response' tab on the right shows the JSON output of the request, which is a 200 OK status. The response is a JSON object containing detailed information about the network interface, including its name, type, status, and various identifiers.

```
1 {
2   "response": [
3     {
4       "addresses": [],
5       "adminStatus": "UP",
6       "className": null,
7       "deviceId": "6ce3a88f-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      "instanceBuildId": "18208b55-383c-4970-ab8b-db942252974d",
37      "instanceTenantId": "6657140b1efef061f865c6d5",
38      "id": "18208b55-383c-4970-ab8b-db942252974d"
39    }
40  ],
41  "version": "1.0"
42 }
```

solicitud GET aplicar una consulta

Conclusión

Las API ascendentes de Catalyst Center proporcionan una gran flexibilidad a los administradores de red. Al pasar de una gestión basada en GUI a flujos de trabajo impulsados por API, puede automatizar tareas repetitivas, garantizar la coherencia de la configuración e integrar la inteligencia de red en su ecosistema de TI más amplio.

Acerca de esta traducción

Cisco ha traducido este documento combinando la traducción automática y los recursos humanos a fin de ofrecer a nuestros usuarios en todo el mundo contenido en su propio idioma.

Tenga en cuenta que incluso la mejor traducción automática podría no ser tan precisa como la proporcionada por un traductor profesional.

Cisco Systems, Inc. no asume ninguna responsabilidad por la precisión de estas traducciones y recomienda remitirse siempre al documento original escrito en inglés (insertar vínculo URL).