Gestion des listes de destinations d'accès sécurisé avec Python et l'API REST

Table des matières

Introduction

Conditions préalables

- Exigences
- Composants utilisés

Configurer

- Script

Erreurs

Dépannage

Informations connexes

Déflecteur (Surligner pour lire)
Veuillez noter que Cisco ne fournit pas d'assistance officielle pour cette conception de développement. Il s'agit uniquement d'un exemple de référence destiné à faciliter la compréhension de la manière dont l'API s'interface avec les applications. Les utilisateurs doivent utiliser cette conception à des fins éducatives uniquement et non comme base pour une mise en oeuvre au niveau de la production. L' exécution du code présenté dans cet article est à vos propres risques et Cisco décline expressément toute responsabilité pour tout problème découlant de son utilisation.
Veuillez noter que Cisco ne fournit pas d'assistance officielle pour cette conception de développement. Il s'agit uniquement d'un exemple de référence destiné à faciliter la compréhension de la manière dont l'API s'interface avec les applications. Les utilisateurs doivent utiliser cette conception à des fins éducatives uniquement et non comme base pour une mise en oeuvre au niveau de la production. L'exécution du code présenté dans cet article est à vos propres risques et Cisco décline expressément toute responsabilité pour tout problème découlant de son utilisation.

Introduction

Ce document décrit comment effectuer toutes les opérations possibles sur les listes de destination en utilisant l'API Python et REST.

Conditions préalables

Cisco vous recommande de prendre connaissance des rubriques suivantes :

1. Python
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.
- Créez vos clés d'API 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 3.x

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

Il existe plusieurs façons d'écrire ce code en tenant compte de plusieurs aspects tels que la gestion des erreurs, la validité du jeton (3 600 secondes), etc.

Assurez-vous que ces bibliothèques Python sont installées avant d'exécuter le script :

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

Script

Assurez-vous de remplacer les`client_id`et`client_secret`par vos`API Key`et`Key Secret`dans ce script, respectivement.

```
from oauthlib.oauth2 import BackendApplicationClient
from oauthlib.oauth2 import TokenExpiredError
from requests_oauthlib import OAuth2Session
from requests.auth import HTTPBasicAuth
import time
import requests
import pprint
import json
```

```
def fetch_headers(BToken):
    BT = f"Bearer {BToken}"
    headers = { 'Authorization':BT,
            "Content-Type": "application/json",
            "Accept": "application/json"
            }
    return headers

# GET OAUTH 2.0 TOKEN
def getToken():
    token_url = 'https://api.sse.cisco.com/auth/v2/token'
    try:
        #ASSIGN your API Key to the variable client_id and Secret Key to the variable client_secret
        client_id = "XXXXXXXXXXXXXXXXXXXXXXXXXXXX"
        client_secret = "XXXXXXXXXXXXXXXXXXXXXXXXXXXX"

        auth = HTTPBasicAuth(client_id, client_secret)
        client = BackendApplicationClient(client_id=client_id)
        oauth = OAuth2Session(client=client)
        token = oauth.fetch_token(token_url=token_url, auth=auth)
        print("\n######Token Generated Successfully######\n")
        return token
    except e as Exception:
        print(f"Encountered an error while Fetching the TOKEN :: {e}")


# 1 - GET DESTINATION LISTS
def fetch_destinationlists(h):
    url = "https://api.sse.cisco.com/policies/v2/destinationlists"
    try:
        response = requests.request('GET', url, headers=h)
        json_object = json.loads(response.content)

        #pprint.pprint(json_object)
        x=1
        for item in json_object["data"]:
            print(f"Destination List : {x}")
            pprint.pprint(f"Name : {item['name']}")
            pprint.pprint(f"ID : {item['id']}")
            #pprint.pprint(f"Destination Count : {item['meta']['destinationCount']}")
            print("\n")
            x+=1
    except e as Exception:
        print(f"Encountered an Error while Fetching the Destination Lists :: {e}")


# 2 - GET DESTINATION LIST
def get_destinationlist(h):
    try:
        choice = input("Enter the ID of the DestinationList:: ")
        url = "https://api.sse.cisco.com/policies/v2/destinationlists/" + choice
        response = requests.request('GET', url, headers=h)
        json_object = json.loads(response.content)
        print("\n\n")
        pprint.pprint(json_object)
        print("\n\n")
    except e as Exception:
        print(f"Encountered an Error while Fetching the Destination List Details :: {e}")


# 3 - CREATE DESTINATION LIST
def create_destinationlist(h):
```

```
    url = "https://api.sse.cisco.com/policies/v2/destinationlists"
    try:
        naav = input("Name of the DestinationList :: ")
        payload = {
            "access": "none",
            "isGlobal": False,
            "name": naav,
            }
        response = requests.request('POST', url, headers=h, data = json.dumps(payload))
        json_object = json.loads(response.content)
        print("\n\n")
        pprint.pprint(json_object)
        print("\n\n")
    except e as Exception:
        print(f"Encountered an Error while Creating the Destination List :: {e}")


# 4 - UPDATE DESTINATION LIST NAME
def patch_destinationlist(h):
    try:
        choice = input("Enter the ID of the DestinationList for changing it's name :: ")
        url = "https://api.sse.cisco.com/policies/v2/destinationlists/" + choice
        naav = input("Enter New Name :: ")
        payload = {"name": naav}

        response = requests.request('PATCH', url, headers=h, data = json.dumps(payload))
        json_object = json.loads(response.content)
        print("\n\n")
        pprint.pprint(json_object)
        print("\n\n")
    except e as Exception:
        print(f"Encountered an Error while Updating the Destination List :: {e}")


# 5 - DELETE DESTINATION LIST
def delete_destinationlist(h):
    try:
        choice = input("Enter the ID of the DestinationList for DELETION :: ")
        url = "https://api.sse.cisco.com/policies/v2/destinationlists/" + choice

        response = requests.request('DELETE', url, headers=h)
        json_object = json.loads(response.content)
        print("\n\n")
        pprint.pprint(json_object)
        print("\n\n")
    except Exception as e:
        print(f"Encountered an Error while Deleting the Destination List :: {e}")


# 6 - GET DESTINATIONS FROM A DESTINATION LIST
def fetch_detail(h):
    try:
        choice = input("DestinationList ID: ")

        url2 = "https://api.sse.cisco.com/policies/v2/destinationlists/" + choice + "/destinations"

        response = requests.request('GET', url2, headers=h)
        print("\n")
        json_dest = json.loads(response.content)
        pprint.pprint(json_dest)
        print("\n\n")
    except e as Exception:
```

```
        print(f"Encountered an Error while Fetching the Destinations from the Destination List :: {e}")


# 7 - ADD DESTINATIONS TO A DESTINATION LIST
def add_destinations(h):
    try:
        choice = input("Enter the ID of the DestinationList :: ")
        url = "https://api.sse.cisco.com/policies/v2/destinationlists/" + choice + "/destinations"
        destination_to_add = input("\nEnter the destination that you want to add :: ")
        payload = [{"destination": destination_to_add}]

        response = requests.request('POST', url, headers=h, data = json.dumps(payload))
        print("\n")
        json_dest = json.loads(response.content)
        pprint.pprint(json_dest)
        print("\n\n")
    except e as Exception:
        print(f"Encountered an Error while Adding the Destination to the Destination List :: {e}")


# 8 - DELETE DESTINATIONS FROM DESTINATION LIST
def delete_entry(h):
    try:
        choice_del = input("\nCONFIRM DestinationList ID from which you want to delete the Destination
        url3 = "https://api.sse.cisco.com/policies/v2/destinationlists/" + choice_del + "/destinations/
        dest = int(input("ID of the Destination that you want to remove: "))

        payload = f"[{dest}]"
        response = requests.request('DELETE', url3, headers=h, data=payload)
        json_del = json.loads(response.content)
        print("\n\n")
        pprint.pprint(json_del)
        print("\n\n")
    except e as Exception:
        print(f"Encountered an Error while Deleting a Destination from the Destination List :: {e}")



#FETCH COOKIE
possess_cookie = " "
while possess_cookie not in ["Y","y","N","n"]:
    possess_cookie = input("Token Already Generated? (Y/N) :: ")
    if possess_cookie.upper() =="N":
        cook = getToken()
        with open("cookie.txt","w") as wr:
            wr.writelines(cook["access_token"])
#         print(f"Access Token = {cook["access_token"]}")


#FETCH HEADERS
with open("cookie.txt","r") as ree:
    h = fetch_headers(ree.readline())


print("\n")
while True:
    action = input("""Available operations:
1. Get Destination Lists
2. Get Destination List
3. Create Destination List
4. Update Destination List Name
5. Delete Destination List
```

```
6. Get Destinations from Destination List
7. Add Destinations to a Destination List
8. Delete Destinations from Destination List
9. Exit

Enter Your Choice :: """)

    print("\n")
    operation = action.replace(" ","")
    if operation == "1":
        fetch_destinationlists(h)

    elif operation == "2":
        fetch_destinationlists(h)
        get_destinationlist(h)

    elif operation == "3":
        create_destinationlist(h)

    elif operation == "4":
        fetch_destinationlists(h)
        patch_destinationlist(h)

    elif operation == "5":
        fetch_destinationlists(h)
        delete_destinationlist(h)

    elif operation == "6":
        fetch_destinationlists(h)
        fetch_detail(h)

    elif operation == "7":
        fetch_destinationlists(h)
        add_destinations(h)

    elif operation == "8":
        fetch_destinationlists(h)
        fetch_detail(h)
        delete_entry(h)

    elif operation == "9":
        break

    else:
        print("\n")
        print("==========INCORRECT INPUT==========")
        print("\n")

print("Thank You!!! Good Bye!!!")
time.sleep(5)
```

Sortie :

Le résultat de ce script doit ressembler à ceci :

```
Token Already Generated? (Y/N) :: y
```

```
Available operations:
1. Get Destination Lists
2. Get Destination List
3. Create Destination List
4. Update Destination List Name
5. Delete Destination List
6. Get Destinations from Destination List
7. Add Destinations to a Destination List
8. Delete Destinations from Destination List
9. Exit

Enter Your Choice :: 1
```

Une fois que ce programme est exécuté avec succès, une question est posée au début sur le cookie -Cookie Already Generated? (Y/N). La raison de cette question est de vous assurer que vous ne générez pas le cookie plusieurs fois, car un cookie, une fois généré, est valide pendant 3600 secondes (1 heure). Si vous entrezyouY, aucun nouveau cookie n'est généré. Cependant, si vous entreznouN, un nouveau cookie est généré et enregistré dans un fichier texte local dans le même répertoire/dossier. Le cookie de ce fichier est utilisé dans vos demandes ultérieures.

Erreurs

Vous pouvez rencontrer cette erreur si vous entrez un ID incorrect pour une opération qui vous oblige à mentionner l'ID DestinationList :

```
{'message': 'no Route matched with those values'}
```

Lors de la création d'un DestinationList, si vous mentionnez le nom de DestinationList qui dépasse 255 caractères, vous voyez cette erreur :

```
{'code': 400,
'code_text': 'Bad Request',
'error': 'invalid_request',
'message': {'name': {'code': 'string_invalid',
'code_text': 'must be fewer than 255 characters long'}},
'statusCode': 400,
'txId': 'l23f0cjk62fe'}
```

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
- Configurer Secure Access pour utiliser l'API REST avec Python
- Gérer les listes de destinations via cURL

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.