The bridge to possible

Intro to Meraki Dashboard API Python Library

Adrian ILIESIU, Senior Technical Leader
@aidevnet

Cisco Webex App

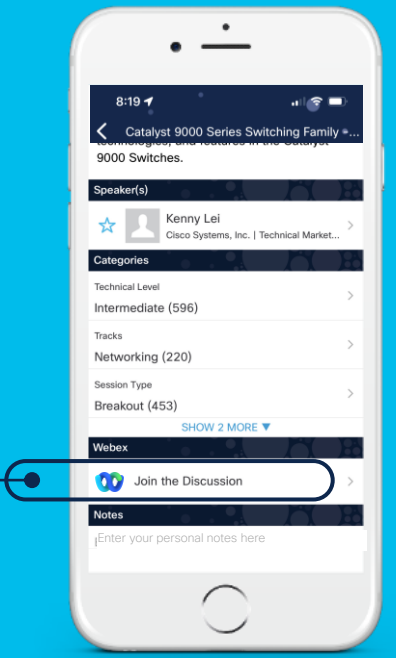
Questions?

Use Cisco Webex App to chat
with the speaker after the session

How

1. Find this session in the Cisco Live Mobile App
2. Click “Join the Discussion”
3. Install the Webex App or go directly to the Webex space
4. Enter messages/questions in the Webex space

Webex spaces will be moderated until February 24, 2023.



Agenda

- Introduction
- Meraki APIs
- Meraki Dashboard API Python library
- AsyncIO
- Conclusion

Introduction

Adrian ILIESIU

Senior Developer Advocate

Enterprise Networking (Cisco DNAC, SD-WAN, Meraki & IOS XE)

- Building networks since 2003
- Network engineer, systems engineer, team leader, QA engineer
- CCNA, CCNP, CCIP, CCIE EI
- Co-author of the DEVASC official certification guide

Meraki APIs

The Value of APIs

Why do customers use APIs?

AUTOMATE

Accelerate everything from minor adjustments to enterprise-sized deployments by **automating configuration changes, provisioning, and data collection.**

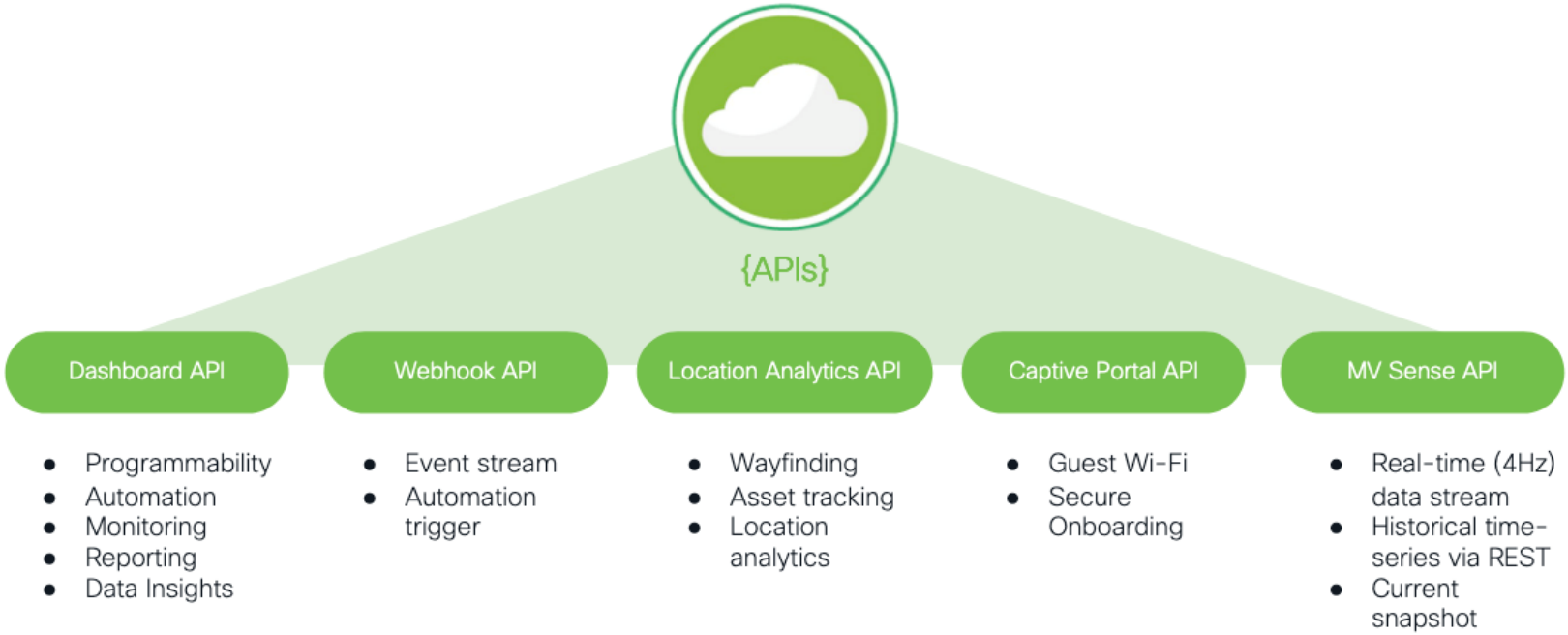
INTEGRATE

Unlock new insights, streamline business workflows and **realize additional value from your existing business systems** by integrating Meraki dashboard with your line-of-business applications.

ANTICIPATE

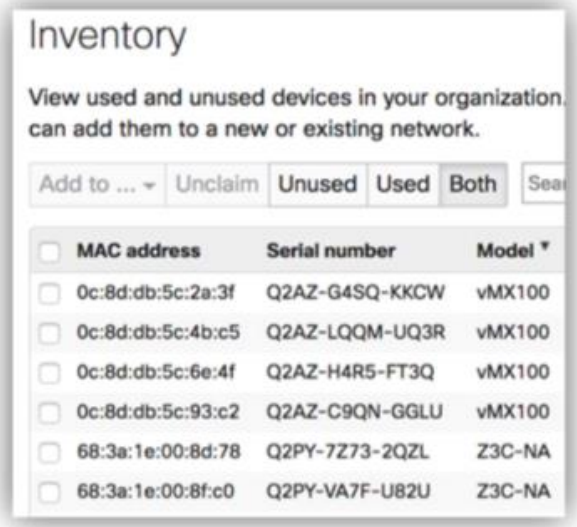
Plan for the future with historical metrics, real-time telemetry and client location analytics to **make data-driven business decisions.**

Meraki API Services, Each with Different Purposes



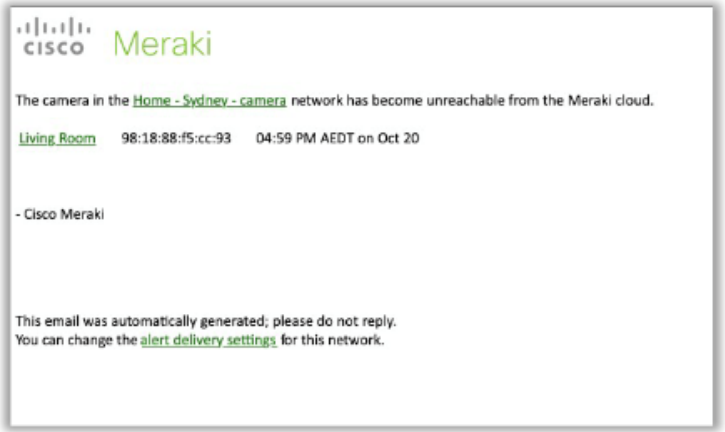
Dashboard API

Dashboard Web UI	APIs



Webhook API

Dashboard Web UI	APIs



cisco Meraki

The camera in the Home - Sydney - camera network has become unreachable from the Meraki cloud.

Living Room 98:18:88:f5:cc:93 04:59 PM AEDT on Oct 20

- Cisco Meraki

This email was automatically generated; please do not reply.
You can change the alert delivery settings for this network.


```
{
    "version": "0.1",
    "sharedSecret": "",
    "sentAt": "2022-10-
20T06:28:24.377119Z",
    "organizationId": "734417",
    "organizationName": "Yuji Terada
Personal",
    "organizationUrl":
"https://n290.meraki.com/o/vSj8Qb/manage
/organization/overview",
    "networkId": "L_726205439913496800",
    "networkName": "Home - Sydney",
    "networkUrl":
"https://n290.meraki.com/Home-Sydney-
came/n/leTphdIe/manage/nodes/list",
    "networkTags": [],
    "deviceMac": "98:18:88:f5:cc:93",
    "deviceName": "Living Room",
```

Location Analytics API

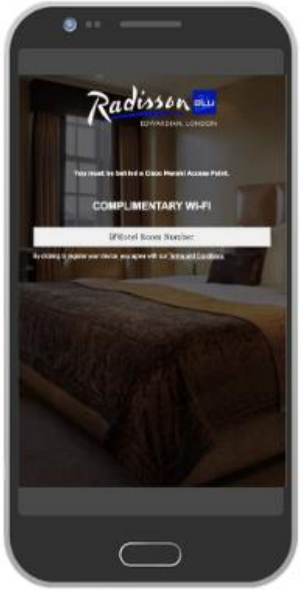
Dashboard Web UI

APIs


```
{
    "ipv4": "/192.168.0.63",
    "location": {
        "lat": 51.5355157,
        "lng": -0.06990350000000944,
        "unc": 0.20755340376944298,
        "x": □,
        "y": □
    },
    "seenTime": "2016-09-24T00:06:27Z",
    "ssid": ".interwebs",
    "os": null,
    "clientMac": "18:fe:34:e1:b4:7a",
    "seenEpoch": 1474675587,
    "rssi": 56,
    "ipv6": null,
    "manufacturer": "Espressif"
```

Captive Portal API

Dashboard Web UI | APIs




```
<html>
<head><title>Internet Access Login</title></head>
<body>
<h2>Internet Access Login</h2>
<form method="POST" action="https://example.meraki.com/splash
/
login/?mauth=ABCDEFG123456">
<input type="hidden" name="success_url" value="http://www.
example.com/success/" />
Username: <input type ="text" name="username" />
Password: <input type ="text" name="password" />
<input type="submit" value="Login" />
</form>
</body>
</html>
```

MV Sense API

Dashboard Web UI

APIs


```
'{"lux": 88.4}'
'{"ts":1666242984676, "counts":{"person":0}}'
'{"lux": 88.6}'
'{"ts":1666242984753, "counts":{"person":0}}'
'{"lux": 88.4}'
'{"ts":1666242984947, "counts":{"person":0}}'
'{"ts":1666242985077, "counts":{"person":0}}'
'{"lux": 88.6}'
'{"ts":1666242985151, "counts":{"person":0}}'
'{"ts":1666242985361, "counts":{"person":0}}'
'{"lux": 88.4}'
'{"ts":1666242985478, "counts":{"person":0}}'
'{"lux": 88.6}'
'{"ts":1666242985553, "counts":{"person":0}}'
'{\n\t"objects" : [],\n\t"ts" : 1666242985753\n}'
'{"ts":1666242985753, "counts":{"person":0}}'
'{"lux": 88.4}'
'{"ts":1666242985876, "counts":{"person":0}}'
'{"lux": 88.4}'
'{"ts":1666242985953, "counts":{"person":0}}'
'{"lux": 88.2}'
'{"ts":1666242986156, "counts":{"person":0}}'
```

Meraki Dashboard API Python library

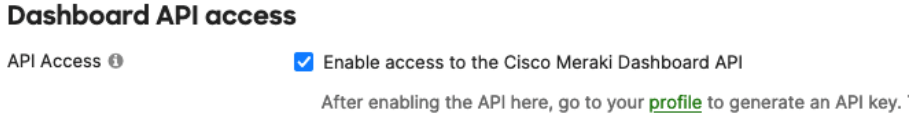
cisco Live!

Introduction

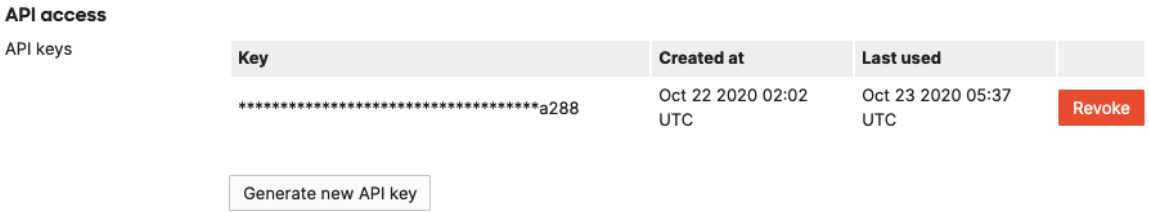
- The Meraki Dashboard API Python library provides all current Meraki Dashboard API calls to interface with the Cisco Meraki cloud-managed platform
- Python >= 3.7
- pip install meraki
- Meraki Dashboard API versions:
  - v0: August-05-2020 deprecation, February-05-2022 sunset, August-05-2022 End of Grace period (no support escalations, operation not guaranteed)
  - v1: Active version (API endpoint to help with migration: GET /organizations/:organizationId/apiRequests?version=0)

Meraki Dashboard API access

- Enable API access: Organization > Settings > Dashboard API access



- Generate API key: My Profile



Features

- **Support for all API endpoints**, as it uses the OpenAPI specification to generate source code
- **Log all API requests** made to a local file as well as on-screen console
- **Automatic retries** upon 429 rate limit errors, using the Retry-After field within response headers
- Get all (or a specified number of) pages of data with **built-in pagination control**
- Tweak settings such as **maximum retries, certificate path, suppress logging**, and other options
- **Simulate POST/PUT/DELETE calls to preview first**, so that network configuration does not get changed
- Includes the legacy module's (version 0.34 and prior) functions for **backward compatibility**

DashboardAPI Class Parameters

```
class DashboardAPI(builtins.object)
 |  DashboardAPI(api_key=None, base_url='https://api.meraki.com/api/v1', single_request_timeout=60, certificate_path='', requests_proxy='', wait_on_rate_limit
=True, nginx_429_retry_wait_time=60, action_batch_retry_wait_time=60, retry_4xx_error=False, retry_4xx_error_wait_time=60, maximum_retries=2, output_log=True,
 log_path='', log_file_prefix='meraki_api_', print_console=True, suppress_logging=False, simulate=False, be_geo_id='', caller='', use_iterator_for_get_pages=F
alse, inherit_logging_config=False)
 |
 |  **Creates a persistent Meraki dashboard API session**
 |
 |  - api_key (string): API key generated in dashboard; can also be set as an environment variable MERAKI_DASHBOARD_API_KEY
 |  - base_url (string): preceding all endpoint resources
 |  - single_request_timeout (integer): maximum number of seconds for each API call
 |  - certificate_path (string): path for TLS/SSL certificate verification if behind local proxy
 |  - requests_proxy (string): proxy server and port, if needed, for HTTPS
 |  - wait_on_rate_limit (boolean): retry if 429 rate limit error encountered?
 |  - nginx_429_retry_wait_time (integer): Nginx 429 retry wait time
 |  - action_batch_retry_wait_time (integer): action batch concurrency error retry wait time
 |  - retry_4xx_error (boolean): retry if encountering other 4XX error (besides 429)?
 |  - retry_4xx_error_wait_time (integer): other 4XX error retry wait time
 |  - maximum_retries (integer): retry up to this many times when encountering 429s or other server-side errors
 |  - output_log (boolean): create an output log file?
 |  - log_path (string): path to output log; by default, working directory of script if not specified
 |  - log_file_prefix (string): log file name appended with date and timestamp
 |  - print_console (boolean): print logging output to console?
 |  - suppress_logging (boolean): disable all logging? you're on your own then!
 |  - inherit_logging_config (boolean): Inherits your own logger instance
 |  - simulate (boolean): simulate POST/PUT/DELETE calls to prevent changes?
 |  - be_geo_id (string): optional partner identifier for API usage tracking; can also be set as an environment variable BE_GEO_ID
 |  - caller (string): optional identifier for API usage tracking; can also be set as an environment variable MERAKI_PYTHON_SDK_CALLER
 |  - use_iterator_for_get_pages (boolean): list* methods will return an iterator with each object instead of a complete list with all items
```

Usage

- Export the API key as an environment variable 👍:

```
export MERAKI_DASHBOARD_API_KEY=093b24e85df15a3e66f1fc359f4c48493eaa1b73
```

- Alternatively, define the API key as a variable in the source code 👎:

```
API_KEY = 'fd6dd87d96915f21bc0e0b3d96a866ff0e53e381'
```

Usage (cont.)

```
import os
import meraki

dashboard = meraki.DashboardAPI(
        api_key=os.getenv('MERAKI_DASHBOARD_API_KEY'),
        base_url='https://api.meraki.com/api/v1/',
        output_log=True,
        log_file_prefix=os.path.basename(__file__)[:-3],
        log_path='',
        print_console=False
)
```

Usage – Get networks in organization

```
# Get list of organizations to which API key has access
organizations = dashboard.organizations.getOrganizations()

# Iterate through list of orgs
for org in organizations:
        print(f'\nAnalyzing organization {org["name"]}:')
        org_id = org['id']

        # Get list of networks in organization
        try:
                networks = dashboard.organizations.getOrganizationNetworks(org_id)
        except meraki.APIError as e:
                print(f'Meraki API error: {e}')
                print(f'status code = {e.status}')
                print(f'reason = {e.reason}')
                print(f'error = {e.message}')
                continue
        except Exception as e:
                print(f'some other error: {e}')
                continue
```

Usage – Get clients in networks

```
# Iterate through networks
total = len(networks)
counter = 1
print(f' - iterating through {total} networks in organization {org_id}')
for net in networks:
        print(f'Finding clients in network {net["name"]} ({counter} of {total})')
        try:
        # Get list of clients on network, filtering on timespan of last 14 days
        clients = dashboard.networks.getNetworkClients(net['id'],
                        timespan=60*60*24*14, perPage=1000, total_pages='all')
        except meraki.APIError as e:
                print(f'Meraki API error: {e}')
                print(f'status code = {e.status}')
                print(f'reason = {e.reason}')
                print(f'error = {e.message}')
        except Exception as e:
                print(f'some other error: {e}')
        counter += 1
```

Usage – Create new network and SSID

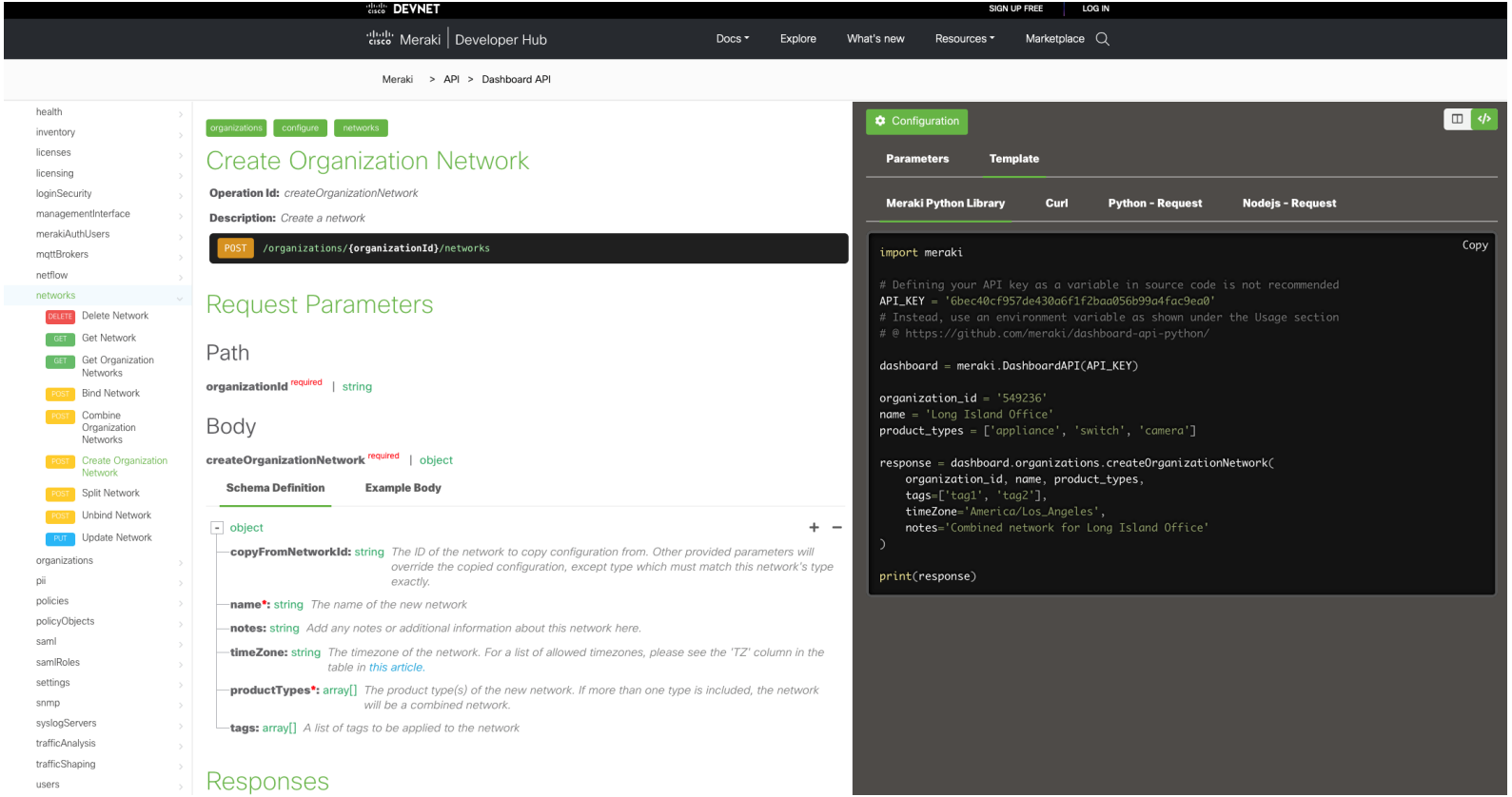
```
# Create a new organization network
organization_id = '549236'
name = 'Long Island Office'
product_types = ['appliance', 'switch', 'camera']

new_network = dashboard.organizations.createOrganizationNetwork(
        organization_id, name, product_types,
        tags=['tag1', 'tag2'],
        timeZone='America/Los_Angeles',
        notes='Combined network for Long Island Office'
)

# Create a new SSID
network_id = 'L_646829496481105433'
number = '0'

new_ssid = dashboard.wireless.updateNetworkWirelessSsid(
        network_id, number,
        name='My SSID',
        enabled=True
)
```

Meraki Python Library Documentation



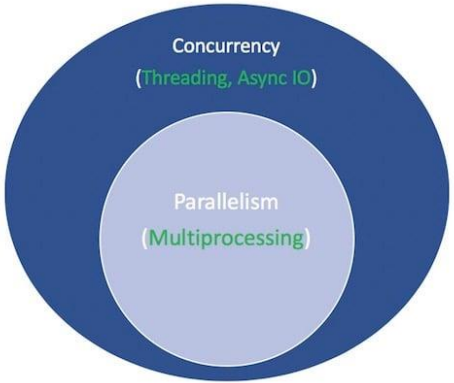
Usage

- Make dashboard API calls using the format ***client.scope.operation***
  - ***client*** is the name of the DashboardAPI hook (*dashboard* in our example)
  - ***scope*** is the corresponding scope that represents the *first tag from the OpenAPI spec*
    - *organizations*
    - *networks*
    - *switch*
    - *wireless*
  - ***operation*** is the operation of the API endpoint
    - *createOrganization*
    - *updateNetworkSwitchRoutingOspf*
    - *deleteNetworkMqttBroker*
    - *getNetworkSwitchAccessPolicies*

AsyncIO

Introduction

- asyncio is a library to write concurrent code using the async/await syntax
- asyncio is not threading or multiprocessing
- asyncio takes long waiting periods in which functions would otherwise be blocking a [illegible] tions to run during that downtime



Source: https://realpython.com/async-io-python/

Usage

```
import asyncio
import os
import meraki.aio

async def main():
        # Instantiate a Meraki dashboard API session
        async with meraki.aio.AsyncDashboardAPI(
                        api_key=os.getenv("MERAKI_DASHBOARD_API_KEY"),
                        base_url="https://api.meraki.com/api/v1",
                        log_file_prefix=__file__[:-3],
                        print_console=False,
        ) as aiomeraki:
                # Get list of organizations to which API key has access
                organizations = await aiomeraki.organizations.getOrganizations()

                # Create a list of all organizations so we can call them all concurrently
                organizationTasks = [listOrganization(aiomeraki, org) for org in organizations]
                for task in asyncio.as_completed(organizationTasks):
                # as_completed returns an iterator, so we just have to await the iterator
        organizationName = await task
                        print(f"finished organization: {organizationName}")
```

Usage – Get networks in organizations

```
async def listOrganization(aiomeraki: meraki.aio.AsyncDashboardAPI, org):
        print(f'Analyzing organization {org["name"]}:’)
        org_id = org["id"]

        # Get list of networks in organization
        try:
                networks = await aiomeraki.organizations.getOrganizationNetworks(org_id)
        except meraki.AsyncAPIError as e:
                print(f"Meraki API error: {e}")
                return org["name"]
        except Exception as e:
                print(f"some other error: {e}")
                return org["name"]
```

Usage – Get clients in networks

```
async def listNetworkClients(aiomeraki: meraki.aio.AsyncDashboardAPI, network):
        print(f'Finding clients in network {network["name"]}')
        try:
                # Get list of clients on network, filtering on timespan of last 14 days
                clients = await aiomeraki.networks.getNetworkClients(
                        network["id"],
                        timespan=60 * 60 * 24 * 14,
                        perPage=1000,
                        total_pages="all",
                )
        except meraki.AsyncAPIError as e:
                print(f"Meraki API error: {e}")
        except Exception as e:
                print(f"some other error: {e}")
```

Conclusion

- Meraki provides several APIs: Dashboard API, Webhook API, Location Analytics API, Captive Portal API, MV Sense API
- Meraki Dashboard API Python library provides all current Meraki Dashboard API calls
- Extra functionality compared to manually building API calls (simulate calls, tweak settings, automatic retries, logging, etc.)
- AsyncIO option also available

Resources

- Meraki Dashboard API Python Library
  - https://github.com/meraki/dashboard-api-python
- Meraki API Documentation
  - https://developer.cisco.com/meraki/api-v1/
- Meraki OpenAPI spec
  - https://api.meraki.com/api/v1/openapiSpec
- Meraki Dashboard Automation using Python - LABMER-2405
  - https://www.ciscolive.com/emea/learn/sessions/session-catalog.html?search=LABMER-2405

Complete your Session Survey

- Please complete your session survey after each session. Your feedback is important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (open from Thursday) to receive your Cisco Live t-shirt.

- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Session Catalog and clicking the " Attendee Dashboard" at https://www.ciscolive.com/emea/learn/sessions/session-catalog.html

Continue Your Education

Visit the Cisco Showcase for related demos.

Book your one-on-one Meet the Engineer meeting.

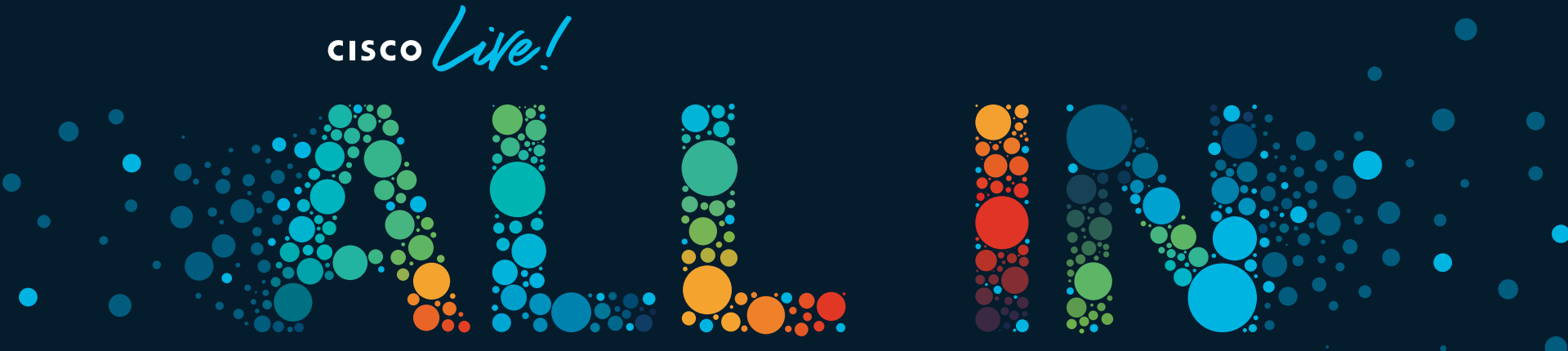
Attend any of the related sessions at the DevNet, Capture the Flag, and Walk-in Labs zones.

Visit the On-Demand Library for more sessions at ciscolive.com/on-demand.

The bridge to possible

Thank you

cisco Live!


CISCO Live!
ALL IN