Empuje objetos en masa a FMC mediante REST-API

Contenido

Introducción

Este documento describe cómo un administrador de la interfaz de programación de aplicaciones (API) puede enviar objetos de red, puerto y URL de forma masiva a Firepower Management Center (FMC).

Prerequisites

Requirements

Cisco recomienda que tenga conocimiento sobre estos temas:

- Comprensión de varias llamadas de API REST. (¿Qué son las API REST?)
- Revisión de la Guía de Inicio Rápido de la API de FMC
- Revisión de objetos reutilizables de FMC
- Conocimiento básico de la biblioteca de solicitudes de Python

Componentes Utilizados

- Firepower Management Center que admite API REST (versión 6.1 o superior) con API REST habilitada
- Interacciones de API REST con Python.

Limitaciones

- FMC no acepta que el nombre del objeto sea superior a 64 caracteres.
- El nombre del objeto no debe tener espacio al principio del nombre del objeto y punto y coma al final.
- La carga útil no puede contener más de 1000 entradas en una única inserción masiva.

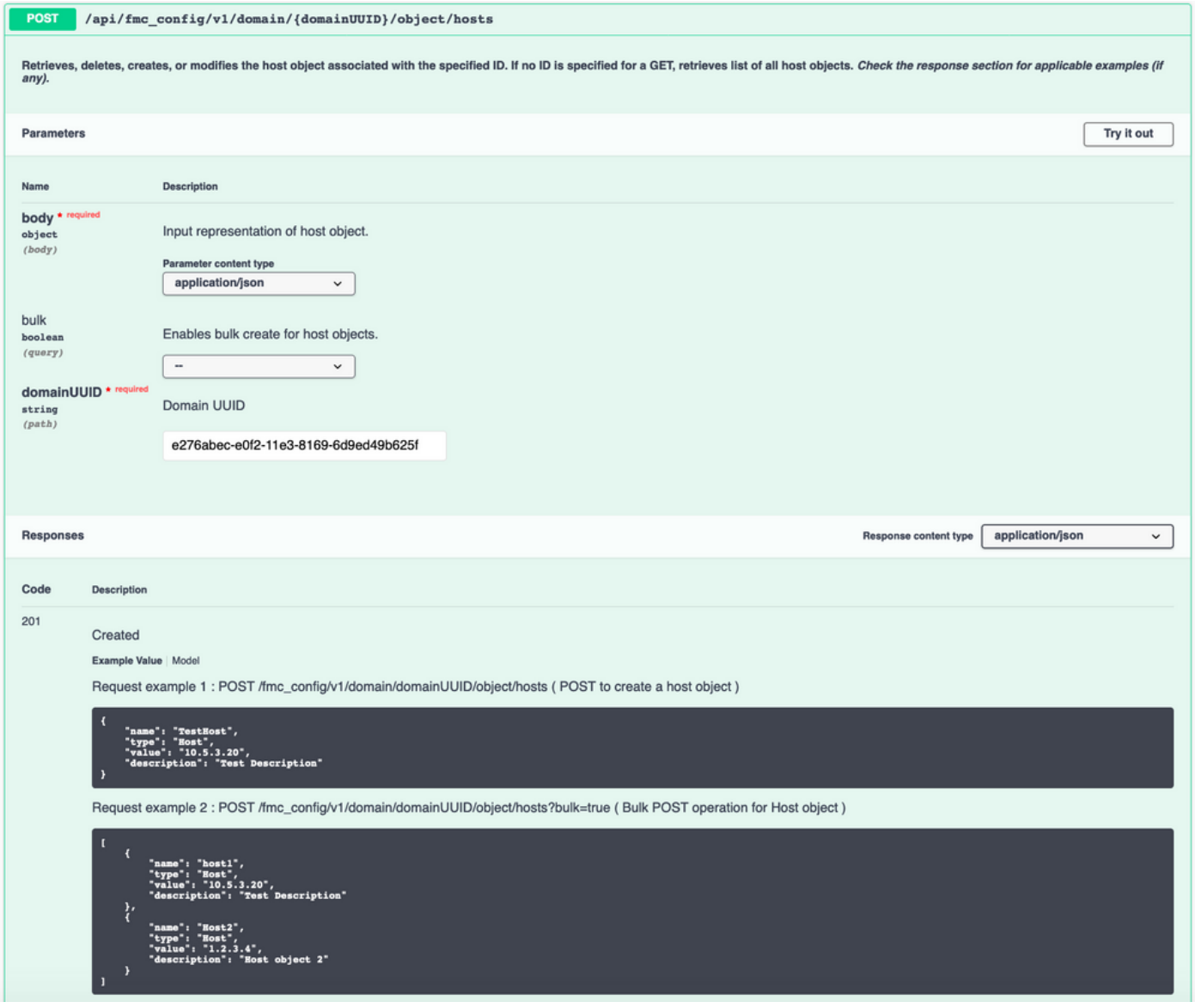
- El tamaño de la carga útil no puede ser superior a 2 MB en una única inserción masiva.

Antecedentes

Las API REST son cada vez más populares debido al enfoque programable ligero que los administradores de red pueden utilizar para configurar y gestionar sus redes. FMC admite la configuración y la gestión mediante cualquier cliente REST y también mediante el explorador API integrado.

El ejemplo de este documento toma un archivo CSV como entrada y envía los objetos a FMC a través de la interfaz API REST. El documento cubre solamente la inserción masiva de red de host y una lógica similar se puede extender para los otros objetos. Se adjunta un código de ejemplo al documento para los objetos URL y Port.

Esta es la referencia de la API para el POST en los hosts de red que se utilizan, como se muestra en la imagen:



Configurar

Paso 1. Habilite la API REST y genere el token de autenticación. Para ver los pasos detallados de

la configuración y ejemplos, consulte Generar token de autenticación en 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"]
```

Paso 2. Convierta el archivo CSV proporcionado a un diccionario para utilizarlo como carga útil de JSON para la solicitud. El archivo CSV de ejemplo para cada tipo de objeto se adjunta al documento.

	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)
```

Host_payload en esta etapa se ve igual que se muestra en la imagen:

```
[{ "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" } ]
```

Paso 3. Cree la solicitud desde la entrada recibida de los pasos anteriores y envíe la solicitud si la carga útil no está vacía.

```
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")
```

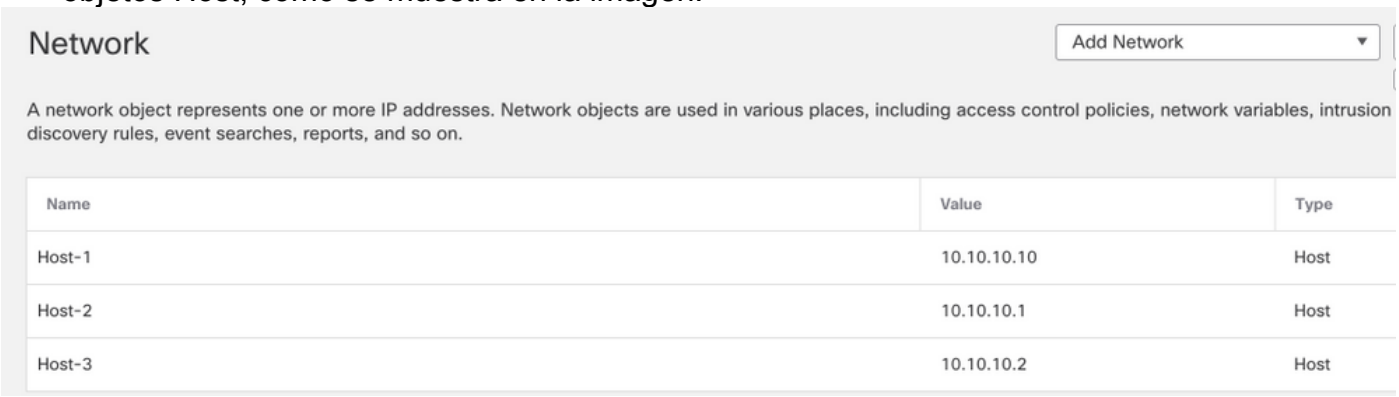
Verificación

- Imprima el código de estado de la respuesta para comprobar si la solicitud se ha realizado correctamente o no, como se muestra aquí.

```
if response.status_code == 201 or response.status_code == 202: print("Host Objects successfully
```

```
pushed") else: print("Host Object creation failed")
```

- Inicie sesión en FMC Navegue hasta **Object > Object Management > Network** y verifique los objetos Host, como se muestra en la imagen:



Troubleshoot

- Al utilizar el cliente REST, puede ver errores relacionados con el problema del certificado SSL debido a un certificado autofirmado. Puede desactivar esta validación en función del cliente que esté utilizando.
- Los tokens de autenticación de la API FMC REST son válidos durante 30 minutos y se pueden actualizar hasta tres veces.
- El error relacionado con la solicitud se puede extraer del cuerpo de respuesta. Esto se puede recopilar como un archivo de registro para ayudar con la resolución de problemas.

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

- Todas las solicitudes REST se registran en estos dos archivos de registro en FMC. Busque su URL (p. ej. .../object/hosts) con la operación correcta(Si busca un error para la operación GET, asegúrese de que el registro inicie algo como GET ...objeto/hosts)

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