Configurer l'accès sécurisé pour utiliser l'API REST avec Python

Table des matières

Introduction

Ce document décrit les étapes pour configurer l'accès à l'API et l'utiliser pour récupérer des informations de ressources à partir de l'accès sécurisé.

Conditions préalables

Cisco vous recommande de prendre connaissance des rubriques suivantes :

1. Python 3.x
2. API REST
3. Accès sécurisé Cisco

Exigences

Ces exigences doivent être remplies avant de poursuivre :

- Compte utilisateur Cisco Secure Access avec le rôle Administrateur complet.
- Compte Cisco Security Cloud Single Sign On (SCSO) pour se connecter à Secure Access.

Composants utilisés

Les informations contenues dans ce document sont basées sur les versions de matériel et de logiciel suivantes :

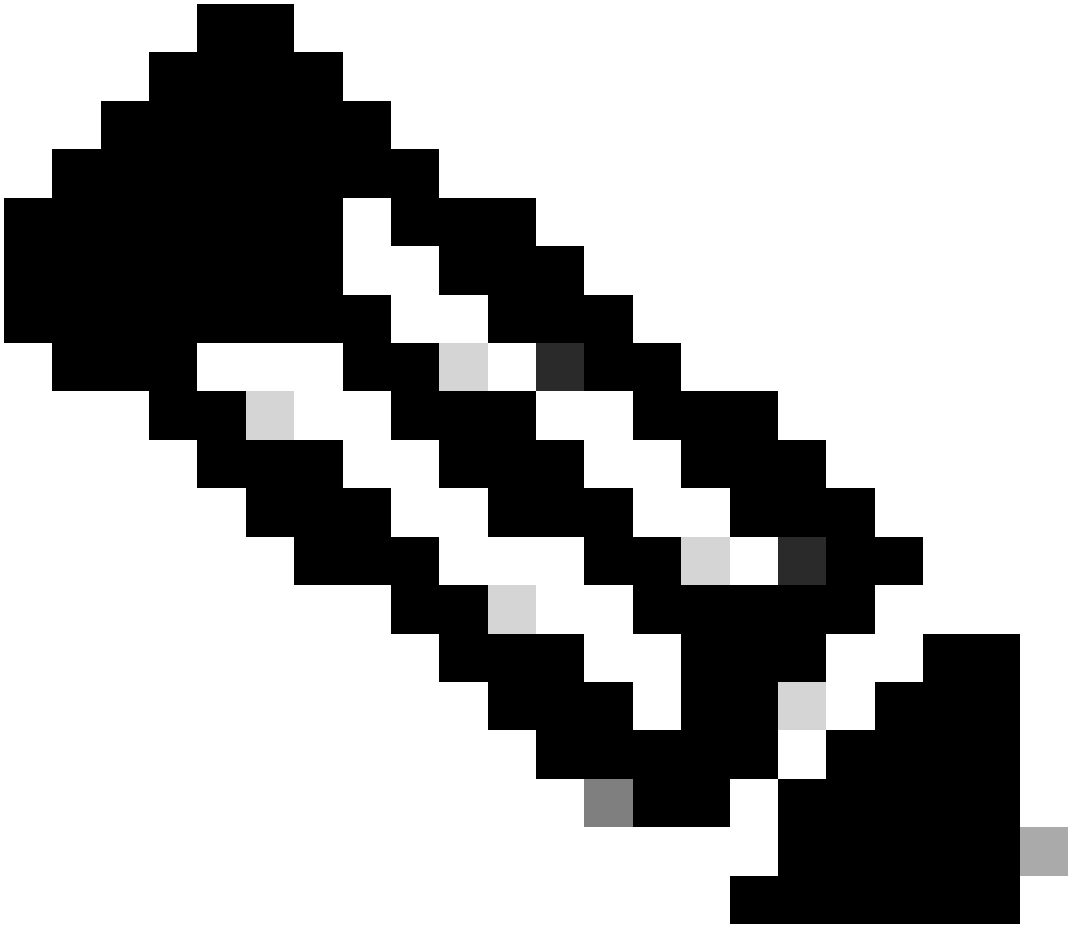
- Tableau de bord Secure Access

- Python

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. Si votre réseau est en ligne, assurez-vous de bien comprendre l’incidence possible des commandes.

Configurer

L'API d'accès sécurisé fournit une interface REST standard et prend en charge le flux d'informations d'identification du client OAuth 2.0. Pour commencer, connectez-vous à Secure Access et créez vos clés d'API Secure Access. Ensuite, utilisez vos informations d'identification API pour générer un jeton d'accès à l'API.



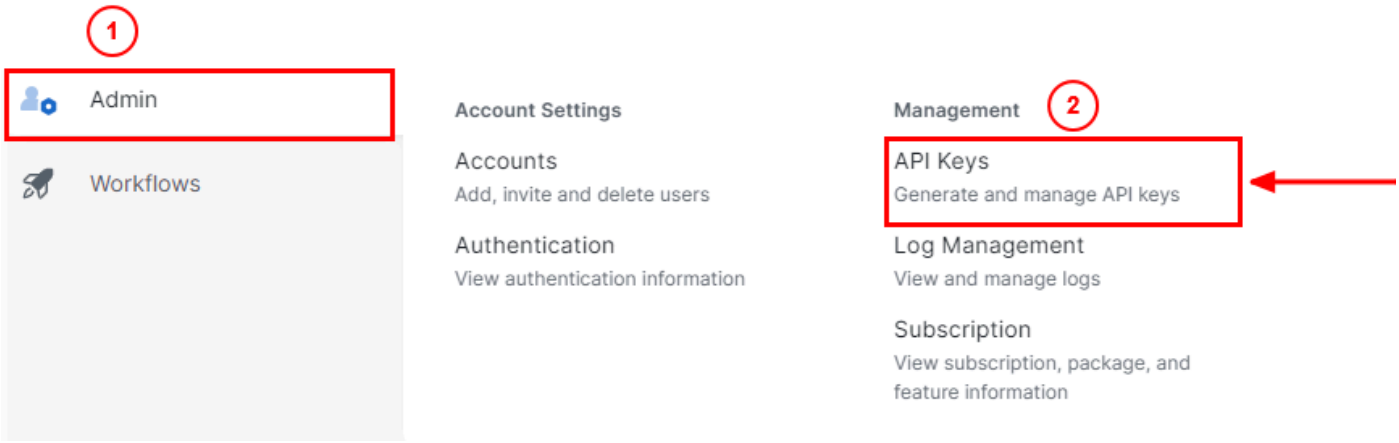
Remarque : les clés d'API, les mots de passe, les secrets et les jetons permettent d'accéder à vos données privées. Vous ne devez jamais partager vos informations d'identification avec un autre utilisateur ou une autre organisation.

Configurez la clé API à partir du tableau de bord Secure Access avant d'exécuter les scripts mentionnés dans cet article.

Créer une clé API

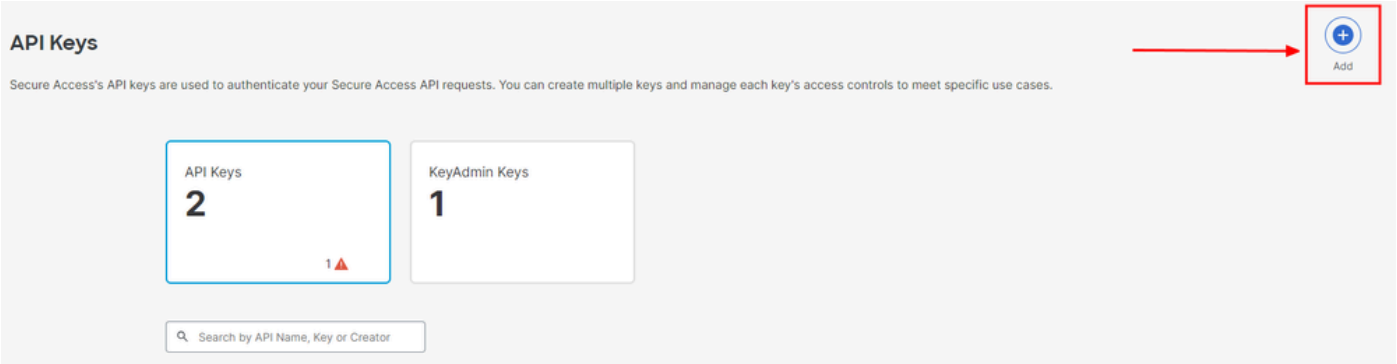
Créez une clé et un secret API en procédant comme suit. Connectez-vous à Secure Access avec l'URL : [Secure Access]

1. Dans la barre latérale gauche, sélectionnez l'option Admin.
   - Sous Admin sélectionner l'option **API Keys**:



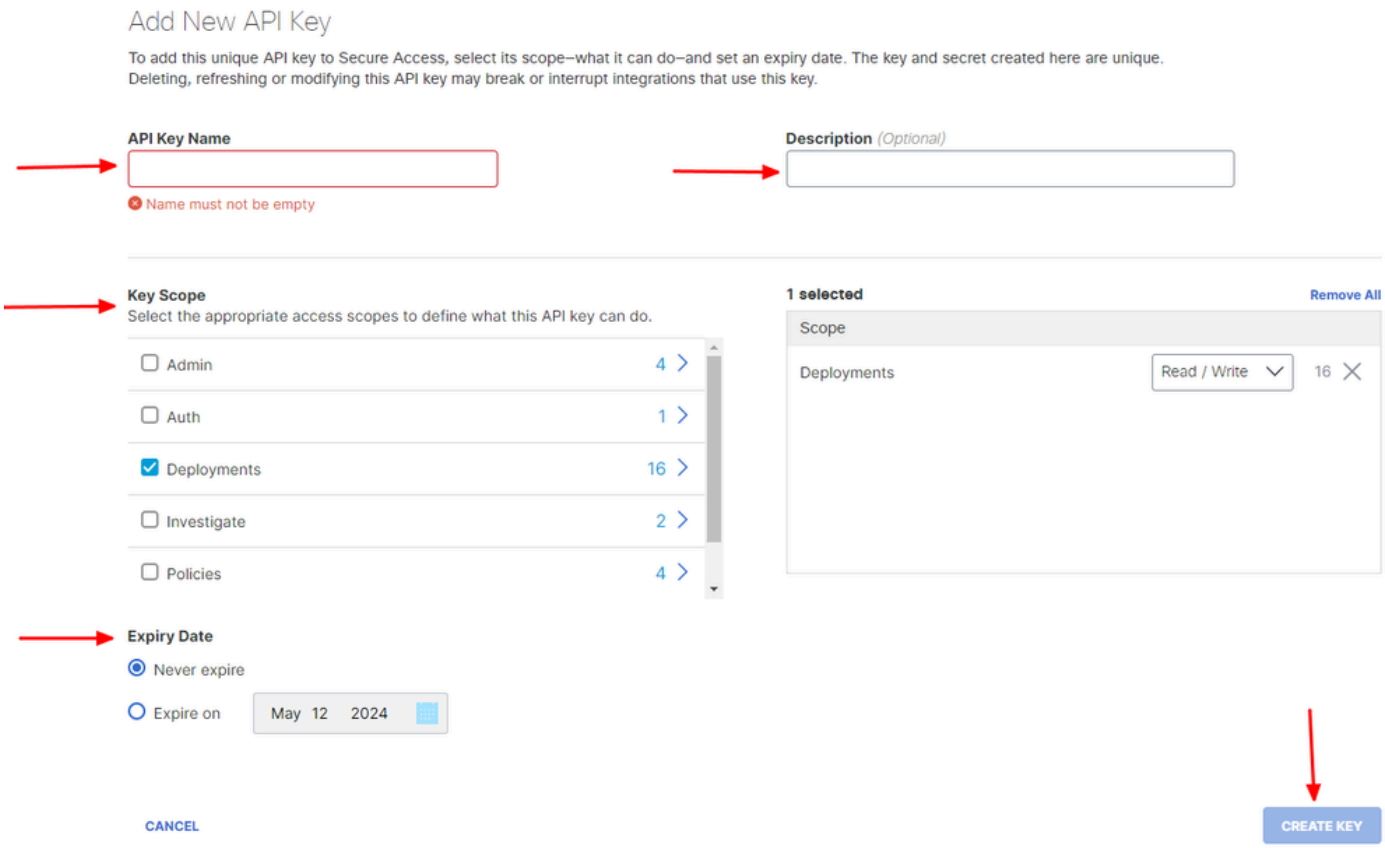
Administrateur du tableau de bord d'accès sécurisé - Clés API

3. Dans le coin supérieur droit, cliquez sur le + bouton Ajouter une nouvelle clé API :



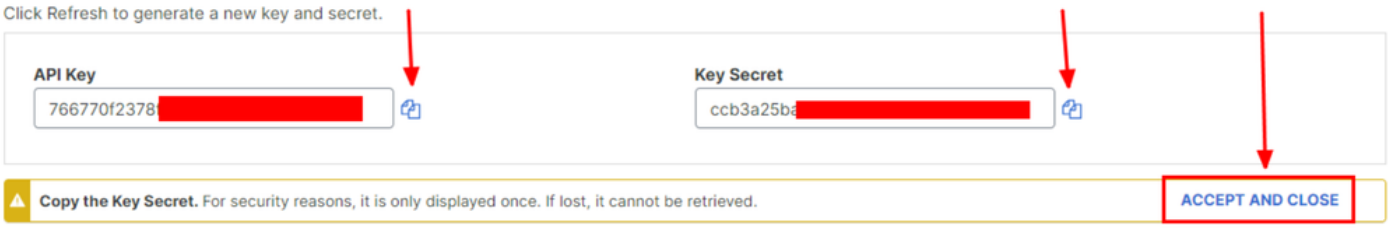
Accès sécurisé - Ajouter une clé API

4. Fournissez le **API Key Name, Description**(facultatif), puis sélectionnez le Key scope et Expiry date selon vos besoins. Une fois terminé, cliquez sur le bouton **Create**:

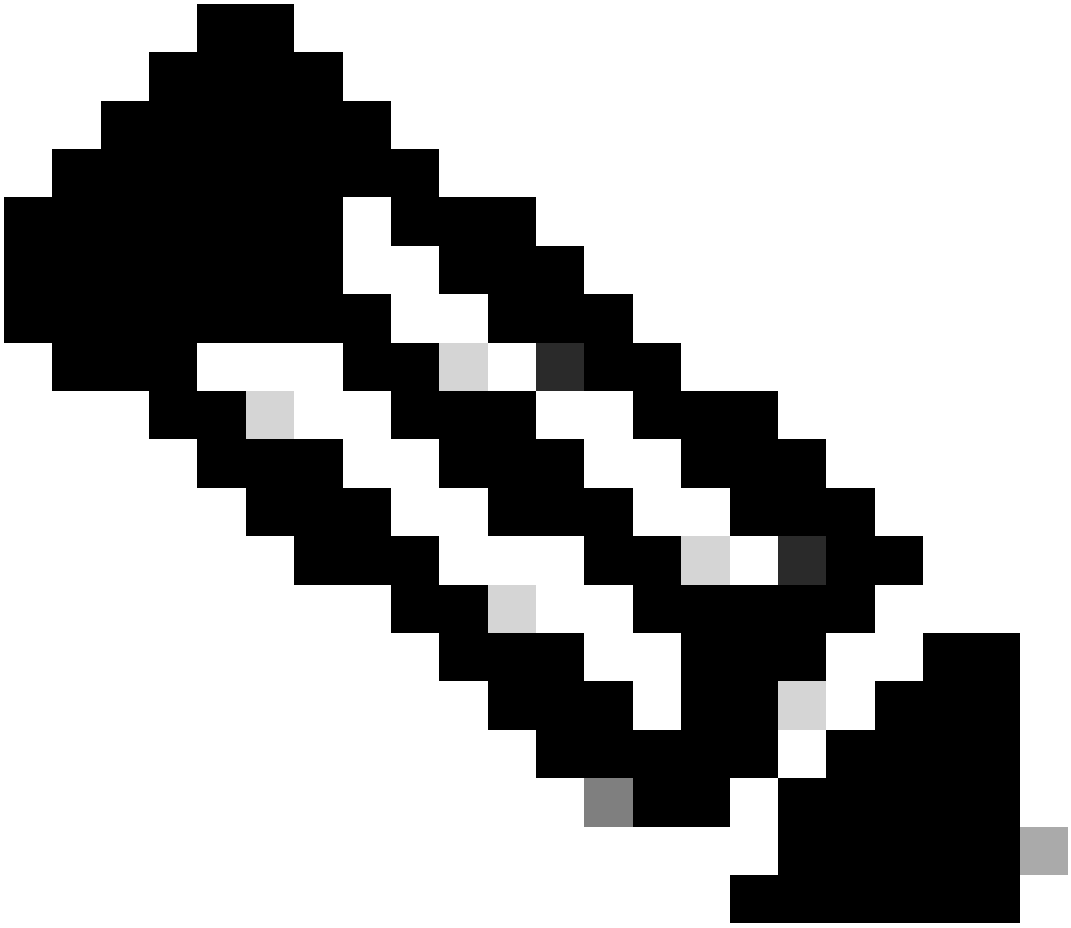


Accès sécurisé - Détails de la clé API

5. Copiez le API Keyet le **Key Secret** et cliquez sur ACCEPT AND CLOSE:



Accès sécurisé : clé API et secret



Remarque : il n'y a qu'une seule opportunité de copier votre secret API. Secure Access n'enregistre pas votre secret API et vous ne pouvez pas le récupérer après sa création initiale.

Code Python

Il existe plusieurs façons d'écrire ce code, étant donné que le jeton généré est valide pendant 3 600 secondes (1 heure). Vous pouvez soit créer deux scripts distincts dans lesquels le premier script peut être utilisé pour générer le jeton de support, puis un second script dans lequel ce jeton de support peut être utilisé pour effectuer l'appel API (extraction/mise à jour ou suppression) à la ressource qui vous intéresse, soit écrire un script unique pour effectuer les deux actions tout en s'assurant que si un jeton de support est déjà généré, une condition est conservée dans le code qu'un nouveau jeton de support ne soit pas généré chaque fois que le script est exécuté.

Afin de le faire fonctionner en python, veuillez vous assurer d'installer ces bibliothèques :

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

Script 1 :

Assurez-vous de mentionner le bon client_idet client_secretdans ce script :

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

Sortie :

Le résultat de ce script doit ressembler à ceci :

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

Le access_tokenest très long avec des milliers de caractères et, par conséquent, pour garder la sortie lisible, il a été raccourci juste pour cet exemple.

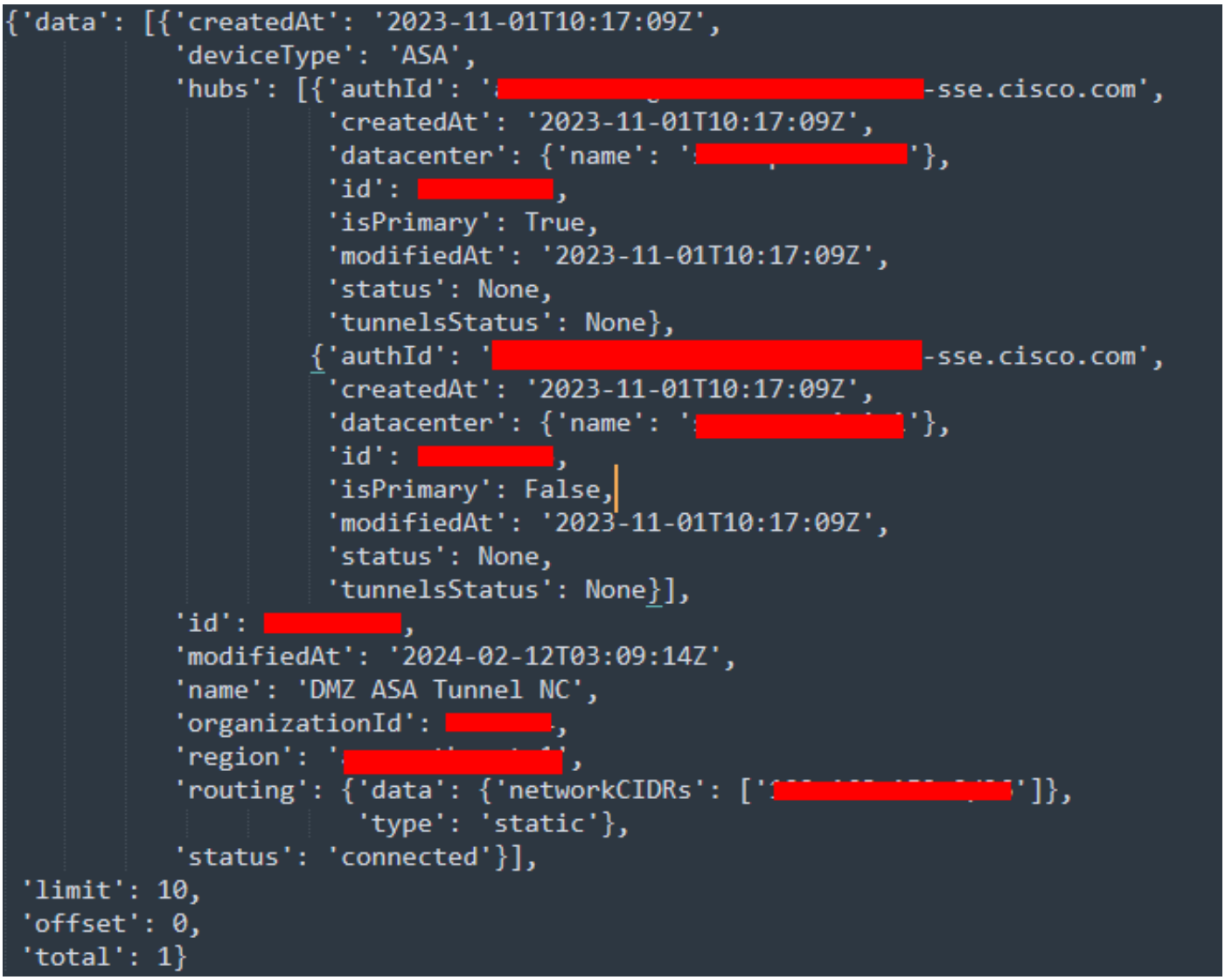
Script 2 :

Le scriptaccess_token du script 1 peut ensuite être utilisé dans ce script pour effectuer des appels d'API. Par exemple, utilisez le script 2 pour récupérer les informations sur les groupes de tunnels réseau à l'aide de la ressource /deployments/v2/networktunnelgroups:

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

Sortie :

Le résultat de ce script doit ressembler à ceci :



Sortie Python - Groupes de tunnels réseau

Vous pouvez également rechercher des informations sur les stratégies, les ordinateurs en itinérance, les rapports, etc., à l'aide du Guide de l'utilisateur Secure Access Developers.

Dépannage

Les points de terminaison de l'API d'accès sécurisé utilisent des codes de réponse HTTP pour indiquer la réussite ou l'échec d'une requête API. En général, les codes de la plage 2xx indiquent la réussite, les codes de la plage 4xx indiquent une erreur qui résulte des informations fournies et les codes de la plage 5xx indiquent des erreurs de serveur. L'approche à adopter pour résoudre le problème dépend du code de réponse reçu :

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.

API REST - Codes de réponse 1

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

API REST - Codes de réponse 2

Informations connexes

- Guide de l'utilisateur Cisco Secure Access
- Assistance technique de Cisco et téléchargements
- Ajouter des clés d'API Secure Access
- Guide de l'utilisateur développeur

À propos de cette traduction
Cisco a traduit ce document en traduction automatisée vérifiée par une personne dans le cadre d'un service mondial permettant à nos utilisateurs d'obtenir le contenu d'assistance dans leur propre langue.

Il convient cependant de noter que même la meilleure traduction automatisée ne sera pas aussi précise que celle fournie par un traducteur professionnel.