Meraki API Demo

Acerca de las APIs

https://documentation.meraki.com/General_Administration/Other_Topics/Cisco_Meraki_Dashboard_API#API_Key_Lifecyc

▼ *Ciclo de vida - API Key*

Mantén tu clave API segura, ya que proporciona autenticación a todas tus organizaciones con acceso a la API habilitado. Por razones de seguridad, el Dashboard no almacena las claves API en texto plano, por lo que esta será la ***única oportunidad para registrarla****. Si pierdes u olvidas tu clave API, tendrás que revocarla y generar una nueva.*

- ***Dos claves API por perfil de usuario****: Después de generar dos claves, el botón "Generate API key" se desactiva. Debes revocar una de las claves existentes antes de poder generar una nueva.*
- ***Las claves API pueden acceder a todas las organizaciones a las que el administrador tiene acceso****: Si un administrador genera una clave API y tiene acceso a múltiples organizaciones, esa clave tendrá acceso a todas las organizaciones a las que pertenezca el administrador.*
- ***Eliminación de administradores y revocación de claves API****: Si se elimina a un administrador del Dashboard de una organización, las claves API generadas por ese administrador dejarán de funcionar para esa organización específica. Si el administrador aún tiene acceso a otras organizaciones, la clave API seguirá funcionando para esas otras organizaciones. Revocar la clave API eliminará completamente el acceso a todas las organizaciones asociadas a ese administrador.*
- ***Las claves API no caducan****: El vencimiento de la contraseña de un administrador no afecta el funcionamiento de su clave API, ya que puede seguir utilizándose independientemente del estado de la contraseña del administrador.*

Meraki Dashboard API Documentation - Ver Base URI

https://developer.cisco.com/meraki/api-v1/introduction/

Guia de Laboratorio 1

https://developer.cisco.com/learning/modules/dne-meraki-dashboard-api/setup-postman-environment/

Guía de Laboratorio 1:

Ejercicio Previo:

- Inicia sesión en el Dashboard y verifica que la organización que usarás en el laboratorio tenga habilitado el acceso por API. Esta configuración se encuentra en:
  - *Organization > API & WebHooks > API keys and access*

- **importar la colección de Postman para Meraki**

https://documenter.getpostman.com/view/897512/SzYXYfmJ

1. Run in Postman → Postman Desktop App to Import
2. En la pestaña *Authorization*, agregar {{apiKey}} a Collection Variables — no olvidar guardar los cambios!

3. **Get Organization ID - List the Organizations - es necesario apiKey como variable.**
4. **Get the Networks in the Organization - es necesario el OrganizationId como variable.**
5. **Get the devices in a Network - es necesario el networkId como variable. {obtener el serial de algún dispositivo como variable}**
6. **Get network Information [platform/configure/getNetwork] - es necesario el networkId como variable**
7. **Get device information [platform/configure/getDevice] - es necesario serialNumber de algún dispositivo**
8. **Get SSID information**
    a. **Listar SSIDs en una red: [products/wireless/configure/ssids/getNetworkWirelessSsids]**

Guía de Laboratorio 2 :

https://developer.cisco.com/meraki/build/automation-with-python-api-lab/

Repositorio con scripts necesarios:

> https://github.com/AngelBautistaC/Meraki-Lab-Python.git

Guía probada en python 3.13.2

Ejercicio A : Instalar python

```
################
Opcional, crear una venv e instalar las librerias requeridas

python3 -m venv labenv
source labenv/bin/activate

Windows

python3 -m venv labenv
labenv/bin/Activate.ps1
################


pip install requests
pip install --upgrade meraki
```

Ejercicio B:

Este es un ejercicio simple en el que ejecutarás un script de Python previamente escrito que recopila el estado del *uplink* de todos los dispositivos en tu organización . El script genera la información en dos archivos CSV: uno para los *appliances*, y otro para todos los demás dispositivos.

1. Abre el archivo **uplink.py** en tu editor de texto o código.
2. Revisa el archivo y observa los comentarios (líneas que empiezan con el símbolo `#`) al inicio de cada sección, para obtener una idea general del flujo de trabajo del script.

3. (**Crear archivo**) Guarda tu clave API y el ID de la organización en un archivo separado llamado **login.py** en la misma carpeta.

Ejemplo de lo que debe contener el archivo **login.py**:

```
api_key = '093b24e85df15a3e66f1fc359f4c48493eaa1b73'
org_id = '537758'
```

1. Ejecuta el script en la terminal o consola de comandos, según tu sistema operativo:

```
python3 uplink.py
```

Ejercicio C:

Este ejercicio tiene como objetivo ayudarte a familiarizarte con el uso del módulo de Python para la API de Meraki Dashboard mediante un script mayormente preescrito, en el cual solo deberás editar algunas secciones específicas y completar ciertos espacios en blanco.

El ejercicio te guiará paso a paso para construir un script que:

1. Cree una red dentro de la organización.
2. Consulte el inventario general de la organización.
3. Agregue un dispositivo del inventario a tu red.
4. Actualice la ubicación de ese dispositivo, y finalmente
5. Modifique la configuración del SSID de la red.

Solo necesitarás editar **las líneas de código** ubicadas entre los bloques de comentarios **##EDITAR##** .

🔧 Para comenzar:

1. Descarga y abre el archivo `Ejercicio_C.py` en un editor de texto.
2. Alternarás entre editar el script `Ejercicio_C.py` , ejecutarlo, consultar la documentación para buscar funciones específicas, y verificar el progreso en la organización del Dashboard.

⚠️ No debes realizar ningún cambio directamente desde la interfaz gráfica del Dashboard. Toda la configuración debe hacerse mediante llamadas a la API.

Parte 1. Crear una red

En esta primera parte, solo necesitas completar la sección de variables con tu nombre, algunas etiquetas (tags) para aplicar a la red y la zona horaria deseada.

Ten en cuenta que esta parte llama dos funciones: `getOrganizationNetworks` (usando el ID de organización) y `createOrganizationNetwork` .

Parte 2. Obtener el inventario

Aquí deberás averiguar cómo llamar a la función `getOrganizationInventoryDevices` .

Consulta la definición de la función en el módulo, o también consultar la documentación.

El código usa una lista por comprensión para asignar a la variable `unused` todos los dispositivos que **no tienen un** `networkId` , es decir, que aún no han sido asignados a ninguna red.

Parte 3. Reclamar un dispositivo a la red

En esta parte del script, se comprueba primero si la red ya tiene un dispositivo asignado. Si **no tiene ninguno**, se recorre la lista de dispositivos no utilizados (`unused`) para identificar algún punto de acceso (AP) cuyo modelo comience con `MR` **o** `CW` .

Luego, deberás **editar manualmente la variable** `my_serial`, ingresando el número de serie (`serial`) de uno de esos APs que se haya mostrado en la consola como disponible.

> ⚠️ A diferencia de otras partes, aquí no se selecciona el dispositivo automáticamente: tú eliges el serial de forma manual, completando la línea indicada.

Después, el script realiza una llamada a la API para **reclamar ese dispositivo (`claim`) en la red** usando `claimNetworkDevices`.

⚠️ Solo reclamarás **un dispositivo** a la red.

Parte 4. Actualizar propiedades del dispositivo

En esta parte, deberás actualizar ciertos atributos del punto de acceso (AP) que fue reclamado en la Parte 3.

Para comenzar, edita la variable `my_address` con una **dirección distinta** a la que viene por defecto. Esta dirección será utilizada para mover el marcador del dispositivo en el mapa del Dashboard de Meraki.

Luego, deberás **editar la línea correspondiente a la llamada de actualización del dispositivo**, utilizando la función `updateDevice(...)`. Esta llamada debe incluir los parámetros adecuados para:

- Identificar el serial(`serial`)
- Activar el movimiento del marcador (`moveMapMarker=True`)
- Actualizar la dirección (`address`)
- Asignar el nombre del dispositivo (`name`)
- Asignar etiquetas (`tags`)

Una vez realizada la llamada, el script obtiene los detalles del dispositivo nuevamente usando su número de serie (`my_serial`) y verifica que:

1. El **nombre** coincida con el definido en `my_name`
2. Las **etiquetas (`tags`)** coincidan con las de `my_tags`
3. La **dirección** coincida con la definida en `my_address`

Parte 5. Actualizar configuración del primer SSID

En esta sección, actualizarás la configuración de un SSID en tu red inalámbrica Meraki. Específicamente, configurarás un SSID con autenticación WPA2-Personal, en la **posición 0**, que representa el primer SSID disponible en la red.

Primero, debes definir el **nombre del SSID** (`my_ssid_name`) y una **clave de seguridad PSK** (`my_ssid_psk`) con al menos 8 caracteres.

A continuación, se realiza una llamada a la función `updateNetworkWirelessSsid`, proporcionando los siguientes parámetros clave:

- `networkId`: ID de la red inalámbrica a configurar
- `number`: número de SSID a actualizar (0 para el primero)
- `name`: nombre del nuevo SSID
- `psk`: clave precompartida (mínimo 8 caracteres)
- `authMode`: tipo de autenticación (`psk`)
- `encryptionMode`: método de cifrado (`wpa`)
- `wpaEncryptionMode`: modalidad WPA2 únicamente
- `enabled`: se activa el SSID

Después de enviar la configuración, el script realiza las siguientes modificaciones:

1. Que el SSID en la posición 0 tenga el **nombre actualizado**
2. Que el nombre definido esté **presente en la lista general** de SSIDs
3. Que el **PSK** coincida con el ingresado

4. Que el **SSID esté habilitado**

> 💡 Esta parte asegura que sabes cómo configurar la seguridad y visibilidad de una red inalámbrica usando exclusivamente la API del Dashboard de Meraki.

🎉***Has completado exitosamente el laboratorio de la API de Meraki en Python. ¡Felicitaciones!***