Push Objects in Bulk to FMC using REST-API

Contents

[Introduction](#)
[Prerequisites](#)
[Requirements](#)
[Components Used](#)
[Limitations](#)
[Background Information](#)
[Configure](#)
[Verify](#)
[Troubleshoot](#)

Introduction

This document describes how an Application Programming Interface (API) administrator can push Network, Port, and URL Objects in bulk to Firepower Management Center(FMC).

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Understanding of various REST API calls. (What are REST APIs?)
- Review of the FMC API Quick Start Guide
- Review of FMC Reusable Objects
- Basic knowledge of Python requests library

Components Used

- Firepower Management Center that supports REST APIs (version 6.1 or higher) with REST API enabled
- REST API interactions using Python.

Limitations

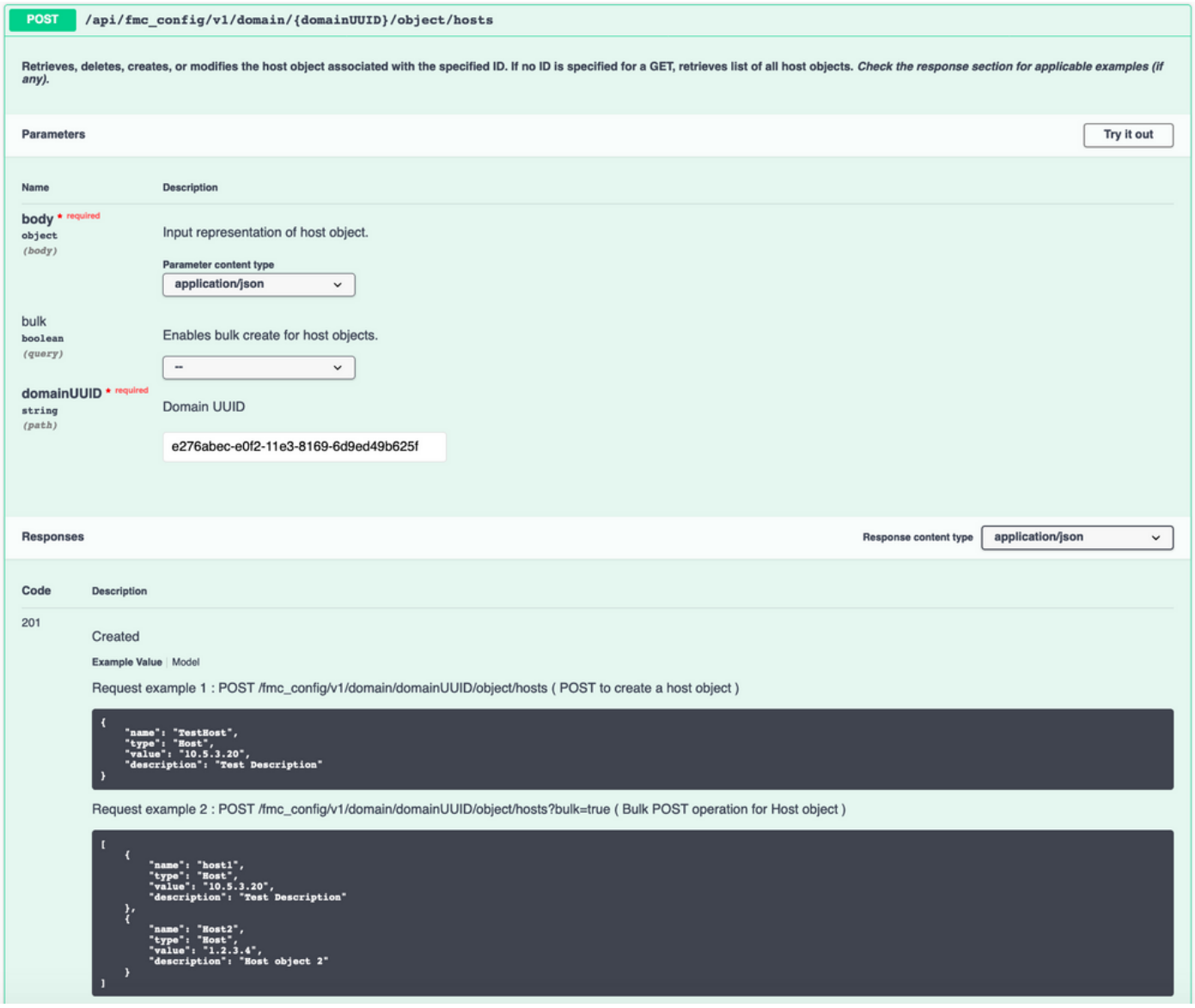
- FMC does not accept the name of the object to be greater than 64 characters.
- Object Name should not have space at the beginning of the object name and semicolon at the end.
- The payload cannot contain more than 1,000 entries in a single Bulk Push.
- Payload size cannot be more than 2MB in a single Bulk Push.

Background Information

REST APIs are increasingly popular due to the lightweight programmable approach that network managers can use to configure and manage their networks. FMC supports configuration and management using any REST Client and also using the in-built API explorer.

The example in this document takes a CSV file as an input and pushes the objects to FMC via the REST API interface. The document covers only the Host Network Bulk push and a similar logic can be extended for the other objects. A sample code is attached to the document for URL and Port objects.

Here is the API reference for the POST on network hosts that are used, as shown in the image:



Configure

Step 1. Enable REST API and generate Authentication Token. For detailed configuration steps and examples Refer [Generate Authentication Token on FMC].

```
import requests import csv import json from requests.auth import HTTPBasicAuth from getpass
```

```
import getpass address = input("Enter IP Address of the FMC: ") username = input ("Enter Username: ") password = getpass("Enter Password: ") api_uri = "/api/fmc_platform/v1/auth/generatetoken" url = "https://" + address + api_uri response = requests.request("POST", url, verify=False, auth=HTTPBasicAuth(username, password)) accesstoken = response.headers["X-auth-access-token"] refreshtoken = response.headers["X-auth-refresh-token"] DOMAIN_UUID = response.headers["DOMAIN_UUID"]
```

Step 2. Convert the CSV File provided to a Dictionary to be used as JSON Payload for the request. Sample CSV File for each Object Type is attached to the document.

	A	B	C	D	
1	name	description	type	value	
2	Host-1	Host-1	Host	10.10.10.10	
3	Host-2	Host-2	Host	10.10.10.1	
4	Network-1	Network-1	Network	10.10.9.0/24	
5	Host-3	Host-3	Host	10.10.10.2	
6	Range-1	Rannge-1	Range	10.20.20.1-10.20.20.20	
7					

```
csvFilePath = input("Please enter the CSV Filepath (For eg. : path/to/file/objects.csv) :") host = [] with open(csvFilePath, encoding='utf-8-sig') as csvf: csvReader = csv.DictReader(csvf) for rows in csvReader: if rows['type'] == "Host": host.append(rows) host_payload = json.dumps(host)
```

The host_payload at this stage looks the same as shown in the image:

```
[{ "name": "Host-1", "description": "Host-1", "type": "Host", "value": "10.10.10.10" }, { "name": "Host-2", "description": "Host-2", "type": "Host", "value": "10.10.10.1" }, { "name": "Host-3", "description": "Host-3", "type": "Host", "value": "10.10.10.2" } ]
```

Step 3. Create the request from the input received from previous steps and send the request if payload is not empty.

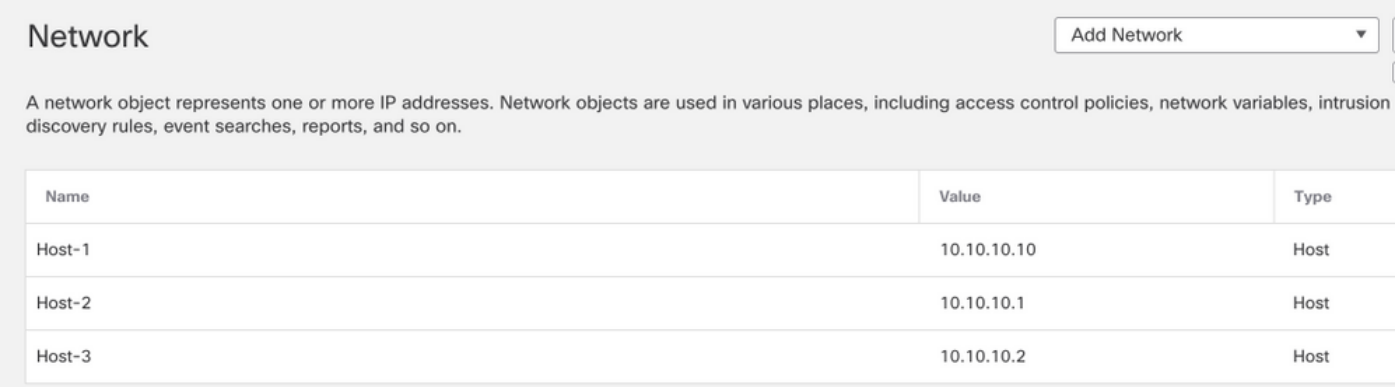
```
host_api_uri = "/api/fmc_config/v1/domain/" + DOMAIN_UUID + "/object/hosts?bulk =true" host_url = "https://" + address + host_api_uri headers = { 'Content-Type': 'application/json', 'x-auth-access-token': accesstoken } if host != []: response = requests.request("POST", host_url, headers=headers, data = host_payload, verify = False) else : print("Please Validate that the CSV file provided is correct or at correct location")
```

Verify

- Print the status code of the response to verify if the request was successful or failed, as shown here.

```
if response.status_code == 201 or response.status_code == 202: print("Host Objects successfully pushed") else: print("Host Object creation failed")
```

- Login to FMC Navigate to **Object > Object Management > Network** and verify the Host Objects, as shown in the image:



Troubleshoot

- When using the REST client, you may see errors related to the SSL certificate problem due to a self-signed certificate. You can turn off this validation depending on the client you are using.
- The FMC REST API authentication tokens are valid for 30 minutes and can be refreshed up to three times.
- The error related to the request can be extracted from the response body. This can be collected as a log file to help with troubleshooting.

```
logfile = "requestlog.txt" log = open(logfile,"w+") log.write(response.text) log.close
```

- All REST requests are logged into these two log files on FMC. Search for your URL (ex. .../object/hosts) with the correct operation(If you are looking for error for GET operation, ensure that your log starts something like GET ...object/hosts)

```
tail -f /var/opt/CSCOpx/MDC/tomcat/logs/stdout.logs tail -f
/var/opt/CSCOpx/MDC/log/operation/usmsharedsvcs.log
```