Konfigurieren von sicherem Zugriff zur Verwendung der REST-API mit Python

Inhalt

Einleitung

In diesem Dokument werden die Schritte zum Konfigurieren des API-Zugriffs und zum Abrufen von Ressourceninformationen aus dem sicheren Zugriff beschrieben.

Voraussetzungen

Cisco empfiehlt, dass Sie über Kenntnisse in folgenden Bereichen verfügen:

1. Python 3.x
2. REST-API
3. Sicherer Zugriff von Cisco

Anforderungen

Diese Anforderungen müssen erfüllt sein, bevor weitere Schritte unternommen werden können:

- Cisco Secure Access-Benutzerkonto mit der Rolle "Vollständiger Administrator"
- Cisco Security Cloud Single Sign On (SCSO)-Konto für die Anmeldung bei Secure Access.

Verwendete Komponenten

Die Informationen in diesem Dokument basierend auf folgenden Software- und Hardware-Versionen:

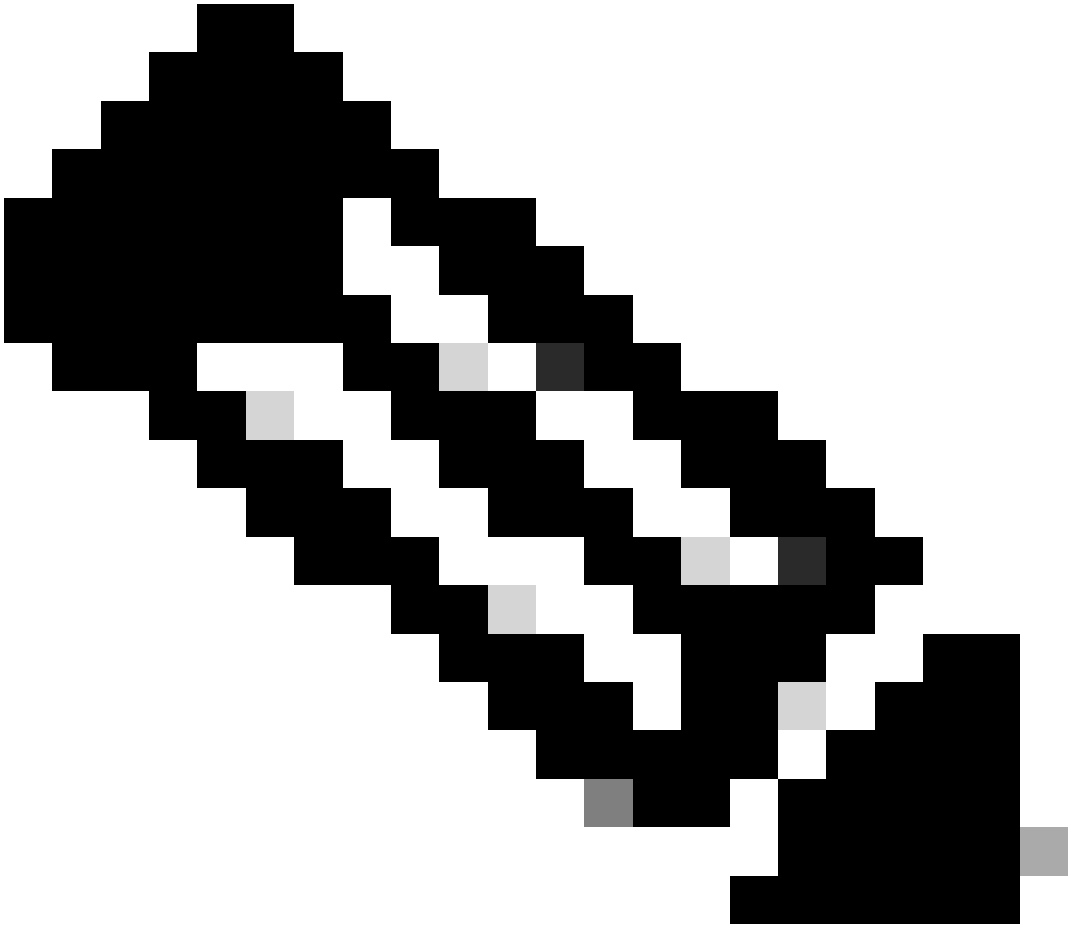
- Dashboard für sicheren Zugriff

- Python

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.

Konfigurieren

Die Secure Access-API stellt eine Standard-REST-Schnittstelle bereit und unterstützt den OAuth 2.0-Client-Anmeldedatenfluss. Melden Sie sich zunächst bei Secure Access an, und erstellen Sie die Schlüssel für die Secure Access-API. Verwenden Sie dann Ihre API-Anmeldeinformationen, um ein API-Zugriffstoken zu generieren.



Hinweis: API-Schlüssel, Kennwörter, Schlüssel und Token ermöglichen den Zugriff auf Ihre privaten Daten. Sie dürfen Ihre Anmeldeinformationen niemals für einen anderen Benutzer oder eine andere Organisation freigeben.

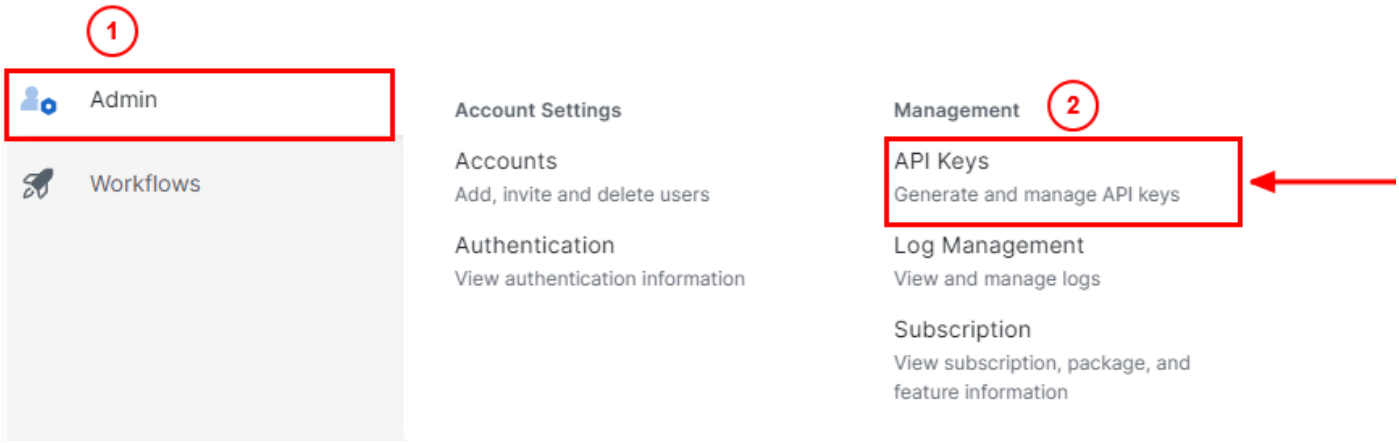
Konfigurieren Sie den API-Schlüssel über das Secure Access Dashboard, bevor Sie die in diesem Artikel erwähnten Skripts ausführen.

API-Schlüssel erstellen

Erstellen Sie mit diesen Schritten einen API-Schlüssel und -Schlüssel. Melden Sie sich mit der URL "Secure Access" beim sicheren Zugriff an.

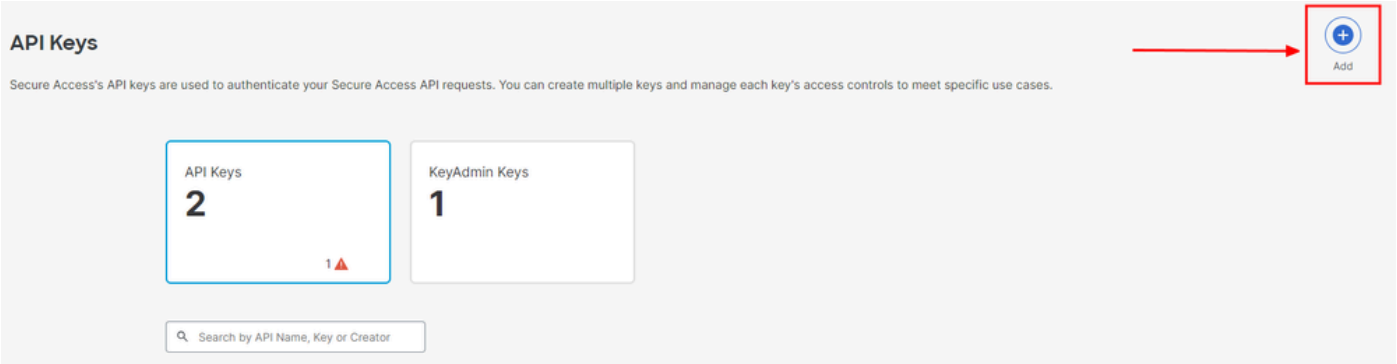
1. Wählen Sie die Option in der linken Seitenleiste ausAdmin.

   - Wählen Sie unter Admin Option **API Keys**:



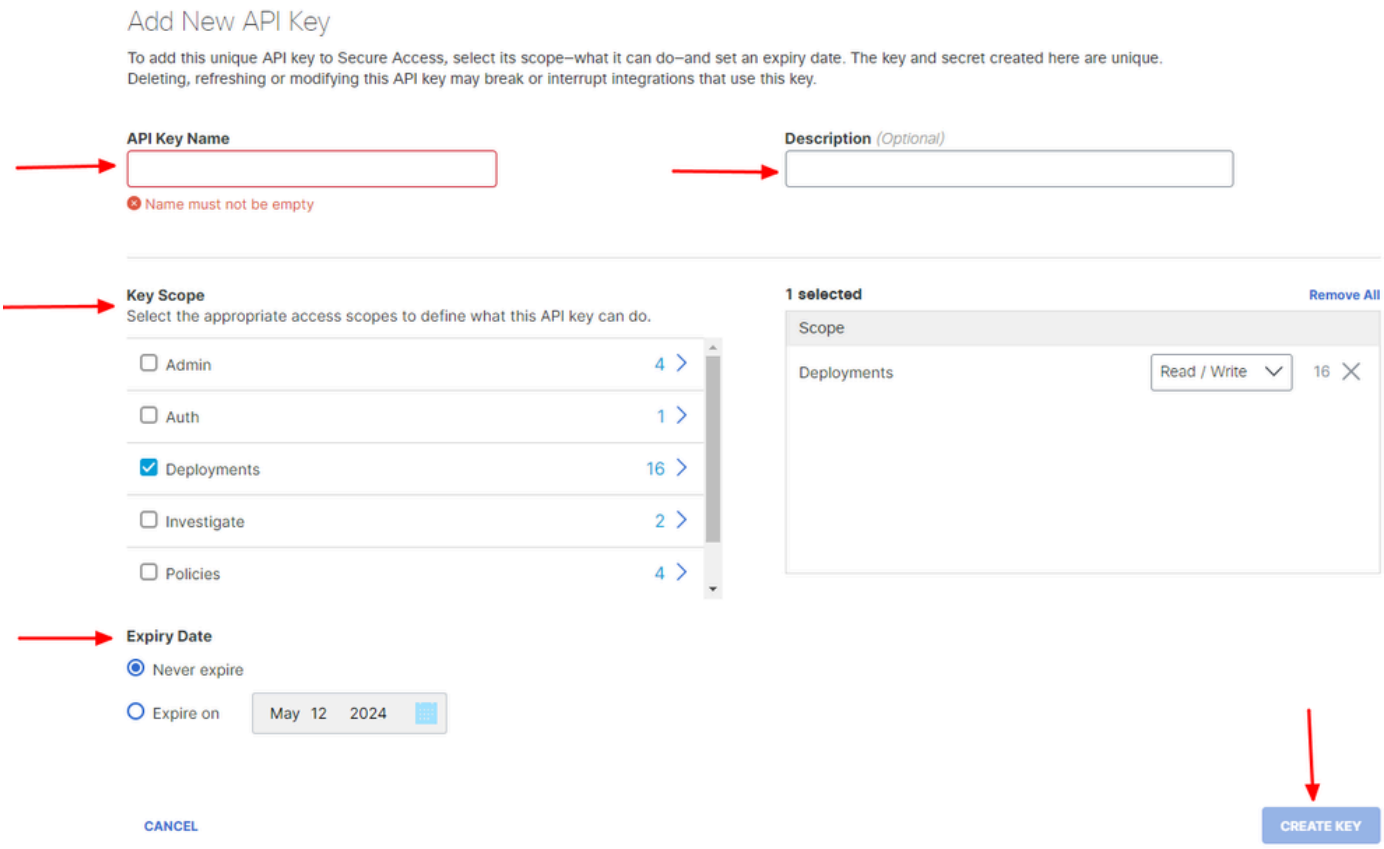
Dashboard-Verwaltung für sicheren Zugriff - API-Schlüssel

3. Klicken Sie oben rechts auf die + Schaltfläche Add a new API Key (Neuen API-Schlüssel hinzufügen):



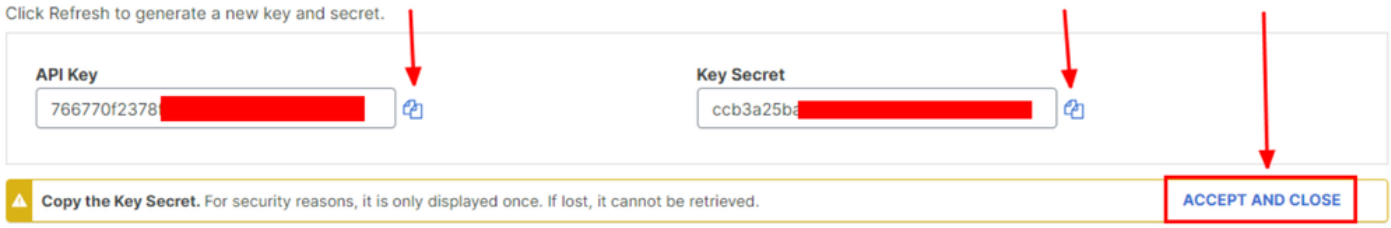
Sicherer Zugriff - API-Schlüssel hinzufügen

4. Geben Sie die **API Key Name, Description**(Optional) an, und wählen Sie die Key scope und Expiry date entsprechend Ihrer Anforderung aus. Klicken Sie abschließend auf die Schaltfläche **Create**:

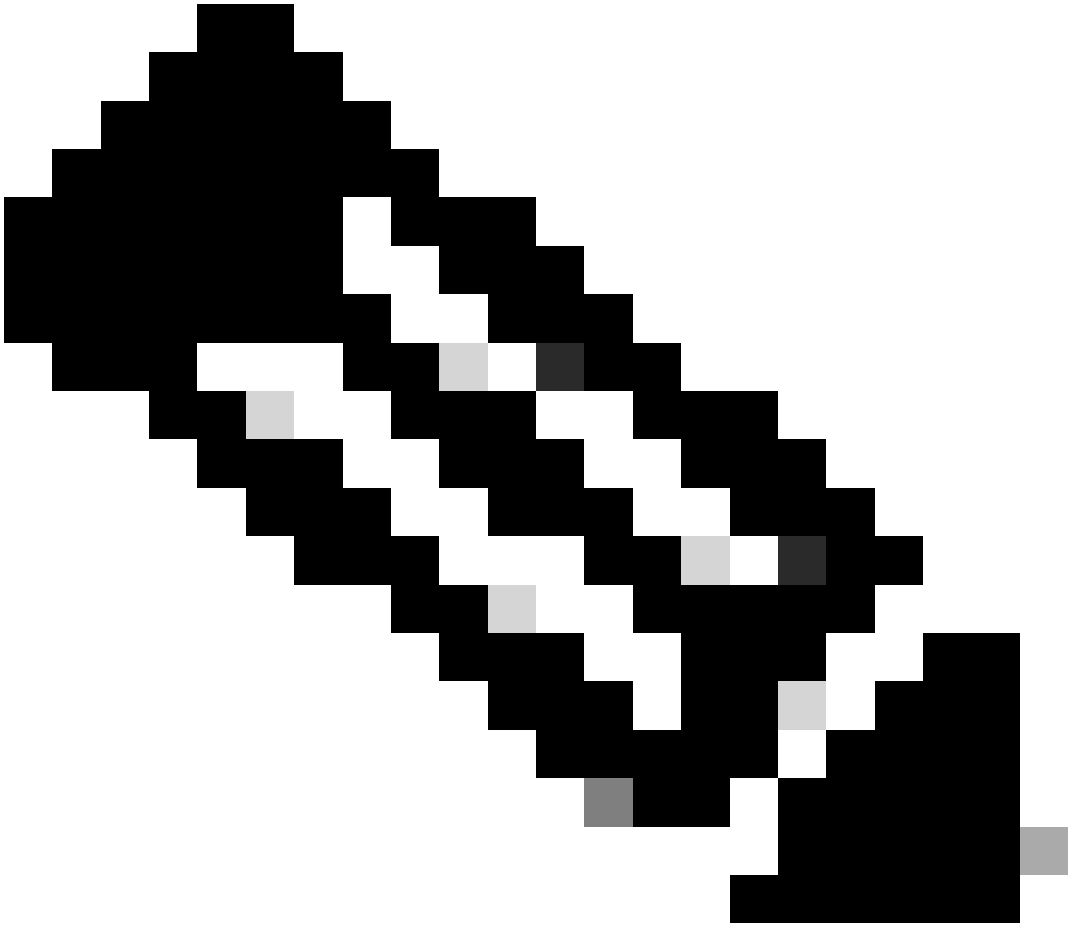


Sicherer Zugriff - API-Schlüsseldetails

5. Kopieren Sie die API Keyund die **Key Secret** und klicken Sie dann auf ACCEPT AND CLOSE:



Sicherer Zugriff - API-Schlüssel und Schlüssel



Hinweis: Es gibt nur eine Möglichkeit, Ihren API-Schlüssel zu kopieren. Secure Access speichert Ihren API-Schlüssel nicht, und Sie können ihn nach der ersten Erstellung nicht abrufen.

Python-Code

Es gibt mehrere Möglichkeiten, diesen Code zu schreiben, wenn man bedenkt, dass das generierte Token für 3600 Sekunden (1 Stunde) gültig ist. Sie können entweder zwei separate Skripts erstellen, in denen das erste Skript zum Generieren des Trägertokens verwendet werden kann, und dann ein zweites Skript, in dem das Trägertoken verwendet werden kann, um den API-Aufruf (Abrufen/Aktualisieren oder Löschen) an die interessierte Ressource durchzuführen, oder ein einzelnes Skript schreiben, um beide Aktionen auszuführen, wobei sichergestellt wird, dass, wenn bereits ein Trägertoken generiert wurde, eine Bedingung im Code beibehalten wird, dass nicht alle ein neues Trägertoken generiert wird Zeit, zu der das Skript ausgeführt wird.

Damit es in python funktioniert, stellen Sie bitte sicher, dass Sie diese Bibliotheken installieren:

```
pip install oauthlib pip install requests_oauthlib
```

Skript 1:

Erwähnen Sie client_secretin diesem Skript unbedingt die richtigen client_idund:

```
import requests from oauthlib.oauth2 import BackendApplicationClient from oauthlib.oauth2 import TokenE
```

Ausgabe:

Die Ausgabe dieses Skripts muss in etwa wie folgt aussehen:

```
Token: {'token_type': 'bearer', 'access_token': 'eyJhbGciOiJSUzI1NiIsImtpZCI6IjcyNmI5MGUzLWxxxxxxxxxxxxx
```

Der access_tokenist sehr lang mit Tausenden von Zeichen und wurde daher, um die Ausgabe lesbar zu halten, nur für dieses Beispiel gekürzt.

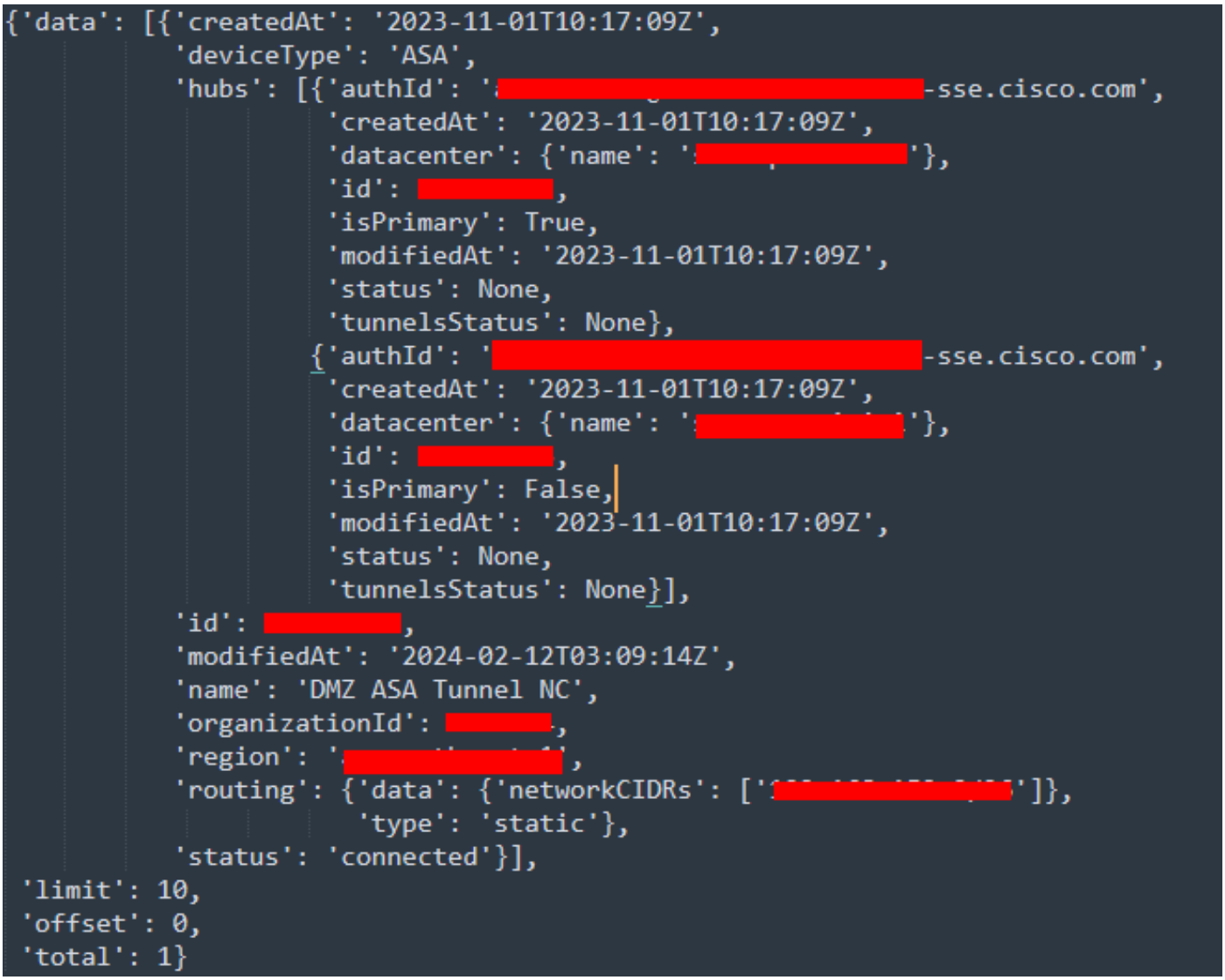
Skript 2:

Das access_token aus Skript 1 kann dann in diesem Skript für API-Aufrufe verwendet werden. Verwenden Sie beispielsweise Skript 2, um die Informationen zu den Netzwerk-Tunnelgruppen mit der Ressource /deployments/v2/networktunnelgroups abzurufen:

```
import requests import pprint import json url = "https://api.sse.cisco.com/deployments/v2/networktunnel
```

Ausgabe:

Die Ausgabe dieses Skripts muss in etwa wie folgt aussehen:



Python-Ausgabe - Netzwerk-Tunnelgruppen

Weitere Informationen zu Richtlinien, Roaming-Computern, Berichten usw. finden Sie im Secure Access Developers User Guide.

Fehlerbehebung

Die Endpunkte der API für sicheren Zugriff verwenden HTTP-Antwortcodes, um den Erfolg oder Misserfolg einer API-Anforderung anzuzeigen. Im Allgemeinen weisen Codes im 2xx-Bereich auf Erfolg hin, Codes im 4xx-Bereich auf einen Fehler hin, der sich aus den bereitgestellten Informationen ergibt, und Codes im 5xx-Bereich auf Serverfehler. Der Ansatz zur Behebung des Problems hängt vom empfangenen Antwortcode ab:

200	**OK**	Success. Everything worked as expected.
201	**Created**	New resource created.
202	**Accepted**	Success. Action is queued.
204	**No Content**	Success. Response with no message body.
400	**Bad Request**	Likely missing a required parameter or malformed JSON. The syntax of your query may need to be revised. Check for any spaces preceding, trailing, or in the domain name of the domain you are trying to query.
401	**Unauthorized**	The authorization header is missing or the key and secret pair is invalid. Ensure your API token is valid.
403	**Forbidden**	The client is unauthorized to access the content.
404	**Not Found**	The requested resource doesn't exist. Check the syntax of your query or ensure the IP and domain are valid.
409	**Conflict**	The client requests that the server create the resource, but the resource already exists in the collection.
429	**Exceeded Limit**	Too many requests received in a given amount of time. You may have exceeded the rate limits for your organization or package.
413	**Content Too Large**	The request payload is larger than the limits defined by the server.

REST API - Antwortcodes 1

500	**Internal Server Error**	Something wrong with the server.
503	**Service Unavailable**	Server is unable to complete request.

REST API - Antwortcodes 2

Zugehörige Informationen

- Cisco Secure Access Benutzerhandbuch
- Technischer Support und Downloads von Cisco
- Hinzufügen von API-Schlüsseln für sicheren Zugriff
- Entwickler-Benutzerhandbuch

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.