Dashboard API: From Zero to Hero

A Beginner's Guide to Dashboard API and the Python SDK

John M. Kuchta

Solutions Architect, Cloud Platform & API, Meraki

@TKIPisalegacycipher on GitHub

John M. Kuchta

Solutions Architect, Cloud Platform & API, Meraki

John has been building networks since StarCraft supported IPX/SPX and 802.11b was WiFi. Recently he's worked in the Cisco Gold service provider space and for networking vendors out of Southern California, leading and mentoring field engineers and customer success teams in areas of network orchestration, WiFi design, and stretched datacenter deployments using VXLAN and ACI. He is presently focused on partner enablement and community outreach for the Meraki Dashboard API platform, where he seeks to drive its growth and adoption. He cares deeply about helping network engineers to build skills in Python, Ansible and PowerShell.

DEVNET Create

Problem

Someone made an undocumented change…
Somewhere…

DEVNET Create

Scenario

You are the senior WiFi admin

You work for a company with offices all over the world

Users are complaining about slow WiFi

Scenario Intensifies

Similar issue reports are coming in from around the world

Your company has hundreds of networks, and dozens of different SSIDs

Reviewing the config on each SSID via the GUI could take hours

DEVNET Create

Solution Overview

1 → 2 → 3 → 4

Pick the right API

Review the different APIs that Meraki offers

Build your API dev env

Create a Dashboard account and API key and prepare your dev environment

Pull config info via the Python SDK

Learn how to pull the relevant config information in bulk via Python in seconds

Remediate using the Python SDK

Fix the problem in minutes, rather than hours

DEVNET Create

So, what are the Meraki APIs?

DEVNET Create

Meraki APIs

Captive Portal API

Extends the power of the built-in Meraki splash page functionality

Scanning API

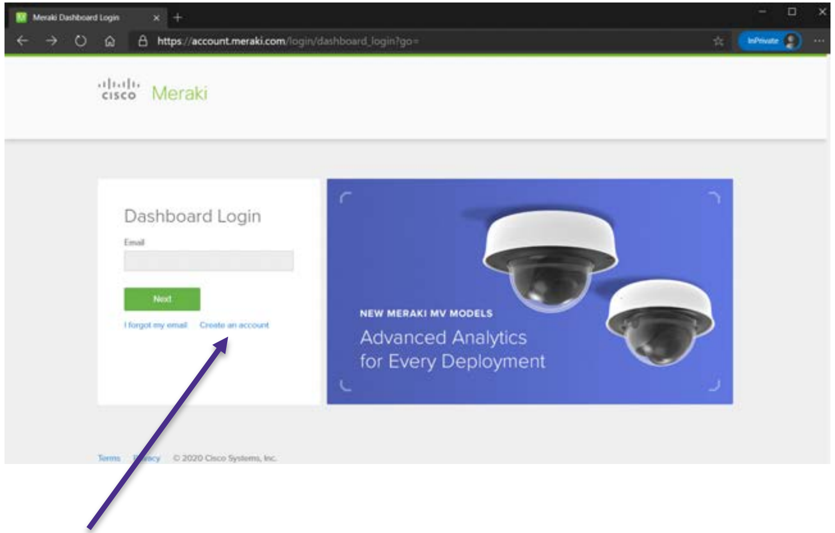
Location analytics to report on foot traffic behavior using WiFi and BLE

Dashboard API

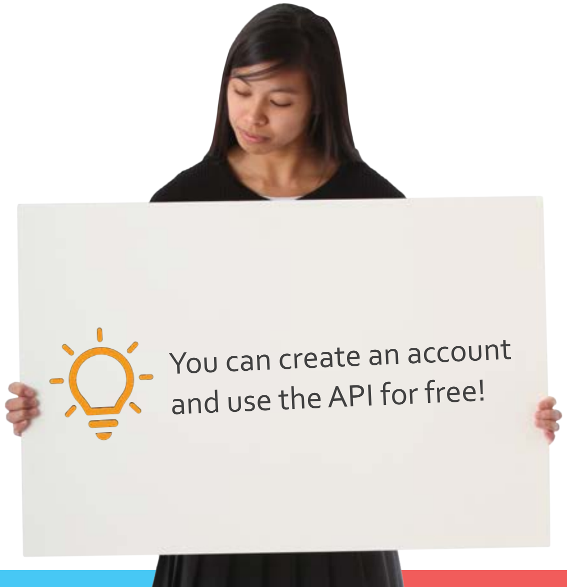
Network programming interface, empowering monitoring and bulk configuration

Build a dev environment for the Meraki Dashboard API

Getting started



Start by logging into your Dashboard account. If you don't have one, create one at dashboard.meraki.com



Enabling API access

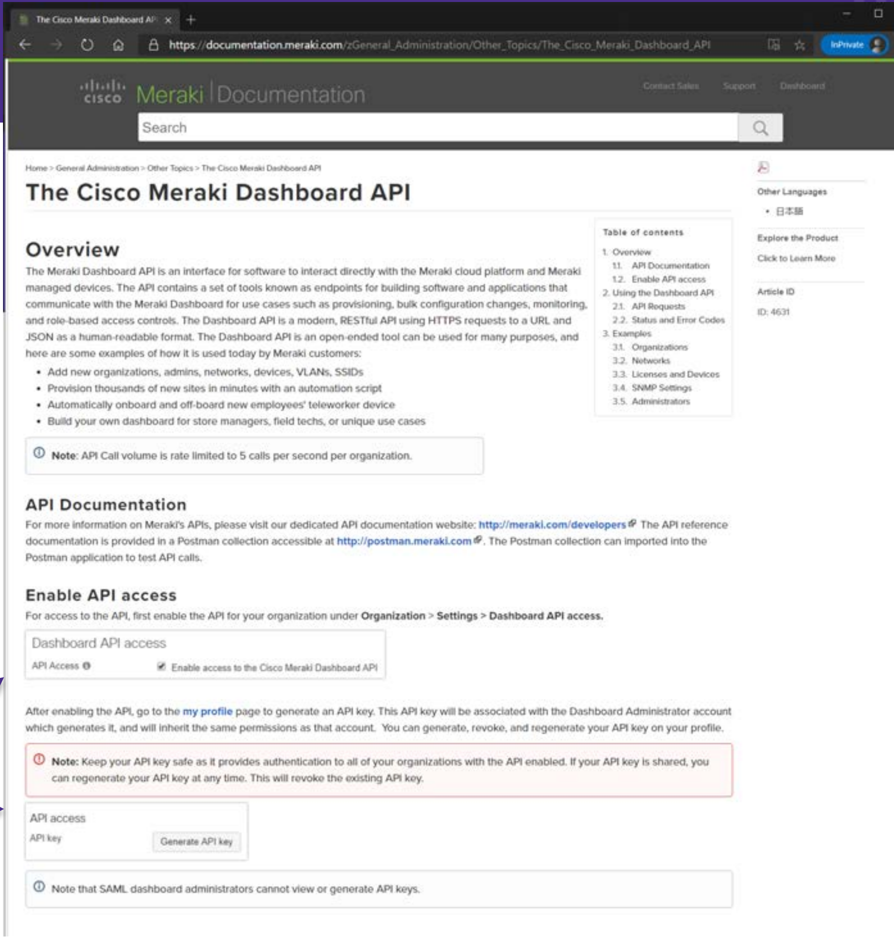
Enable API access and **generate** an API key.

Keep the key in a safe place.

Add it to your dev machine's OS environment variables as MERAKI_DASHBOARD_API_KEY for secure access by your application.

Detailed instructions for different platforms are available here:
https://github.com/meraki/dashboard-api-python/tree/master/notebooks

Treat your API key like a password!


The Cisco Meraki Dashboard AP
https://documentation.meraki.com/zGeneral_Administration/Other_Topics/The_Cisco_Meraki_Dashboard_API
InPrivate
Meraki | Documentation
Contact Sales
Support
Dashboard
Search

Home > General Administration > Other Topics > The Cisco Meraki Dashboard API

The Cisco Meraki Dashboard API

Other Languages
- 日本語

Explore the Product
Click to Learn More

Article ID
ID: 4631

Overview

The Meraki Dashboard API is an interface for software to interact directly with the Meraki cloud platform and Meraki managed devices. The API contains a set of tools known as endpoints for building software and applications that communicate with the Meraki Dashboard for use cases such as provisioning, bulk configuration changes, monitoring, and role-based access controls. The Dashboard API is a modern, RESTful API using HTTPS requests to a URL and JSON as a human-readable format. The Dashboard API is an open-ended tool can be used for many purposes, and here are some examples of how it is used today by Meraki customers:

- Add new organizations, admins, networks, devices, VLANs, SSIDs
- Provision thousands of new sites in minutes with an automation script
- Automatically onboard and off-board new employees' teleworker device
- Build your own dashboard for store managers, field techs, or unique use cases

Note: API Call volume is rate limited to 5 calls per second per organization.

API Documentation

For more information on Meraki's APIs, please visit our dedicated API documentation website: http://meraki.com/developers The API reference documentation is provided in a Postman collection accessible at http://postman.meraki.com. The Postman collection can imported into the Postman application to test API calls.

Enable API access

For access to the API, first enable the API for your organization under **Organization** > **Settings** > **Dashboard API access.**

Dashboard API access
API Access [✓] Enable access to the Cisco Meraki Dashboard API

After enabling the API, go to the my profile page to generate an API key. This API key will be associated with the Dashboard Administrator account which generates it, and will inherit the same permissions as that account. You can generate, revoke, and regenerate your API key on your profile.

Note: Keep your API key safe as it provides authentication to all of your organizations with the API enabled. If your API key is shared, you can regenerate your API key at any time. This will revoke the existing API key.

API access
API key [Generate API key]

Note that SAML dashboard administrators cannot view or generate API keys.

DEVNET Create

Install Python

Install Python on your local dev machine if you haven't already. If you don't have Python, find the downloader for your platform from python.org.

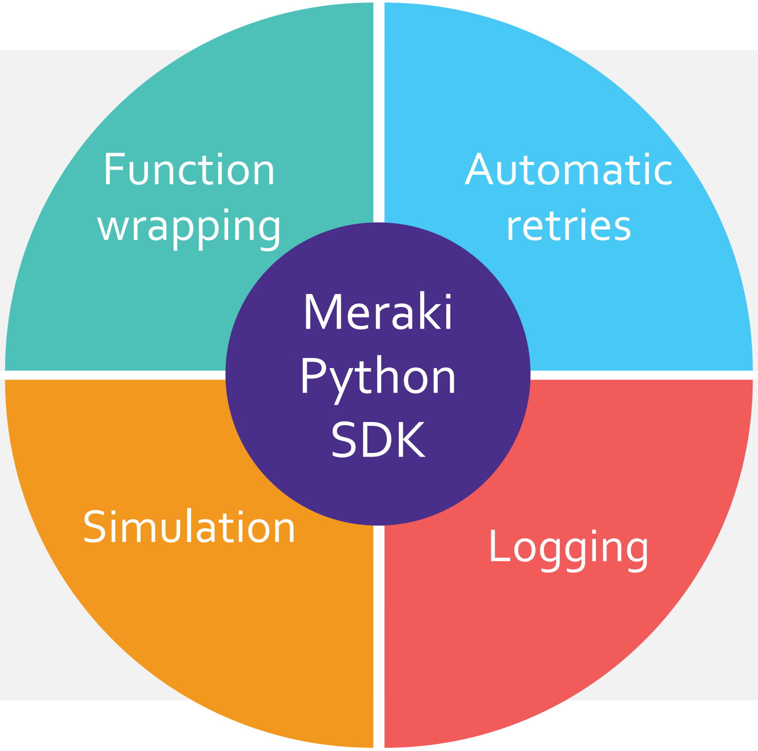
The Meraki Python SDK requires Python 3.6 or later

DEVNET Create

Sidebar: What are the benefits of using Meraki's Python SDK?

Interact with the API like you would a with a normal Python function

If you hit any HTTP 429 rate limit errors, the SDK will automatically retry until successful



Can simulate API calls without changing the prod environment

All API requests are logged a local file as well as the on-screen console

Install the Meraki Python SDK

Now, install the Meraki SDK on the dev machine using pip, the Python package manager.

After installing Python, the following command works from a PowerShell prompt, Command Prompt, Mac Terminal or Linux shell:

```
pip install meraki
```

Once installed, it’s time to start coding.

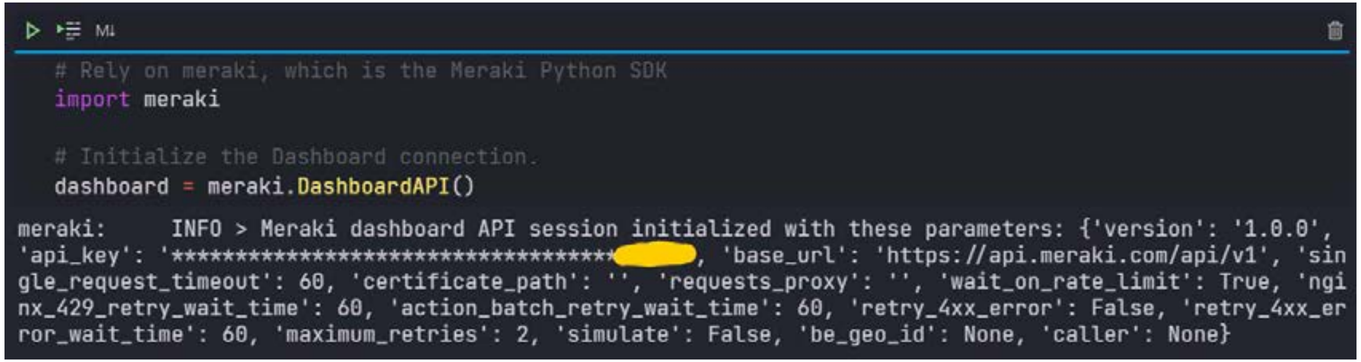
If you have any issues running pip, ensure the python binary has been added to your PATH.

Using the Meraki Python SDK

DEVNET Create

Initialize a Dashboard API session

Using the Python SDK, initializing a Dashboard connection requires only two lines of Python (four, if you count the comments). The console output reflects the successful API session initialization.



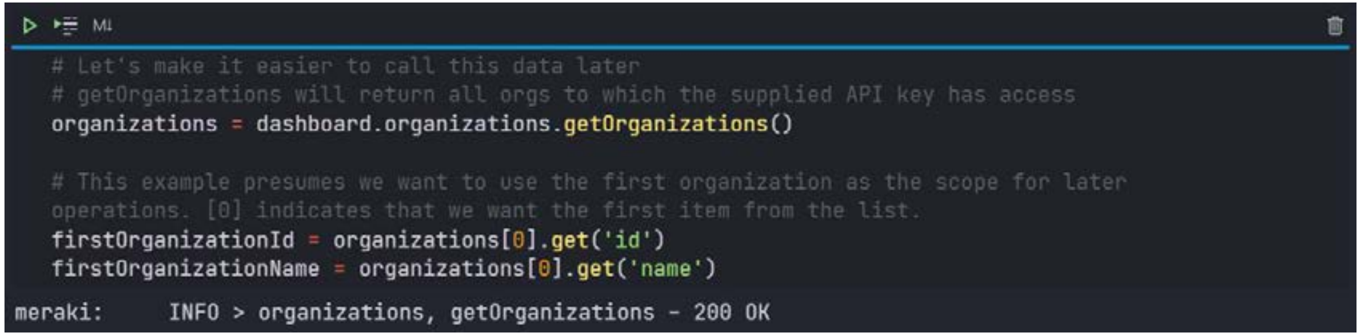
```python
# Rely on meraki, which is the Meraki Python SDK
import meraki

# Initialize the Dashboard connection.
dashboard = meraki.DashboardAPI()
```

```
meraki:     INFO > Meraki dashboard API session initialized with these parameters: {'version': '1.0.0',
'api_key': '************************************[redacted], 'base_url': 'https://api.meraki.com/api/v1', 'sin
gle_request_timeout': 60, 'certificate_path': '', 'requests_proxy': '', 'wait_on_rate_limit': True, 'ngi
nx_429_retry_wait_time': 60, 'action_batch_retry_wait_time': 60, 'retry_4xx_error': False, 'retry_4xx_er
ror_wait_time': 60, 'maximum_retries': 2, 'simulate': False, 'be_geo_id': None, 'caller': None}
```

Gather the required object IDs

Most API calls require passing values for the organization ID and/or the network ID. In this second cell, we fetch a list of the organizations the API key can access. For later operations, we'll use the first organization in the list, but one could instead choose to iterate through every organization, if necessary.

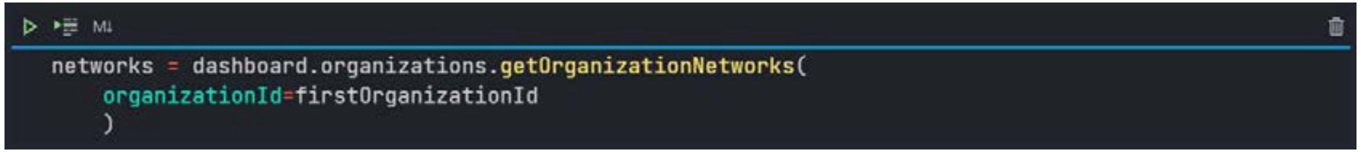


```
# Let's make it easier to call this data later
# getOrganizations will return all orgs to which the supplied API key has access
organizations = dashboard.organizations.getOrganizations()

# This example presumes we want to use the first organization as the scope for later
operations. [0] indicates that we want the first item from the list.
firstOrganizationId = organizations[0].get('id')
firstOrganizationName = organizations[0].get('name')
```

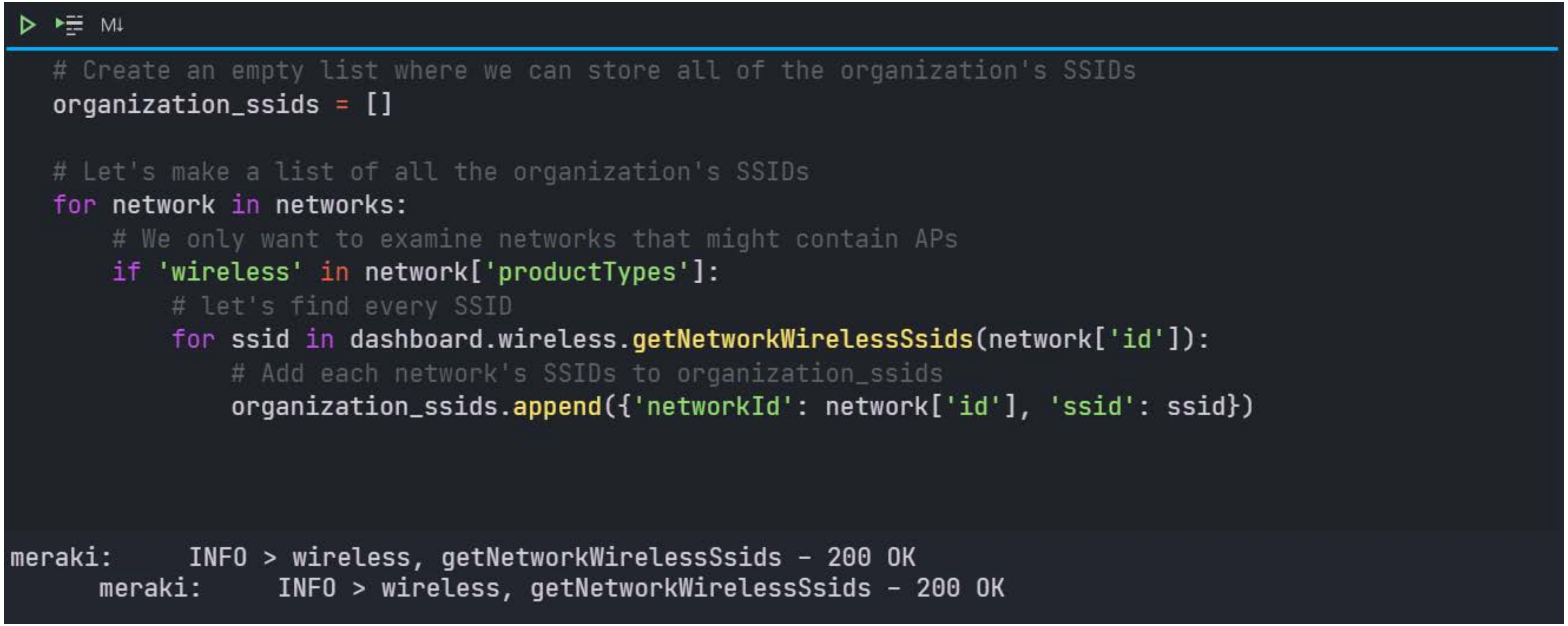
```
meraki:     INFO > organizations, getOrganizations - 200 OK
```

Then the list of networks can be called with a single line of Python, using the getOrganizationNetworks() method:



```
networks = dashboard.organizations.getOrganizationNetworks(
    organizationId=firstOrganizationId
    )
```

Build a list of networks with wireless

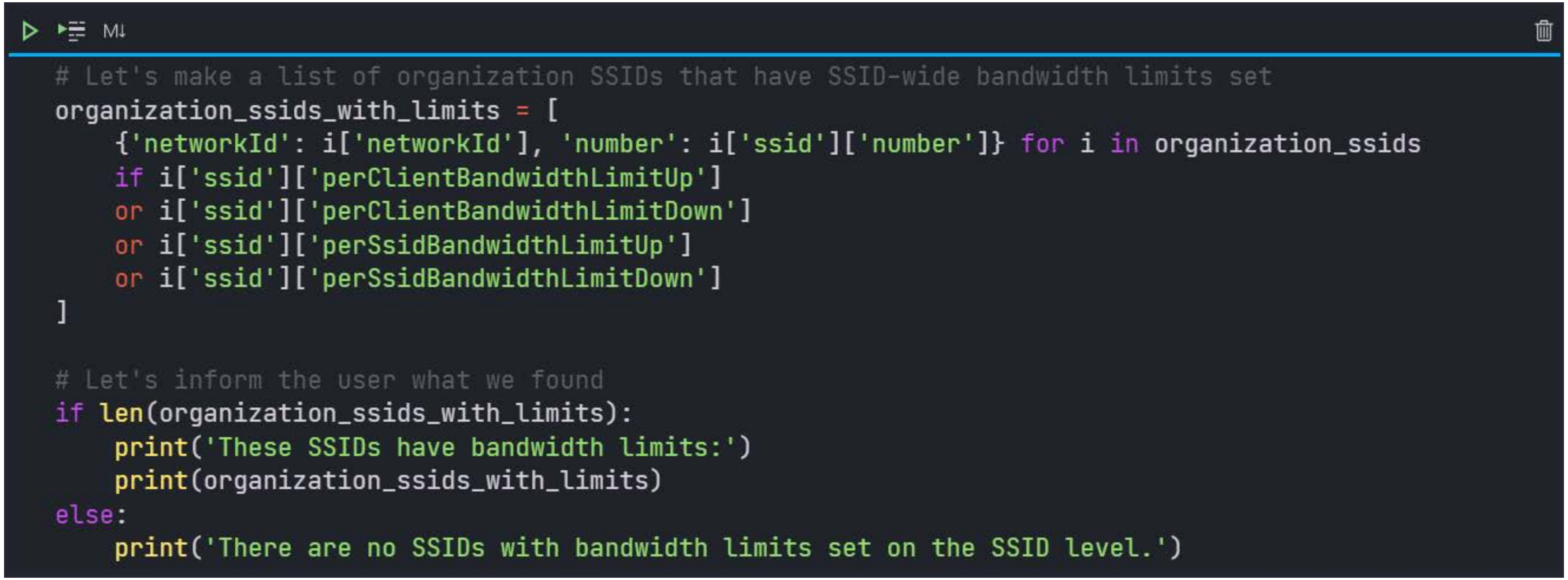


```python
# Create an empty list where we can store all of the organization's SSIDs
organization_ssids = []

# Let's make a list of all the organization's SSIDs
for network in networks:
    # We only want to examine networks that might contain APs
    if 'wireless' in network['productTypes']:
        # let's find every SSID
        for ssid in dashboard.wireless.getNetworkWirelessSsids(network['id']):
            # Add each network's SSIDs to organization_ssids
            organization_ssids.append({'networkId': network['id'], 'ssid': ssid})
```

```
meraki:     INFO > wireless, getNetworkWirelessSsids - 200 OK
      meraki:     INFO > wireless, getNetworkWirelessSsids - 200 OK
```

Build a list of SSIDs that have limits set



M↓

```
# Let's make a list of organization SSIDs that have SSID-wide bandwidth limits set
organization_ssids_with_limits = [
    {'networkId': i['networkId'], 'number': i['ssid']['number']} for i in organization_ssids
    if i['ssid']['perClientBandwidthLimitUp']
    or i['ssid']['perClientBandwidthLimitDown']
    or i['ssid']['perSsidBandwidthLimitUp']
    or i['ssid']['perSsidBandwidthLimitDown']
]

# Let's inform the user what we found
if len(organization_ssids_with_limits):
    print('These SSIDs have bandwidth limits:')
    print(organization_ssids_with_limits)
else:
    print('There are no SSIDs with bandwidth limits set on the SSID level.')
```

Check in!

- So far, we've:
  - Initialized a Dashboard API session
  - Gathered the current-state information for the networks
  - Narrowed that info down to a list of SSIDs that need attention

DEVNET Create

Check in!

- What haven't we done?
  - Manually formatted JSON REST queries!
  - Drilled through dozens of network pages in the UI!

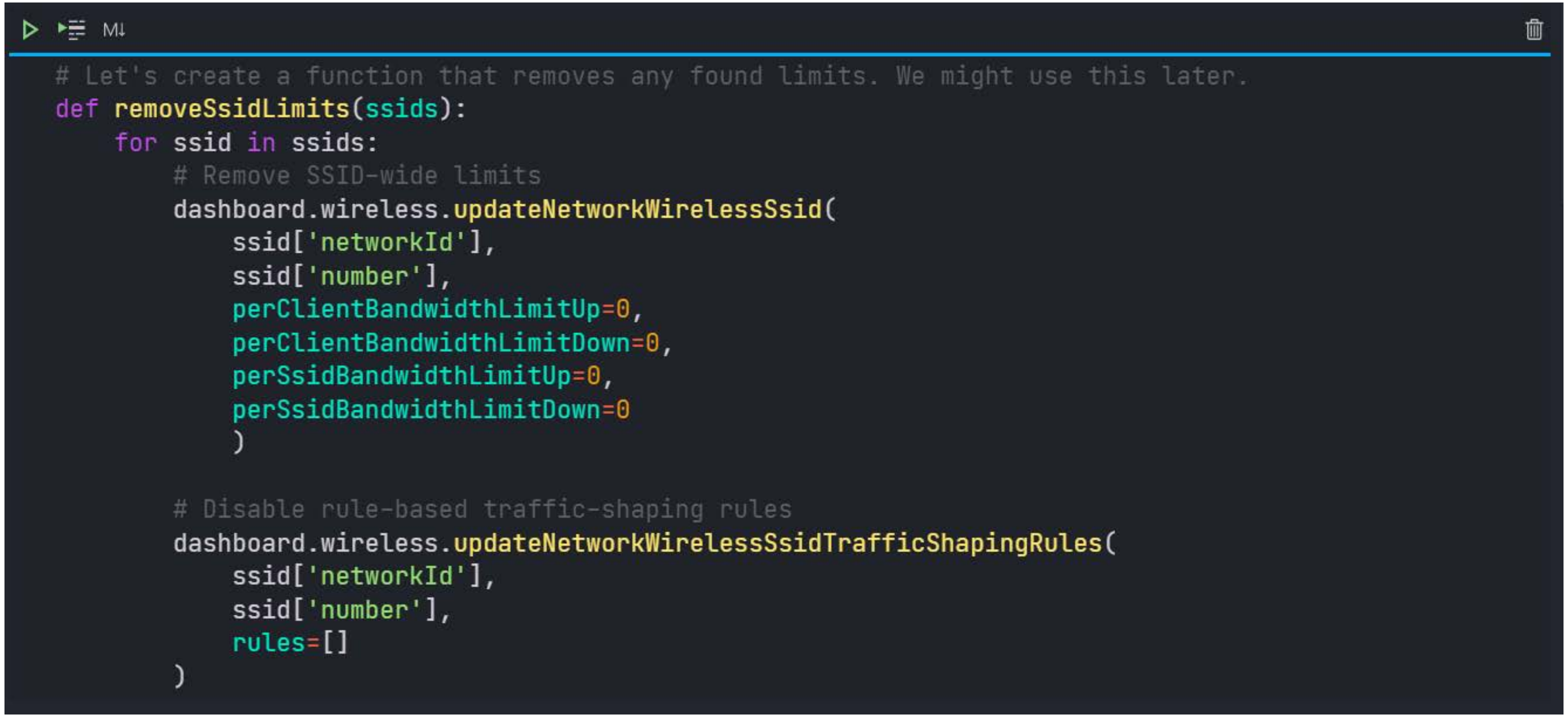
DEVNET Create

Check in!

- What's next?
  - Script the removal of the SSID limits
  - Execute!

DEVNET Create

Build a method that removes SSID limits

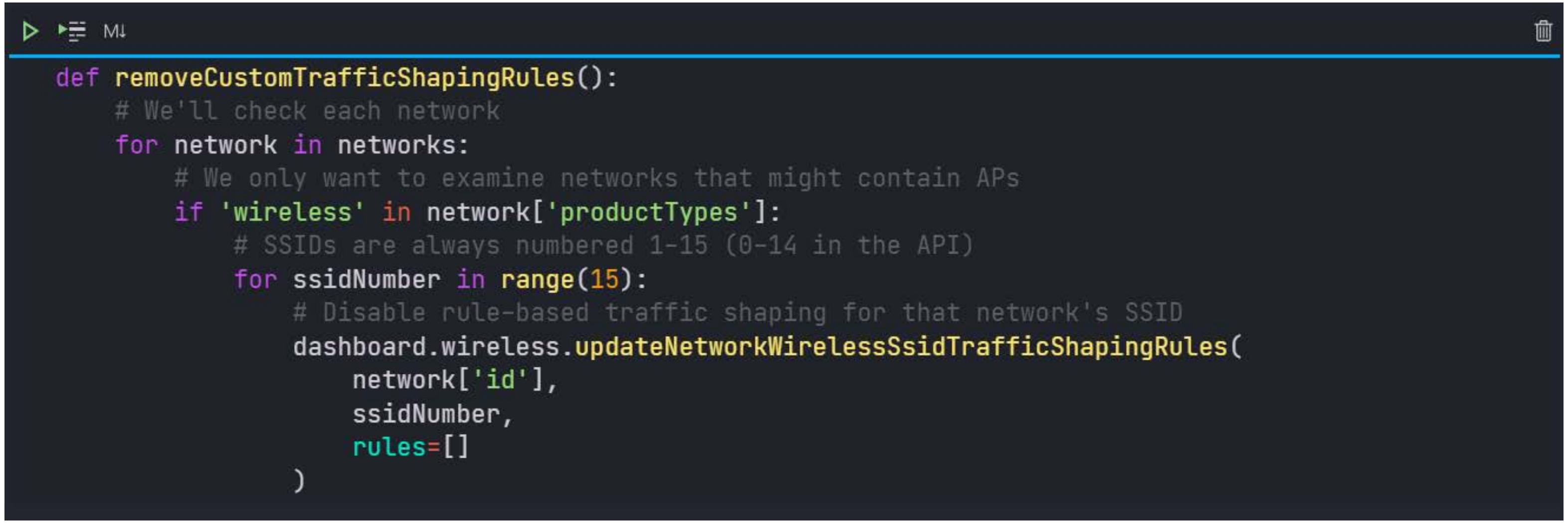


```python
# Let's create a function that removes any found limits. We might use this later.
def removeSsidLimits(ssids):
    for ssid in ssids:
        # Remove SSID-wide limits
        dashboard.wireless.updateNetworkWirelessSsid(
            ssid['networkId'],
            ssid['number'],
            perClientBandwidthLimitUp=0,
            perClientBandwidthLimitDown=0,
            perSsidBandwidthLimitUp=0,
            perSsidBandwidthLimitDown=0
            )

        # Disable rule-based traffic-shaping rules
        dashboard.wireless.updateNetworkWirelessSsidTrafficShapingRules(
            ssid['networkId'],
            ssid['number'],
            rules=[]
        )
```

DEVNET Create

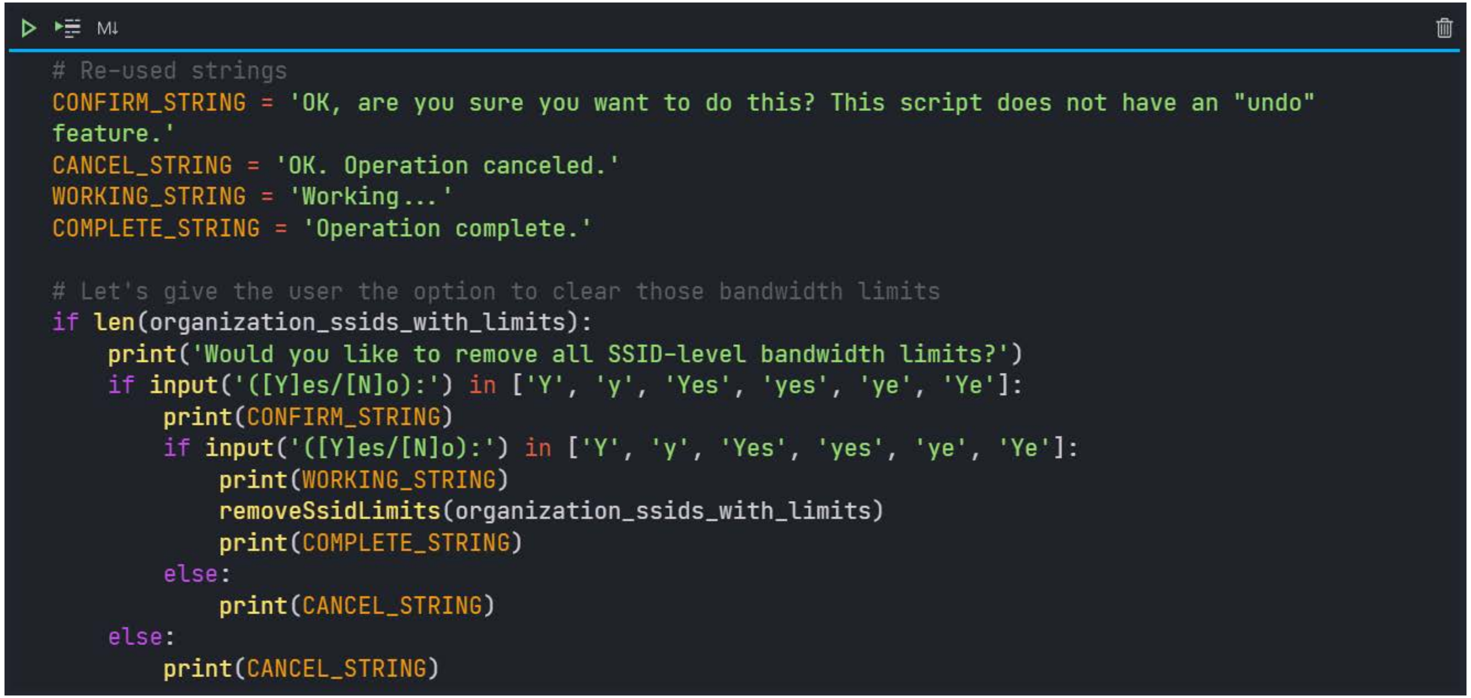
Build a method that removes custom rules



```python
def removeCustomTrafficShapingRules():
    # We'll check each network
    for network in networks:
        # We only want to examine networks that might contain APs
        if 'wireless' in network['productTypes']:
            # SSIDs are always numbered 1-15 (0-14 in the API)
            for ssidNumber in range(15):
                # Disable rule-based traffic shaping for that network's SSID
                dashboard.wireless.updateNetworkWirelessSsidTrafficShapingRules(
                    network['id'],
                    ssidNumber,
                    rules=[]
                )
```

DEVNET Create

Home stretch with some interactivity



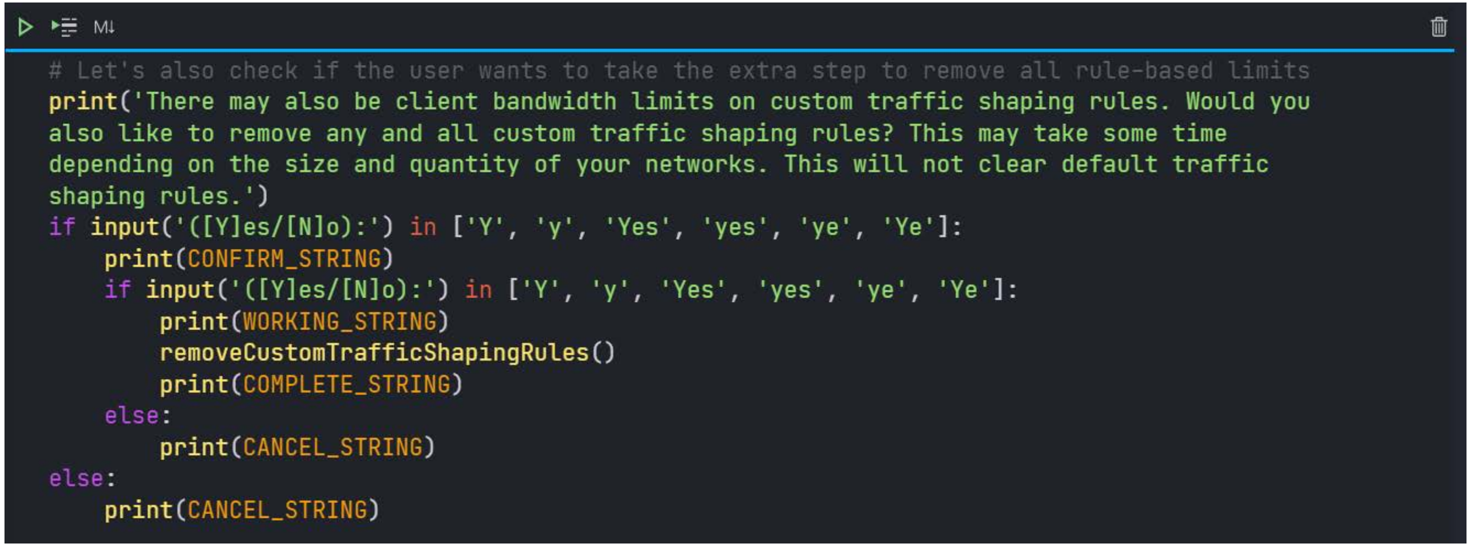
M↓

```
# Re-used strings
CONFIRM_STRING = 'OK, are you sure you want to do this? This script does not have an "undo"
feature.'
CANCEL_STRING = 'OK. Operation canceled.'
WORKING_STRING = 'Working...'
COMPLETE_STRING = 'Operation complete.'

# Let's give the user the option to clear those bandwidth limits
if len(organization_ssids_with_limits):
    print('Would you like to remove all SSID-level bandwidth limits?')
    if input('([Y]es/[N]o):') in ['Y', 'y', 'Yes', 'yes', 'ye', 'Ye']:
        print(CONFIRM_STRING)
        if input('([Y]es/[N]o):') in ['Y', 'y', 'Yes', 'yes', 'ye', 'Ye']:
            print(WORKING_STRING)
            removeSsidLimits(organization_ssids_with_limits)
            print(COMPLETE_STRING)
        else:
            print(CANCEL_STRING)
    else:
        print(CANCEL_STRING)
```

DEVNET Create

Optional extra cleanup



```python
# Let's also check if the user wants to take the extra step to remove all rule-based limits
print('There may also be client bandwidth limits on custom traffic shaping rules. Would you
also like to remove any and all custom traffic shaping rules? This may take some time
depending on the size and quantity of your networks. This will not clear default traffic
shaping rules.')
if input('([Y]es/[N]o):') in ['Y', 'y', 'Yes', 'yes', 'ye', 'Ye']:
    print(CONFIRM_STRING)
    if input('([Y]es/[N]o):') in ['Y', 'y', 'Yes', 'yes', 'ye', 'Ye']:
        print(WORKING_STRING)
        removeCustomTrafficShapingRules()
        print(COMPLETE_STRING)
    else:
        print(CANCEL_STRING)
else:
    print(CANCEL_STRING)
```

DEVNET Create

And that’s it. Well done!

- By writing this application, we were able to quickly find all instances of SSID-level limits that might have been causing a poor user experience.
- If any existed, we gave ourselves the option to remove them with only a few keystrokes.
- If none existed, we were able to confirm this as well, so we could pursue other potential solutions.

DEVNET Create

We did it!

- Together, we built a simple Python application using the Meraki Python SDK, and we never had to worry about formatting JSON REST HTTP requests.
- While simple, this Python application is also modular, so the pieces can be re-used and re-purposed in other Python applications as necessary with minimal effort.

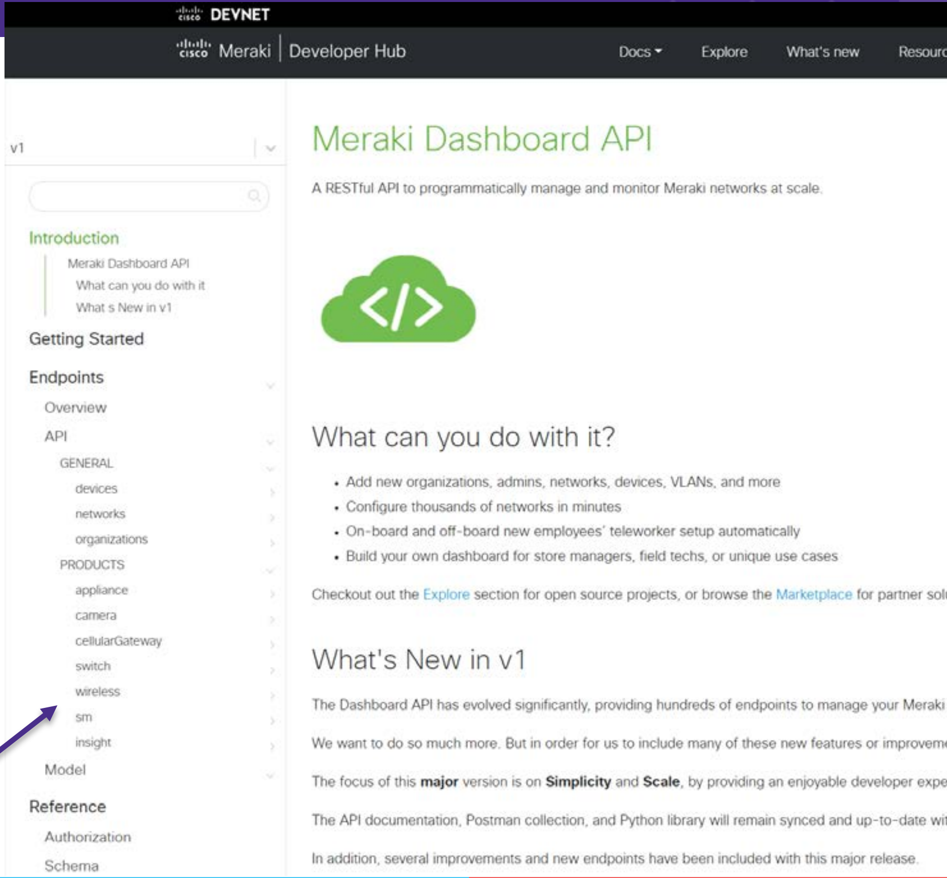
Interactive documentation

The interactive documentation is where you can find the specific API endpoints you'd use to solve any given problem, as well as code examples.

API endpoints are organized by scope (devices, networks, or organizations), or by product, if product-specific.

SSID endpoints are under Products > Wireless.

The interactive API documentation is available here: *https://developer.cisco.com/meraki/api-v1/*



Meraki Dashboard API

A RESTful API to programmatically manage and monitor Meraki networks at scale.

What can you do with it?

- Add new organizations, admins, networks, devices, VLANs, and more
- Configure thousands of networks in minutes
- On-board and off-board new employees' teleworker setup automatically
- Build your own dashboard for store managers, field techs, or unique use cases

Checkout out the Explore section for open source projects, or browse the Marketplace for partner sol

What's New in v1

The Dashboard API has evolved significantly, providing hundreds of endpoints to manage your Meraki

We want to do so much more. But in order for us to include many of these new features or improveme

The focus of this **major** version is on **Simplicity** and **Scale**, by providing an enjoyable developer expe

The API documentation, Postman collection, and Python library will remain synced and up-to-date wi

In addition, several improvements and new endpoints have been included with this major release.

DEVNET Create

DevNet Tools & Resources

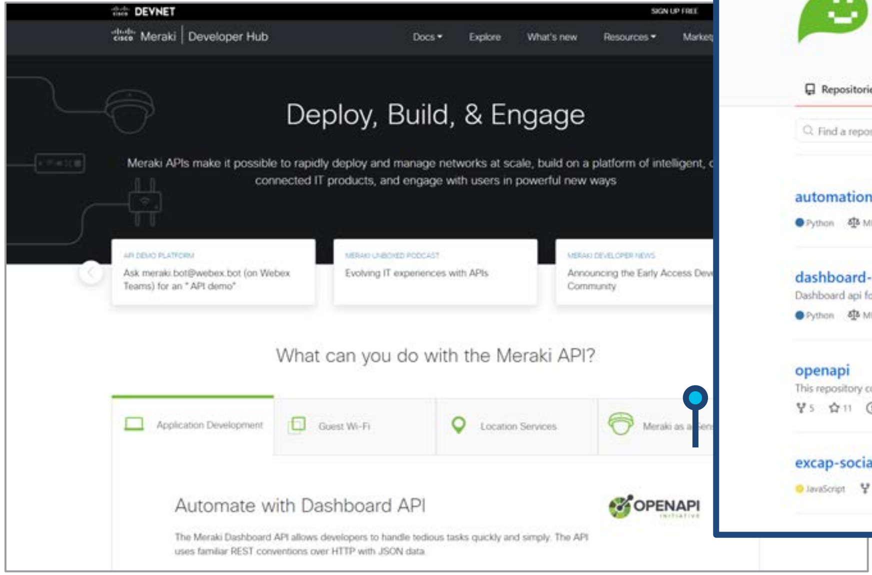
- Meraki Dashboard API Interactive Documentation
  - https://developer.cisco.com/meraki/api-v1/
- Python notebooks for the Meraki Python SDK
  - https://github.com/meraki/dashboard-api-python/tree/master/notebooks
- Meraki Community (Developers & APIs forum)
  - https://community.meraki.com/

Final thoughts

- How might you have done things differently, to meet your specific infrastructure’s technical or procedural needs?
- How might you extend this script to do the following?
  - Make a plain-text, JSON-formatted backup of the original configuration that could be restored using APIs
  - Export a list of affected SSIDs and their networks to a CSV or Excel file for root cause analysis paperwork

DEVNET Create

Code Exchange



Deploy, Build, & Engage

Meraki APIs make it possible to rapidly deploy and manage networks at scale, build on a platform of intelligent, c
connected IT products, and engage with users in powerful new ways

API DEMO PLATFORM
Ask meraki.bot@webex.bot (on Webex Teams) for an " API demo"

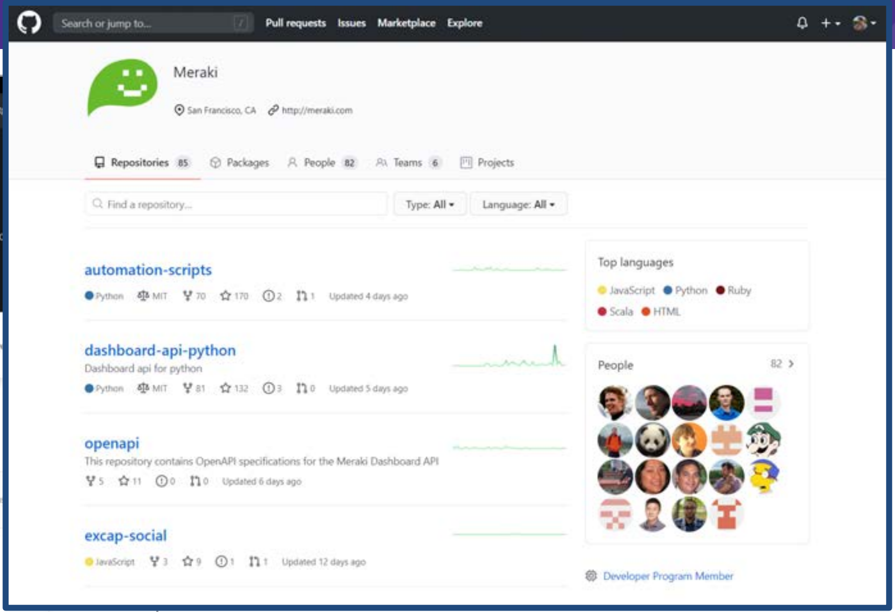
MERAKI UNBOXED PODCAST
Evolving IT experiences with APIs

MERAKI DEVELOPER NEWS
Announcing the Early Access Dev
Community

What can you do with the Meraki API?

Automate with Dashboard API

The Meraki Dashboard API allows developers to handle tedious tasks quickly and simply. The API uses familiar REST conventions over HTTP with JSON data



Meraki

San Francisco, CA http://meraki.com

automation-scripts
Python MIT 70 170 2 1 Updated 4 days ago

dashboard-api-python
Dashboard api for python
Python MIT 81 132 3 0 Updated 5 days ago

openapi
This repository contains OpenAPI specifications for the Meraki Dashboard API
5 11 0 0 Updated 6 days ago

excap-social
JavaScript 3 9 1 1 Updated 12 days ago

Top languages
● JavaScript ● Python ● Ruby
● Scala ● HTML

People 82

Developer Program Member

Find our code at: https://github.com/meraki

DEVNET Create