Manage Secure Access Destination Lists Using Python and REST API

Contents

Spoiler (Highlight to read)
Please be advised that Cisco does not provide official support for this development design. It is intended solely as a reference example to facilitate understanding of how the API interfaces with applications. Users must employ this design for educational purposes only and not as a basis for production-level implementation. Executing the code presented in this article is at your own risk, and Cisco expressly disclaims any liability for any issues arising from its use.
Please be advised that Cisco does not provide official support for this development design. It is intended solely as a reference example to facilitate understanding of how the API interfaces with applications. Users must employ this design for educational purposes only and not as a basis for production-level implementation. Executing the code presented in this article is at your own risk, and Cisco expressly disclaims any liability for any issues arising from its use.

Introduction

This document describes how to perform all possible operations on Destination Lists using Python and REST API.

Prerequisites

Cisco recommends that you have knowledge of these topics:

1. Python
2. REST API
3. Cisco Secure Access

Requirements

These requirements must be fulfilled before proceeding further:

- Cisco Secure Access user account with the**Full Admin**user role.
- Cisco Security Cloud Single Sign On (SCSO) account to sign in to Secure Access.
- Create your Secure Access API keys

Components Used

The information in this document is based on these software and hardware versions:

- Secure Access Dashboard
- 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. If your network is live, ensure that you understand the potential impact of any command.

Configure

There are multiple ways to write this code considering multiple aspects like Error Handling, token validity (3600 seconds), and so on.

Kindly make sure that these Python libraries are installed before running the script:

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

Script

Kindly make sure to substitue the client_id and the client_secret with your API Key and Key Secret in this script, respectively.

```
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}")
```

```python
# 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)
```

Output:

The output from this script must look something like this:

```
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
```

Once this program is successfully executed, a question is being asked at the beginning about the Cookie -

Cookie Already Generated? (Y/N). The reason to ask this question is to make sure that you do not generate the cookie multiple times because a cookie, once generated, is valid for 3600 seconds (1 hour). If you enter y or Y, a new cookie is not generated. However, if you enter nor N, a new cookie is generated and saved to a local text file in the same directory/folder. The cookie from this file is used in your subsequent requests.

Errors

You can encounter this error if you enter an incorrect ID for any operation that requires you to mention the DestinationList ID:

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

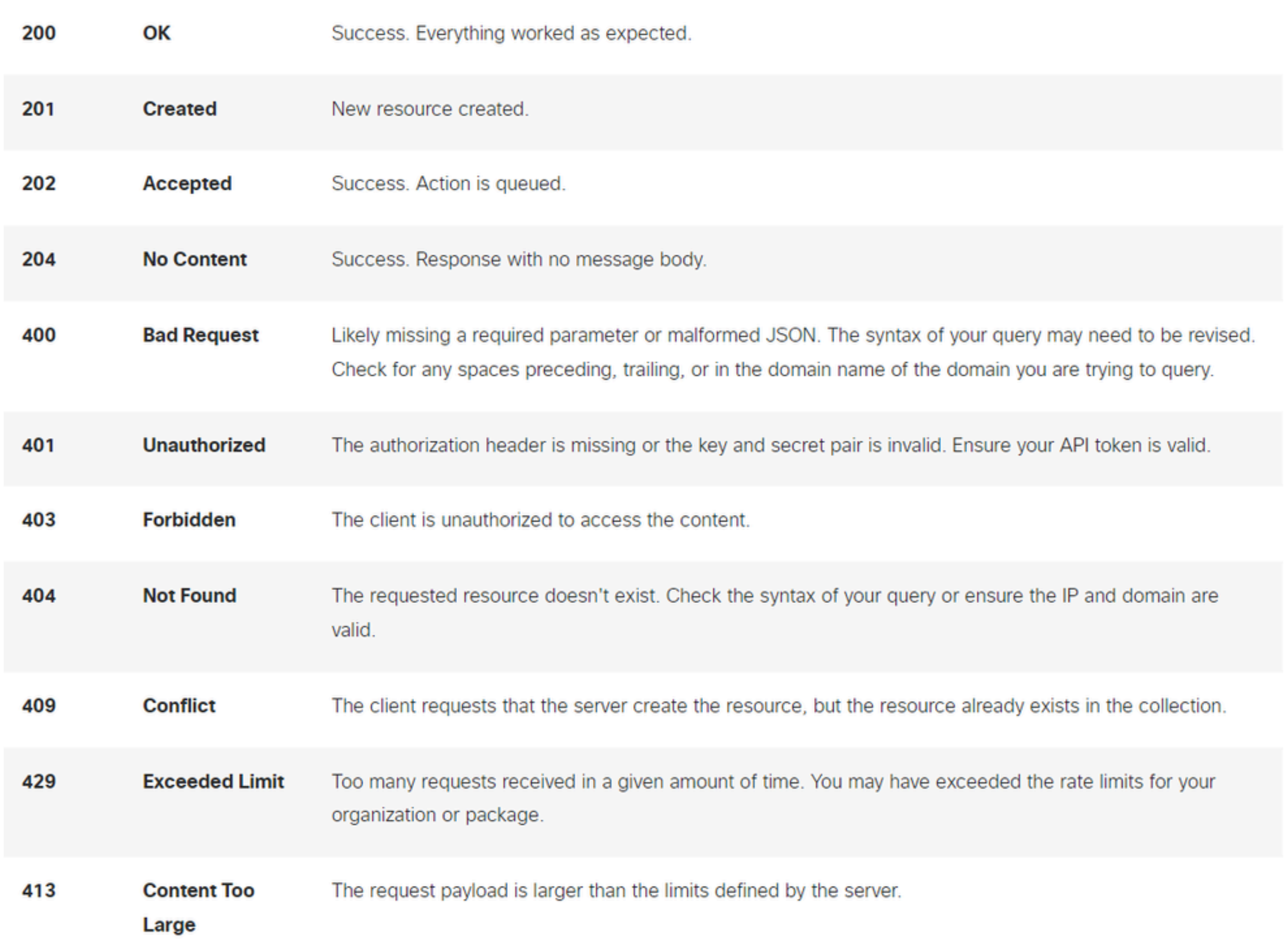
While creating a DestinationList, if you mention the name of DestinationList that exceeds 255 characters, you see this error:

```
{'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': '123f0cjk62fe'}
```

You can also fetch information about Policies, Roaming Computers, Reports, and so on, with the Secure Access Developers User Guide.

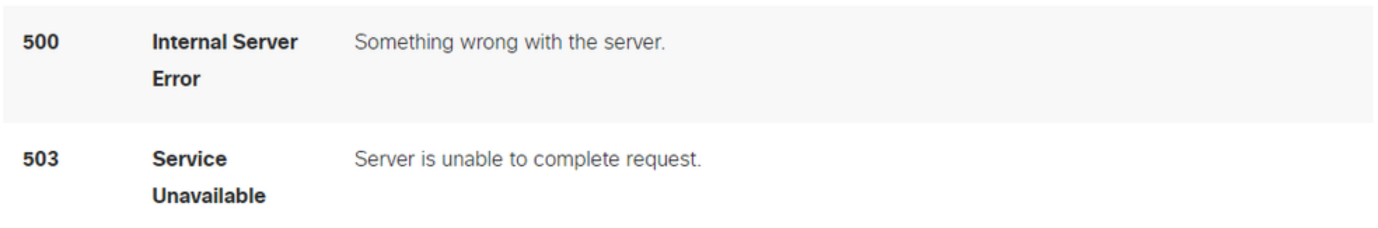
Troubleshoot

The Secure Access API endpoints use HTTP response codes to indicate success or failure of an API request. In general, codes in the 2xx range indicate success, codes in the 4xx range indicate an error that resulted from the provided information, and codes in the 5xx range indicate server errors. The approach to resolve the issue would depend on the response code that is received:



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 - Response codes 1



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

REST API - Response codes 2

Related Information

- [Cisco Secure Access User Guide]
- [Cisco Technical Support and Downloads]
- [Add Secure Access API Keys]
- [Developers User Guide]
- [Configure Secure Access to use REST API with Python]
- [Manage Destination Lists via cURL]