Create Python Script for Rest API Calls to FDM

Contents

Introduction

Prerequisites

- Requirements
- Components Used

Background Information

Configure

- GET
- POST
- PUT
- DELETE
- DEPLOY

Verify

- GET
- PUT
- DELETE

Troubleshoot

Related Information

Introduction

This document describes an example of using Python to make Rest API calls.

Prerequisites

The tools and devices used in the guide are:

- Cisco Firepower Threat Defense (FTD)
- Cisco Firepower Device Management (FDM)
- Mac OS
- Sublime Text

Requirements

Cisco recommends that you have knowledge of these topics:

- HTTPS
- Rest API
- Python
- Json
- Firepower Device Management
- LDAP

Components Used

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

- Python 3.11.4
- FDM API version 6
- FTDv 7.3.1

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.

Background Information

The main intention of this document is to guide you through the steps for creating a Python script to make API calls.

The specific feature to be adjusted and built is LDAP and attribute maps.

The API calls made in the guide are:

GET: Collect information from the server

POST: Create a new object on the server

PUT: Update an existing object on the server

DELETE: Remove an existing object from the server

Configure

GET

Rest API GET example to collect data on existing objects:

```
import requests
import json
import certifi
import urllib3
from pprint import pprint
from getpass import getpass

urllib3.disable_warnings()

#Data collection
u = input ('Input username: ')
p = getpass(prompt='Input password: ')
url = input('Input Hostname or IP address: ')
protocol = 'https://'

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
def auth():

  uri = '/api/fdm/v6/fdm/token'
```

```
  token_url = protocol+url+uri

  payload = {
        'grant_type' : 'password',
        'username' : u,
        'password': p
        }

  response = requests.post(token_url, data=payload, verify=False)

  #ONlY USE "verify=False" in a lab setting!

#Error Checking
  if response.status_code == 400:
    raise Exception("Error Received: {}".format(response.content))
  else:
    access_token = response.json()['access_token']
    return access_token

token = auth()
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

def getactivedirectory():
  uri = "/api/fdm/v6/object/realms"
  ad_url = protocol+url+uri

  headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
    "Authorization":"Bearer {}".format(token)
  }

  response = requests.get(ad_url, headers=headers, verify=False)
  if response.status_code == 200:
    AD = response.json()
    return AD
  else:
    print(response.status_code)
    pass


pprint(getactivedirectory())

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

def revoke():

  uri = '/api/fdm/v6/fdm/token'
  token_url = protocol+url+uri

  headers = {
      "Content-Type": "application/json",
      "Accept": "application/json",
      "Authorization":"Bearer"
  }
  payload = {
        'grant_type' : 'revoke_token',
        "access_token": token,
        "token_to_revoke": token,
        }

  response = requests.post(token_url, data=payload, verify=False)
```

```
  if response.status_code == 200:
    print("Access token revoked")
  else:
    print(response.status_code)


revoke()
```

POST

This section describes an example of making a REST API POST request to create a new LDAP attribute map object:

```
import requests
import json
import certifi
import urllib3
from pprint import pprint
from getpass import getpass

urllib3.disable_warnings()

#Data collection
u = input ('Input username: ')
p = getpass(prompt='Input password: ')
url = input('Input Hostname or IP address: ')
protocol = 'https://'

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
def auth():

  uri = '/api/fdm/v6/fdm/token'
  token_url = protocol+url+uri

  payload = {
        'grant_type' : 'password',
        'username' : u,
        'password': p
        }

  response = requests.post(token_url, data=payload, verify=False)

  #WARNING ONLY USE "verify=False" in a lab setting!

#Error Checking
  if response.status_code == 400:
    raise Exception("Error Received: {}".format(response.content))
  else:
    access_token = response.json()['access_token']
    return access_token

token = auth()

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
def postattributemap():
  uri = "/api/fdm/v6/object/ldapattributemaps"
  ad_url = protocol+url+uri
```

```
  name = input('Object name> ')
  group_policy = input('Group-Policy> ')
  base = input('LDAP DN> ')

  headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
    "Authorization":"Bearer {}".format(token)
  }

  payload = {
  "name": name,
  "ldapAttributeMaps": [
    {
      "ldapName": "memberOf",
      "ciscoName": "GROUP_POLICY",
      "valueMappings": [
        {
          "ldapValue": base,
          "ciscoValue": group_policy,
          "type": "ldaptociscovaluemapping"
        }
      ],
      "type": "ldapattributemapping"
    }
  ],
    "type": "ldapattributemap"
  }
  data = json.dumps(payload)

  response = requests.post(ad_url, headers=headers, data=data, verify=False)

  if response.status_code == 200:
    print(response.status_code)
    print("Created LDAP attribute map")
    map = response.json
    return map
  else:
    print(response.status_code)
    pprint(response.content)
    pass



postattributemap()



#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

def revoke():

  uri = '/api/fdm/v6/fdm/token'
  token_url = protocol+url+uri

  headers = {
      "Content-Type": "application/json",
      "Accept": "application/json",
      "Authorization":"Bearer"
  }
```

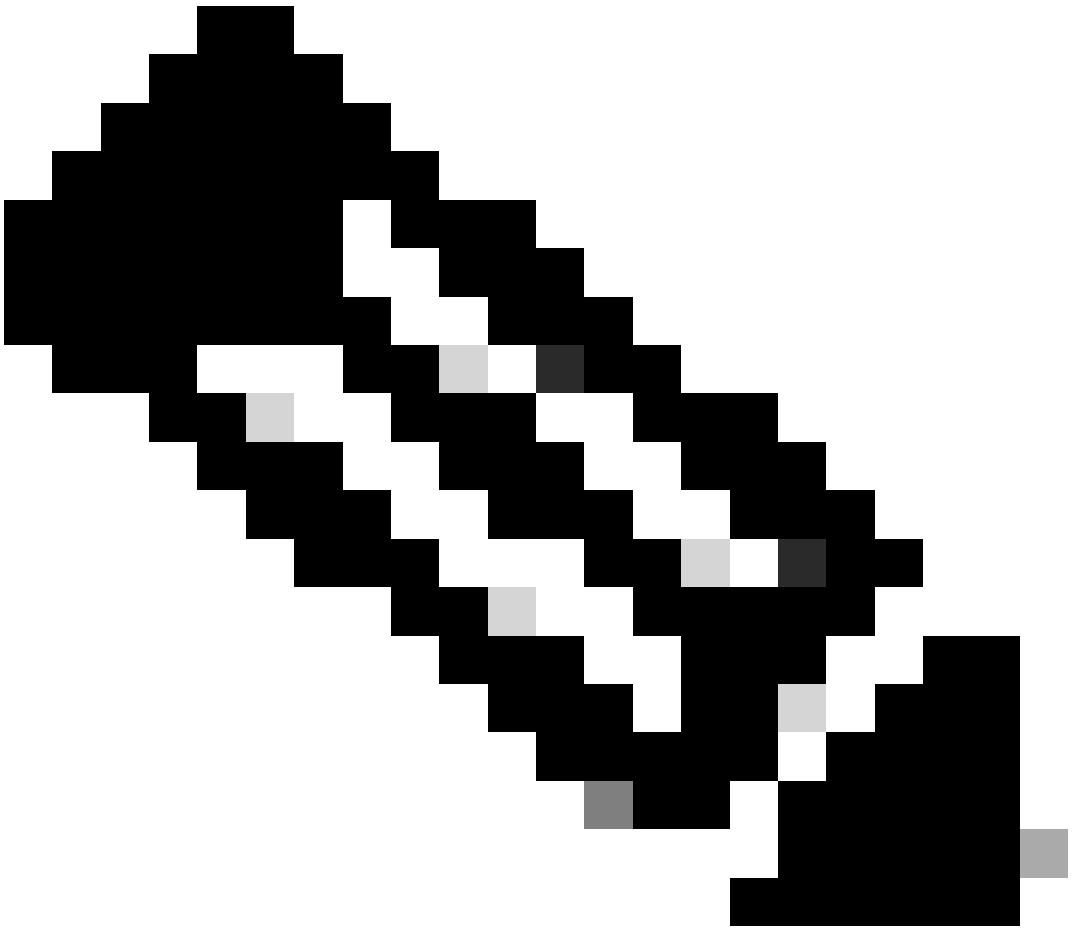
```
  payload = {
        'grant_type' : 'revoke_token',
        "access_token": token,
        "token_to_revoke": token,
        }

  response = requests.post(token_url, data=payload, verify=False)
  if response.status_code == 200:
    print("Access token revoked")
  else:
    print(response.status_code)


revoke()
```

PUT

This section shows an example of a Python script making a PUT Rest API call. This function adds the LDAP attribute map to the existing active directory configurations.



Note: Note: Before proceeding, the information needed to update the object must be collected via the GET function.

URL1 Active Directory realms: /api/fdm/v6/object/realms/

URL2 LDAP attribute map: /api/fdm/v6/object/ldapattributemaps

```
import requests
import json
import certifi
import urllib3
from pprint import pprint
from getpass import getpass

urllib3.disable_warnings()

#Data collection
u = input ('Input username: ')
p = getpass(prompt='Input password: ')
url = input('Input Hostname or IP address: ')
protocol = 'https://'

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
def auth():

  uri = '/api/fdm/v6/fdm/token'
  token_url = protocol+url+uri

  payload = {
        'grant_type' : 'password',
        'username' : u,
        'password': p
        }

  response = requests.post(token_url, data=payload, verify=False)

  #WARNING ONLY USE "verify=False" in a lab setting!

#Error Checking
  if response.status_code == 400:
    raise Exception("Error Received: {}".format(response.content))
  else:
    access_token = response.json()['access_token']
    return access_token

token = auth()
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

def put():
  objId = input('object id of active directory object> ')
  uri = "/api/fdm/v6/object/realms/"
  ad_url = protocol+url+uri+objId

  headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
    "Authorization":"Bearer {}".format(token)
  }
#Place the GET response from the active directory here in "payload" along with the added "ldapAttributeM
```

```
  payload = {
  "version": "p6ueo2w2aulkf",
  "name": "Test",
  "directoryConfigurations": [
    {
      "hostname": "{omitted}",
      "port": 389,
      "encryptionProtocol": "NONE",
      "encryptionCert": None,
      "interface": {
        "version": "mjvylmnd52agk",
        "name": "diagnostic",
        "hardwareName": "Management0/0",
        "id": "a46ef70c-06ca-11ee-9be1-bd712e622992",
        "type": "physicalinterface"
      },
      "type": "directoryconfiguration"
    }
  ],
  "enabled": True,
  "realmId": 3,
  "dirUsername": "{omitted}",
  "dirPassword": "*********",
  "baseDN": "dc={omitted},dc=com",
  "ldapAttributeMap": {
    "id": "7fbf5798-27c9-11ee-a635-a1f4b2c2e66b",
    "type": "ldapattributemap"
  },
  "adPrimaryDomain": "{omitted}.com",
  "id": "5957a304-2662-11ee-a635-a5df7d28e8c4",
  "type": "activedirectoryrealm"
  }

  data = json.dumps(payload)


  response = requests.put(ad_url, headers=headers, data=data, verify=False)

  if response.status_code == 200:
    print("Updated Object")
  else:
    pprint("Error Received: {}".format(response.content))
    pass

put()



#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

def revoke():

  uri = '/api/fdm/v6/fdm/token'
  token_url = protocol+url+uri

  headers = {
      "Content-Type": "application/json",
      "Accept": "application/json",
      "Authorization":"Bearer"
  }
  payload = {
        'grant_type' : 'revoke_token',
```

```
        "access_token": token,
        "token_to_revoke": token,
        }

  response = requests.post(token_url, data=payload, verify=False)

  if response.status_code == 200:
    print("Access token revoked")
  else:
    print(response.status_code)


revoke()
```

DELETE

This section describes a Rest API DELETE example to remove the active directory object:

```
import requests
import json
import certifi
import urllib3
from pprint import pprint
from getpass import getpass

urllib3.disable_warnings()

#Data collection
u = input ('Input username: ')
p = getpass(prompt='Input password: ')
url = input('Input Hostname or IP address: ')
protocol = 'https://'

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
def auth():

  uri = '/api/fdm/v6/fdm/token'
  token_url = protocol+url+uri

  payload = {
        'grant_type' : 'password',
        'username' : u,
        'password': p
        }

  response = requests.post(token_url, data=payload, verify=False)

  #ONlY USE "verify=False" in a lab setting!
```

```
#Error Checking
  if response.status_code == 400:
    raise Exception("Error Received: {}".format(response.content))
  else:
    access_token = response.json()['access_token']
    return access_token

token = auth()
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

def delete():
  objId = input('object id to delete> ')
  uri = "/api/fdm/v6/object/realms/"
  ad_url = protocol+url+uri+objId


  headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
    "Authorization":"Bearer {}".format(token)
  }

  response = requests.delete(ad_url, headers=headers, verify=False)


  if response.status_code == 204:
    print('Object removed')
  else:
    print("Error Received: {}".format(response.content))
    pass

delete()

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

def revoke():

  uri = '/api/fdm/v6/fdm/token'
  token_url = protocol+url+uri

  headers = {
      "Content-Type": "application/json",
      "Accept": "application/json",
      "Authorization":"Bearer"
  }
  payload = {
        'grant_type' : 'revoke_token',
        "access_token": token,
        "token_to_revoke": token,
        }

  response = requests.post(token_url, data=payload, verify=False)

  if response.status_code == 200:
    print("Access token revoked")
  else:
    print(response.status_code)


revoke()
```

DEPLOY

This section describes an example of deploying configuration changes via a POST REST API call and confirming the status of deployments with a GET request.

```
import requests
import json
import certifi
import urllib3
from pprint import pprint
from getpass import getpass

urllib3.disable_warnings()

#Data collection
u = input ('Input username: ')
p = getpass(prompt='Input password: ')
url = input('Input Hostname or IP address: ')
protocol = 'https://'


x="0"
attempts=0


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
def auth():

  uri = '/api/fdm/v6/fdm/token'
  token_url = protocol+url+uri

  payload = {
        'grant_type' : 'password',
        'username' : u,
        'password': p
        }

  response = requests.post(token_url, data=payload, verify=False)

  #ONlY USE "verify=False" in a lab setting!

#Error Checking
  if response.status_code == 400:
    raise Exception("Error Received: {}".format(response.content))
  else:
    access_token = response.json()['access_token']
    return access_token


token = auth()

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


def deploy():

  uri = '/api/fdm/v6/operational/deploy'
  deploy = protocol+url+uri
```

```
  headers = headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
    "Authorization":"Bearer {}".format(token)
  }

  payload = {
  "statusMessage": "string",
  "cliErrorMessage": "string",
  "state": "QUEUED",
  "queuedTime": 0,
  "startTime": 0,
  "endTime": 0,
  "statusMessages": [
    "string"
    ],
  "id": "string",
  "name": "string",
  "modifiedObjects": {},
  "forceRefreshDeploymentData": False,
  "type": "deploymentstatus"
  }

  data = json.dumps(payload)

  response = requests.get(deploy, headers=headers, data=data,verify=False)
  if response.status_code == 200:
    print(response.status_code)
    status = response.json()['items'][0]['statusMessage']
    return status
  else:
    print(response.status_code)
    pass

status = deploy()

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


def deploy2():

  uri = '/api/fdm/v6/operational/deploy'
  deploy = protocol+url+uri

  headers = headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
    "Authorization":"Bearer {}".format(token)
  }

  payload = {
  "statusMessage": "string",
  "cliErrorMessage": "string",
  "state": "QUEUED",
  "queuedTime": 0,
  "startTime": 0,
  "endTime": 0,
  "statusMessages": [
    "string"
    ],
  "id": "string",
  "name": "string",
```

```
  "modifiedObjects": {},
  "forceRefreshDeploymentData": False,
  "type": "deploymentstatus"
  }

  data = json.dumps(payload)

  response = requests.post(deploy, headers=headers, data=data,verify=False)
  if response.status_code == 200:
    print(response.status_code)
    print('Deploying')
  else:
    print(response.status_code)
    pass




#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

def revoke():

  uri = '/api/fdm/v6/fdm/token'
  token_url = protocol+url+uri

  headers = {
      "Content-Type": "application/json",
      "Accept": "application/json",
      "Authorization":"Bearer"
  }
  payload = {
        'grant_type' : 'revoke_token',
        "access_token": token,
        "token_to_revoke": token,
        }

  response = requests.post(token_url, data=payload, verify=False)
  if response.status_code == 200:
    print("Access token revoked")
  else:
    print(response.status_code)
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++




while True:
  x = input("""
Press 1 for deployment status
Press 2 to deploy
Enter Exit to exit
> """).lower()
  if x == "1":
    deploy()
    pprint('Status is: ' + status)
  elif x == "2":
    deploy2()
  elif x == "exit":
    break
```

```
    revoke()
  elif x != "1" or "2" or "exit":
    attempts += 1
    print("The options are 1, 2 or Exit only!")
    if attempts == 3:
      print('Closing the program')
      revoke()
      break
  else:
    raise Exception("Program shutting down")
```

Verify

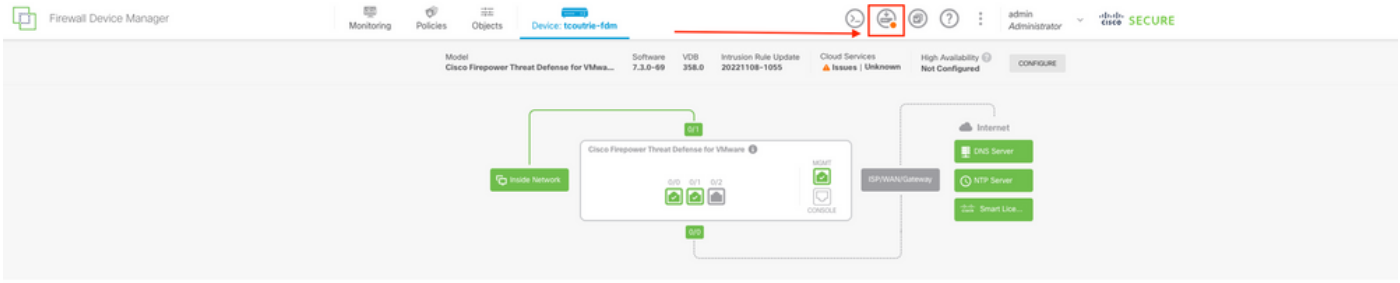
GET

With that script, an HTTP code response "200" is received, and the output is presented, as shown in the image below.

Output from the script

PUT

A new deployment is ready on the FDM GUI with successful attempts, as shown in the image.



Screenshot of FDM admin page

Confirmation can also be confirmed by receiving an HTTP response code 200.

DELETE

A new deployment is ready on the FDM GUI with successful attempts.

Confirmation can also be confirmed by receiving an HTTP response code 204.

Troubleshoot

This section describes how to troubleshoot.

The HTTP response code can inform on the specific issues seen:

Client Errors **4XX**

400- Bad request

401- Unauthorized

403- Forbidden

404- Not Found

408- Request Timeout

415- Unsupported Media Type

Server Errors **5xx**

500- Internal Server Error

501- Not Implemented

502- Bad Gateway

503- Service Unavailable

Related Information

RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1

Introduction to Firepower Threat Defense REST API - FTD-API Reference v6(FTD v7.2) - Document - Cisco Developer