

<https://account.meraki.com/login/dashboard_login?go=%2F>

* **Managing User Accounts using Meraki Cloud Authentication**

[**https://documentation.meraki.com/General\_Administration/Cross-Platform\_Content/Managing\_User\_Accounts\_using\_Meraki\_Cloud\_Authentication**](https://documentation.meraki.com/General_Administration/Cross-Platform_Content/Managing_User_Accounts_using_Meraki_Cloud_Authentication)

**Creating Meraki Cloud Authentication Guest Users**

* **Client VPN OS Configuration**

<https://documentation.meraki.com/MX/Client_VPN/Client_VPN_OS_Configuration>

**Client VPN Overview**

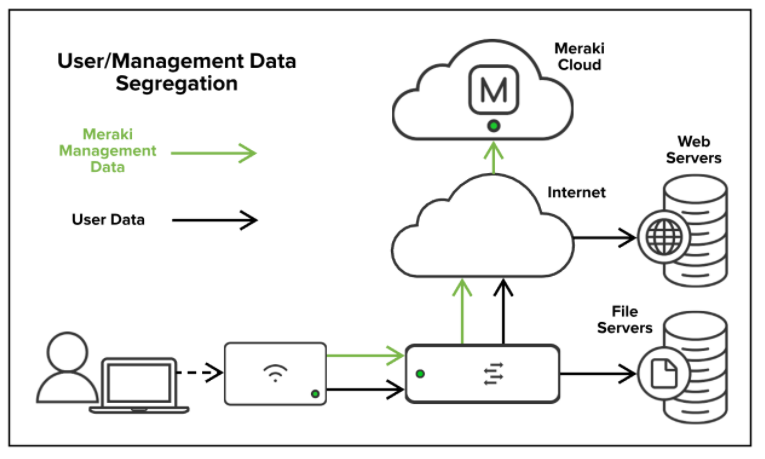
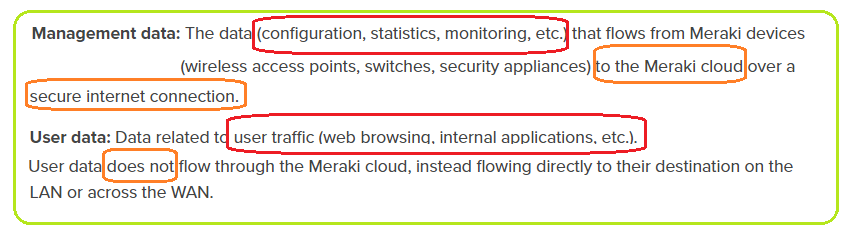
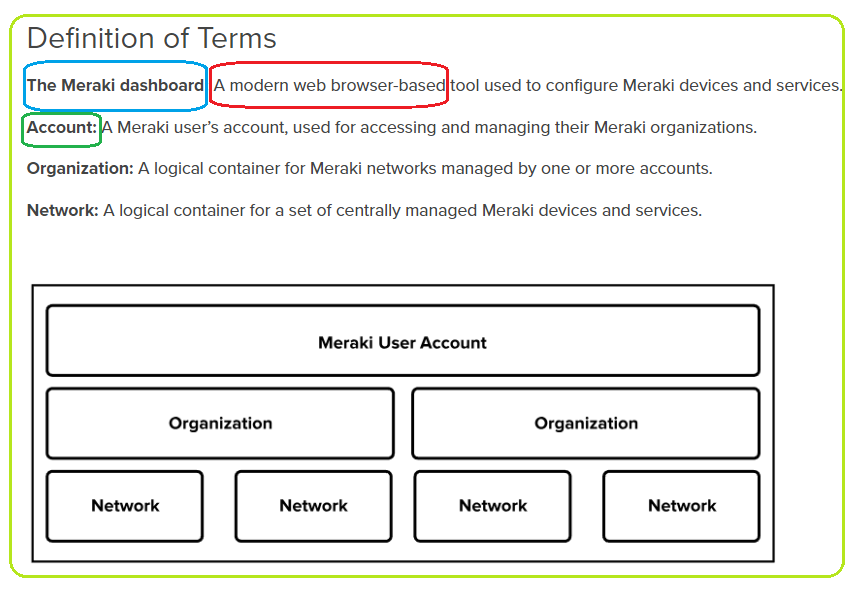
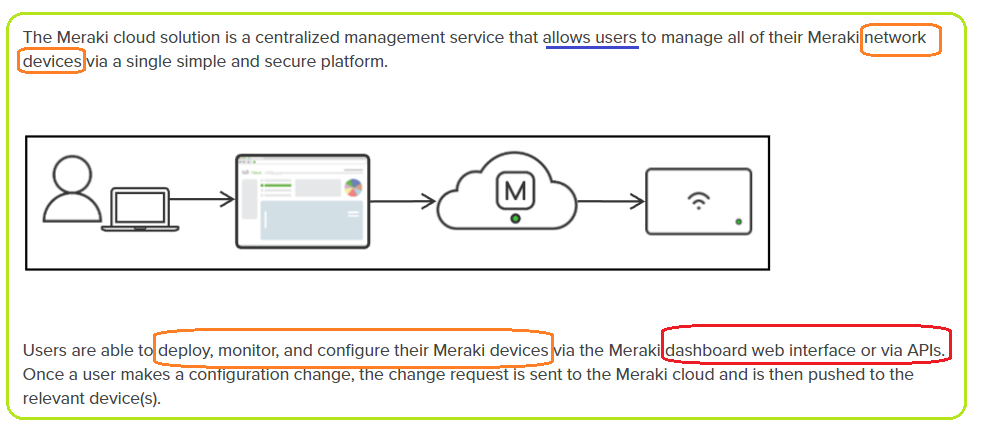
<https://documentation.meraki.com/MX/Client_VPN/Client_VPN_Overview>

**Wireless QoS and Fast Lane**

<https://documentation.meraki.com/MR/WiFi_Basics_and_Best_Practices/Wireless_QoS_and_Fast_Lane>

# Meraki Cloud Architecture

<https://documentation.meraki.com/Architectures_and_Best_Practices/Cisco_Meraki_Best_Practice_Design/Meraki_Cloud_Architecture>



### Meraki Cloud Architecture

The Meraki cloud is the backbone of the Meraki management solution. This "cloud" is a collection of **highly reliable multi-tenant servers** strategically distributed around the world at **Meraki data centers**. The servers at these data centers are powerful hosting computers comprised of many separate user accounts. They are called multi-tenant servers because the accounts share (equal) computing resources **on their host (the server).** However, even though **these accounts share resources**, Meraki ensures that customer information is kept secure by restricting organization access based on account authentication, as well as hashing authentication information such as user passwords or API keys.

#### **Data Centers**

**Customer management data** is replicated across independent same-region data centers in real time. The same data is also replicated in automatic nightly archival backups hosted by in-region third-party cloud storage services. The **Meraki cloud does not store customer user data**. More information about the types of data that are stored in the Meraki cloud can be found in the “Management Data” section below.

All Meraki services (the dashboard and APIs) are also replicated across multiple independent data centers, so they can failover rapidly in the event of a catastrophic data center failure.

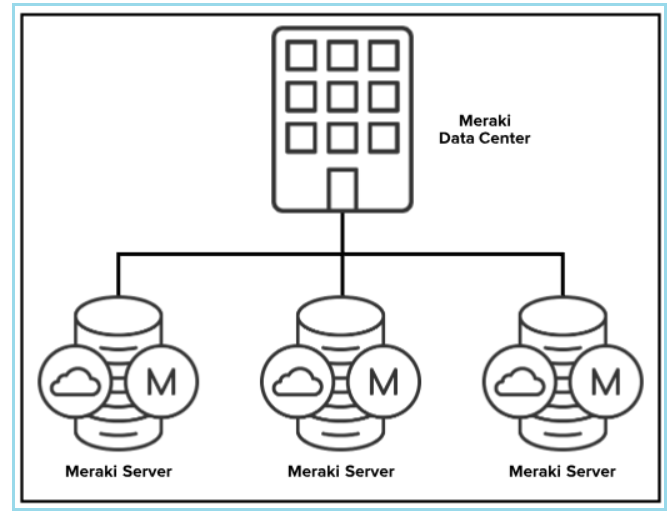
Meraki data centers are located around the world, enabling high-availability local data containment for data sovereignty in sensitive countries and regions, and high-speed connections to facilitate reliable cloud management communication. These data centers hold certifications such as PCI, SAS70 Type II/SSAE, PCI, and ISO27001. Additionally, all Meraki data centers undergo daily penetration testing by an independent third party. More key data center features include:

* 99.99% uptime service level agreement
* 24x7 automated failure detection
* Real-time replication of data between data centers
* All sensitive data (e.g., passwords) is hashed on servers

To learn more about monitoring, redundancy, disaster recovery, security, etc., reference our [data center design](https://meraki.cisco.com/trust#data-centers) page. More details about data center redundancy and reliability is covered in the “Reliability and Availability” section below.

**Data Center Storage**

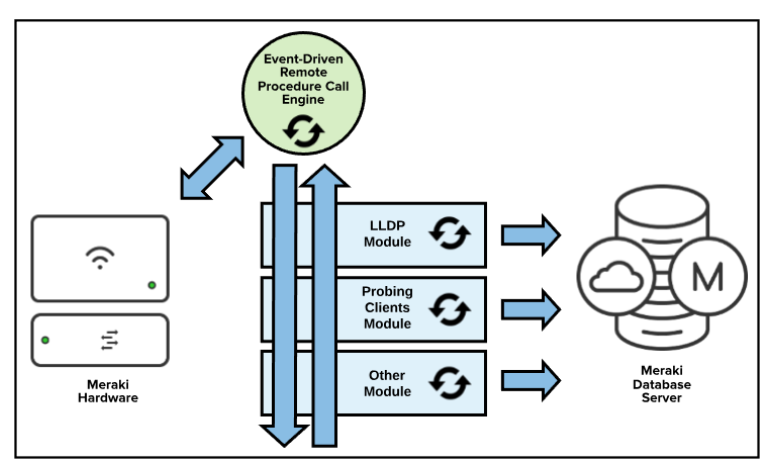
Meraki data centers contain active Meraki device configuration data and historical network usage data. These data centers house multiple compute servers, which are where customers’ management data is contained. These data centers do not store customers’ user data. These data types are covered in more detail in the “Data” section below



### Meraki Device-to-Cloud Communications

https://documentation.meraki.com/Architectures\_and\_Best\_Practices/Cisco\_Meraki\_Best\_Practice\_Design

Meraki uses an event-driven remote procedure call (RPC) engine for Meraki **devices to communicate to the dashboard** and for **Meraki servers to send and receive data**. Meraki hardware devices act as the server/receiver as the Meraki cloud initiates calls to the devices for **data collection** and **configuration deployment**. The cloud infrastructure is the initiator, so configurations can be executed in the cloud before the devices are actually online or even physically deployed.

In the event of cloud connectivity loss (which is most commonly caused by a local ISP or connection failure), the Meraki hardware device will continue to run with its last known configuration until cloud connectivity is restored.

#### Communication Process

**If a device is offline, it will continue to attempt to connect to the Meraki cloud until it gains connectivity.** **Once the device comes online, it automatically receives the most recent configuration settings from the Meraki cloud**. If changes are made to the device configuration while the device is online, the device receives and updates these changes automatically. These changes are generally available on the device in a matter of seconds. However, large quantities of changes may take noticeably longer to reach their devices. If no configuration changes are made by the user, the device continues to periodically check for updates to its configuration on its own.

As the device runs on the network, it will communicate device and network usage analytics back to the Meraki cloud. **Dashboard** analytics based on this information, in the form of graphs and charts, are updated regularly in the Meraki cloud and are displayed in the dashboard of users when they are viewing this information.

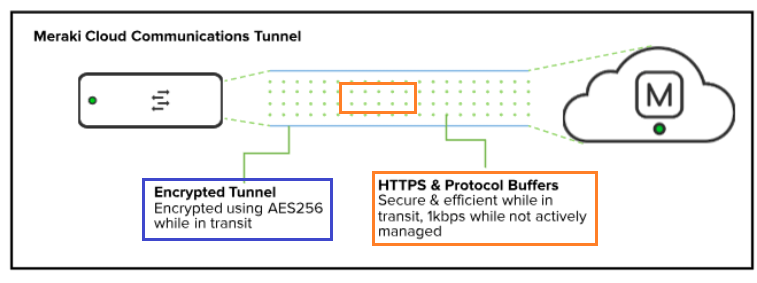
#### Configuration Containers

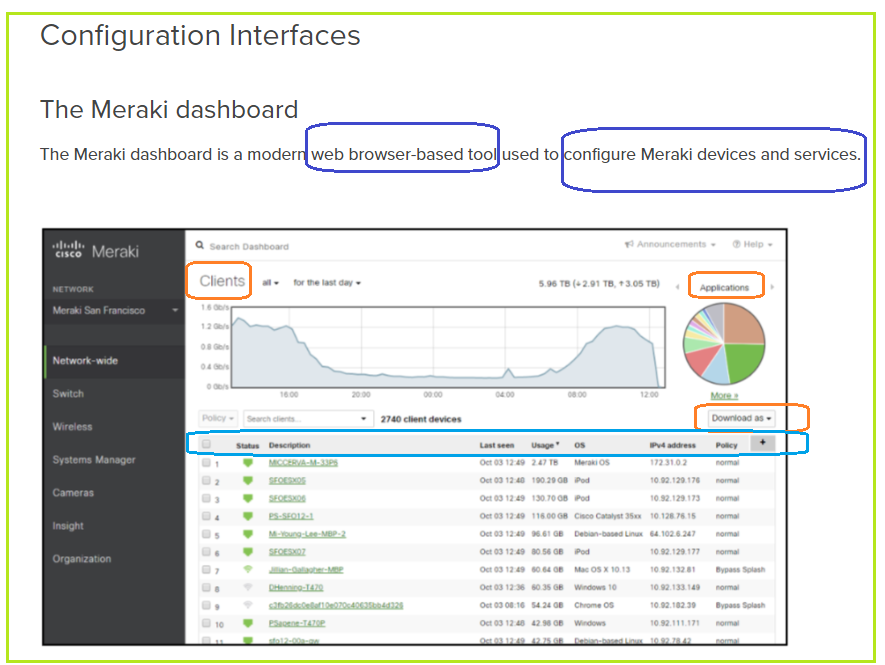
**Dashboard/API makes changes, send to server, server push the changes to devices**

Device configurations are stored as a container in the Meraki backend. **When a device configuration is changed by an account administrator via the dashboard or API**, the container is updated and then **pushed to the device** **(HOW)** the container is associated to via a secure connection. The container also updates the Meraki cloud with its configuration change for failover and redundancy.

#### Secure Device Connectivity

**For devices to communicate with the cloud**, Meraki leverages a proprietary lightweight encrypted tunnel using AES256 encryption while management data is in transit. Within the tunnel itself, Meraki leverages **HTTPS** and protocol buffers for a secure and efficient solution, limited to 1 kbps per device when the device is not being actively managed





The Meraki dashboard is the visual alternative to the traditional command line, which is used to manage many routers, switches, security devices, and more. Instead, Meraki puts all devices within networks in one place and allows users to apply changes in a simple, easy-to-use format.

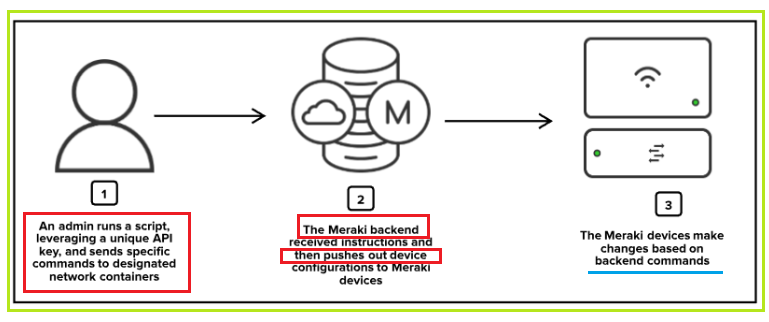
In addition to simplifying device management, the dashboard is also a platform for viewing network analytics, applying network permissions, and keeping track of users. The dashboard allows users to view camera streams, manage users’ mobile devices and computers, set content rules, and monitor upstream connections from a single place.

##### **Meraki APIs**

Meraki APIs provide control of the **Meraki solution in a programmable way**, enabling actions that may not be possible with the dashboard, or proving more granular control. Meraki APIs are **RESTful APIs using HTTPS** for transport and **JSON** for object serialization.

By providing open API accessibility, Meraki leverages the power of the cloud platform on a deeper level to create more efficient and powerful solutions. Through Meraki APIs, users can **automate deployments**, **monitor their networks**, and **build additional solutions on top of the Meraki dashboard**.

**API keys are tied to a specific user account** through the Meraki platform. If an individual has administrative access to multiple Meraki organizations, a single key can configure and control those multiple organizations.



**Reliability and Availability**

Meraki enables a high-availability (HA) architecture in multiple ways to ensure high serviceability to our customers. Network connections through our data centers are high in bandwidth and highly resilient. Shared HA structures ensure data is available in case of a localized failure, and our data center backup architecture ensures customer management data is always available in the case of catastrophic failure. These backups are stored on third-party cloud-based storage services. These third-party services also store Meraki data based on region to ensure compliance with regional data storage regulations.

**Data Center Uplink Connection High Availability**

Meraki constantly monitors the connections for integrity using multiple high-speed connections out of its data centers. Meraki network connectivity performs tests for DNS reachability to determine that integrity and data centers will failover to secondary links in the case of a degraded link.

**Meraki Server High Availability**

**A single device connects to multiple Meraki servers at the same time**, making sure all data is kept up-to-date in case there is need for a failover. This secondary Meraki server connection verifies device configuration integrity and historical network usage data in the case of a Meraki server failure.

#### n the event of server failure or connection loss, node connectivity can failover to the secondary server. Upon recovery of the primary server, the connection will be reestablished without noticeable impact to the connecting nodes.

#### Data Center Backup High Availability

**Meraki keeps active customer management data in a primary and secondary data center in the same region**. These data centers are geographically separated to avoid physical disasters or outages that could potentially impact the same region. Data stored in these data centers are synced in real time. In the case of a data center failure, the primary data center will fail over to the secondary data center with the most recent configuration stored.

#### 

#### Disaster Recovery Plan

The storage of customer management data and the reliability of its dashboard and API services are primary priorities for Meraki. To help prevent data loss in the event of a disaster, Meraki has multiple **major points of redundancy**. Each Meraki data center is paired with another data center in the same region. If a data center is completely wiped out, backups can be brought up within minutes at the other in-region data center. Next, if both data centers are impacted, nightly backups hosted in two different third-party cloud storage services, each with their own physical storage redundancies, can be used to recover data.

### Management Data

The Meraki cloud gathers and stores certain types of “management” data to enable its solutions. All forms of data are encrypted in transit to and from Meraki servers. There are four major types of data stored in the Meraki cloud:

**User records:**Includes account email and company name or other optional information such as user name and address.

**Test data:**devices, workflow, tests made by customers in the lp dashboard/API.

**Analytics data (reports):**Includes client, traffic, and location analytics data, providing visualizations and network insights into traffic patterns across customer sites.

**Customer-uploaded assets:**Includes custom floor plans and splash logos.

#### Server Data Segregation

**User data on Meraki servers is segregated based on user permissions**. **Each user account is authenticated based on organization membership**, meaning that each user only has access to information tied to the organizations they have been added to as users. **Organization administrators add users to their own organizations**, and those users set their own username and secure password. That user is then tied to that organization’s unique ID, and is then only able to make requests to Meraki servers for data scoped to their authorized organization IDs.

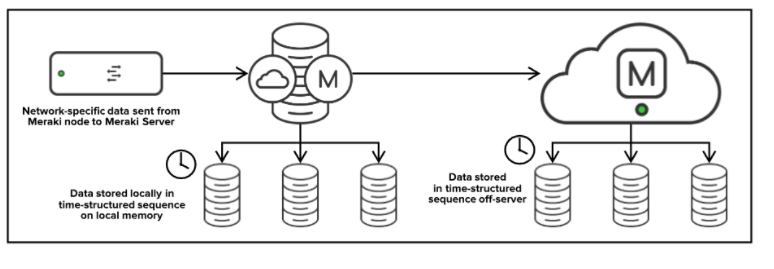
Additionally, **the Meraki development teams have separate servers for development and production, so Meraki never uses live customer data for testing or development**. Meraki user data is never accessible to other users or subject to development changes.

#### Network and Management Data Segregation

The Meraki “out of band” control plane separates management data from user data. **Management data** flows from Meraki devices (e.g. wireless access points, switches, and security appliances) to the Meraki cloud over a secure internet connection. **User data** (network traffic, web browsing, internal applications, etc.) does not flow through the Meraki cloud, and instead flows directly to the destination on the LAN or across the WAN.

#### Network Usage Data Retention

Meraki **stores management data** such as application usage, configuration changes, and event logs within the backend system. Customer data is stored for 14 months in the EU region and for 26 months in the rest of the world. Meraki data storage time periods are based on year-over-year reporting features in the dashboard (12-month periods), plus additional time to ensure data is removed from Meraki backups upon deletion (two months). Meraki uses a proprietary database system to build up easily searchable and referenceable data.



#### Segregated User Assets

Meraki stores **customer-uploaded assets** such as custom floor plans and splash logos. These items are leveraged within the Meraki dashboard for only that specific customer network and therefore are segmented securely based on standard user permissions tied to organization or network ID access. Only users authenticated to access the host network are able to access uploaded assets.

#### Data Security

All data transported to and from Meraki devices and servers is transported via a secure, proprietary communications tunnel (see the “Secure Connectivity” section above). Communications data is encrypted in transit via this tunnel. All client-management connections (dashboard/API) to the Meraki cloud have secure TLS encryption for all application traffic.

Additionally, Meraki data backups are fully encrypted using AES256 and have restricted access (see the “Physical and Operational Internal Security” section).

#### Data Privacy

Connecting to a cloud solution entails storing specific data in the cloud for easy use and access. To maintain integrity and security, a cloud infrastructure must take into account the sensitivity and compliance rules of that data. Specific industries and geographies have laws to protect the user data that Meraki addresses through our flexible cloud infrastructure.

Meraki embeds privacy by design in its product and feature development as well as business practices. Privacy is an integral piece of the Meraki design process and is a consideration from initial product design all the way through to product implementation. Meraki offers a full suite of privacy-driven features to all customers globally. These features allow our customers to manage privacy requirements and help support their privacy initiatives. Customers can read more about some of the Meraki privacy features in our [Data Privacy and Protection Features](https://documentation.meraki.com/General_Administration/Privacy_and_Security/Meraki_Data_Privacy_and_Protection_Features) article.

### Security

Customer security is a top priority for Meraki. Heavy investments in tools, processes, and technologies keep our users and their networks safe, including features like **two-factor authentication for dashboard access** and out-of-band cloud management architecture.

In addition to Meraki and Cisco’s internal security teams, Meraki leverages third parties to provide additional security. Precautions such as daily third-party vulnerability scans, application testing, and server testing are embedded in the Meraki security program. Meraki additionally started a vulnerability rewards program for both hardware and software, which encourages external researchers to collaborate with our security team to keep our infrastructure and customers safe. More information about this program can be found on our [Bugcrowd program page](https://bugcrowd.com/ciscomeraki" \t "_blank).

Meraki intelligent security infrastructure eliminates the management complexities, manual testing, and ongoing maintenance challenges that lead to vulnerabilities. The intuitive and cost-effective security features are ideal for network administrators, while powerful and fine-grained administration tools, account protections, audits, and change management appeal to chief information security officers.

###### Number of User Accounts

**Account Scope General Recommendation**

* 2 or more user accounts per owner
* 1 user account per network admin

Your Meraki account is your first step in building a Meraki solution, and it will also be your only method of gaining access to your devices, and distributing access to other users.

###### Number of Organizations per Account

**Organization Scope General Recommendation**

* One organization per customer

OR

* One organization per service

If you are the customer who will be using and managing Meraki equipment, it is likely that you will only need a **single organization**. Each organization is simply a container for your networks, and a single-organization model is generally the most simple solution if it's realistic for your deployment.

Additionally, it is important to consider **Meraki server and data center limits**. Meraki server architecture is a multi-tenant solution that hosts multiple customers on the same hardware with secure permissions-based segmentation among them. The maximum scale supported in a single organization is 25,000 physical Meraki devices. *If a single business intends to have* more than 25,000 Meraki devices in their solution, they are strongly encouraged to work with their account team to design a deployment strategy across multiple organizations.

###### Number of Devices per Network

The number of Meraki devices (MX, MS, MR, MV, etc., not user devices) per network is a much more variable number that does not have a general recommendation. It will vary from case to case.

Note that there is a**limit of 1000 devices per network**. Networks exceeding this number should be split. However, it is generally uncommon for networks to approach this number unless they have a very large number of cameras or wireless access points. If this is the case, it is recommended to split the networks based on physical areas or use cases.

###### **API Key/Token**

API keys/tokens are tied to **the access of the user** who created them.  Programmatic access should only be granted to those entities who you trust to work within the organizations they are assigned to. Because **API keys/Token are tied to accounts** within an organization.

##### Tagging

Tagging is a way to group or identify devices, networks or ports for specific use cases. These tags can be used to search, filter, identify or assign access to particular functions. The following items can have network tags applied to them:

* Networks
* Meraki devices
* Switch ports
* Systems Manager devices

**Tagging networks** allows specific admins to have network level configuration access without organization-wide access. Access can be scoped based on network tags, which allows for much more granular access control. This is most commonly used for assigning permissions to local IT admins that are not "super users."  Additionally, network tagging allows "visibility-only" roles for users to see the most relevant application data. This is most commonly used for managers interested in the traffic usage of their network, but may not want to make configurations.

**Meraki device tags** are used to easily search for certain types of devices on your networks, based on **user-defined attributes**. Common device tags may include **locations** like "1stFloor," "Dorms," "Classroom," etc., or  **attributes of the device** like "LeftRack" or "MiddleRack" for MX and MS models or "CeilingMount," "WallMount," for MR or MV models. Additionally, APs can be tagged for specific VLANs. There is an option to tag traffic with a particular VLAN on specific APs based on device tags.

**Systems Manager device tags** are used to logically group end-user devices together and associate them with applications and profiles. Users may be given a tag for a certain application that should only be installed on their devices, or a certain security level that should only apply to them.

**Switch port tags** allow administrators to set granular port management privileges. Organization administrators could use port tags to give read-only admins configurations access and packet capture capability on specific ports.

##### Administrators

There are two basic types of dashboard administrators: **Organization administrators** and **Network administrators**.

* **Organization administrators (Organization ID)** have complete access to their organization and all its networks. This type of account is equivalent to a root or domain admin, so it is important to carefully maintain who has this level of control.
  + **Organization - Read-only**: The user is able to access/view most aspects of network and organization-wide settings, but is unable to make any changes.
  + **Organization - Full**: The user has full administrative access to all networks and organization-wide settings. This is the highest level of access available.
* **Network(Group) administrators – Group ID** have access to individual networks and their devices. These users can have complete or limited control over their network configuration, but do not have access to organization-level information (licensing, device inventory, etc).
  + **Network - Guest ambassador**: The user is only able to see the list of Meraki authentication users, add users, update existing users, and authorize/de-authorize users on an SSID or Client VPN. Ambassadors can also remove wireless users, if they are an ambassador on all networks.
  + **Network - Monitor-only**: The user is only able to view a subset of the **Monitor** section in the dashboard and no changes can be made. This can be useful for proving networking monitoring access to customers in service provider deployments.
  + **Network - Read-only**: The user is able to access most aspects of a network, including the **Configure** section of the dashboard, but no changes can be made.
  + **Network - Full**: The user has access to view all aspects of a network and make any changes to it.

##### SNMP vs API for Device Status Reporting

Many deployments will find that they benefit from some type of device reporting, or may have some kind of mechanism in place for **monitoring device status**. Options for monitoring devices include standard dashboard monitoring, SNMP reporting and API device status reporting. SNMP is an available option for users who are used to using an SNMP solution, but for large deployments (20,000+ devices), we highly recommend relying on device status reporting via the API for scalability. Smaller to medium-sized deployments may also find that an API solution for device reporting better suits their needs, so the option should be considered.

You can read more about our available API endpoints for device status reporting in our [API docs](https://dashboard.meraki.com/api_docs).

Meraki differentiates itself through its firmware delivery using the Meraki cloud platform, by providing an exceptionally swift and reliable way to deliver firmware upgrades. The results are evident in our users’ impressive firmware adoption rates. Even given the options for finer controls, the vast majority of our users adopt and run on our latest firmware builds almost immediately after stable release candidates are available. Our extensive testing and our beta adoption process ensures that we deliver reliable builds at a regular cadence, delivering up-to-date security and stability.

### Meraki Firmware Conventions

Meraki firmware nomenclature is the same across products and consists of a major and minor number as part of the name. The firmware version is named using the format given below:

**<Product Name> <Major Firmware Version>.<Minor Firmware Version>**

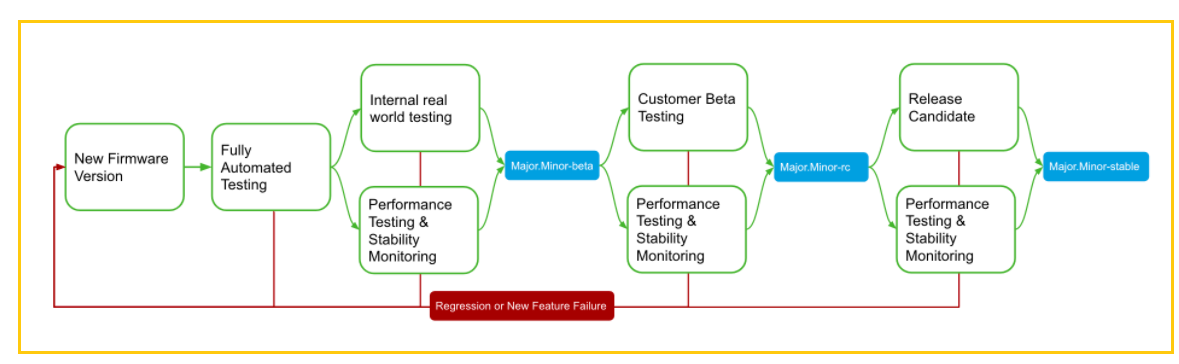
1. **Major Versions**

A new major firmware is released with the launch of new products, technologies and/or major features. New major firmware may also include additional performance, security and/or stability enhancements.

1. **Minor Versions**

A new minor firmware version is released to fix any bugs or security vulnerabilities encountered during the lifecycle of a major firmware release.

Meraki firmware release cycle consists of three stages during the firmware rollout process namely beta, release candidate (RC) and stable firmware. This cycle is covered in more detail in the [Meraki Firmware Development Lifecycle section](https://documentation.meraki.com/Architectures_and_Best_Practices/Cisco_Meraki_Best_Practice_Design#Meraki_Firmware_Development_Lifecycle-243) of this document.



#### Alpha Pre-Release

With all new Meraki firmware including both major and minor releases, we start out every new build by running it through our full alpha testing process. Before any release hits our users’ hands, we validate the release by running it through our ever-expanding testing suites, and check for regressions or new features that are not performing as expected. Each product line has automated and manual testing specific to the product, that are designed to ensure Meraki minimizes the chance of regressions as we continue to create and expand on our software feature set.

As part of our core philosophy, after a new build has successfully passed the testing phase, we deploy the new firmware release on our own personal and engineering networks. We believe it is important that we deploy and run our own firmware before any of our customers deploy our firmware. During this process we will run this firmware in our real world deployments for one or more weeks before we consider releasing the build as a new beta version.



If a build successfully passes all of our release criteria, we will start to make the new build available to our customer base. If any issues are discovered that need to be resolved, we will start the process over once the issue has been addressed before moving the release forward. In some more rare cases, we will move forward with a build with a known regression, due to complexity or timing of the fix, and in this scenario we will note the regression in the release notes for that version.

#### Beta Release

Firmware is made available for production use at first under "Beta." Often customers will run beta firmware in their production network to take advantage of new features and bug fixes. Beta firmware has already gone through internal regression, stability, and performance testing to limit risks when applied to production networks. Customers that opt into beta firmware via the **Try beta firmware** configuration option on dashboard will be automatically notified and scheduled to upgrade to these versions as they are released. These upgrades can be canceled, modified, and reverted using the [**firmware upgrades tool**](https://documentation.meraki.com/General_Administration/Firmware_Upgrades/Managing_Firmware_Upgrades) in the dashboard. Customers can also manually upgrade their networks at any time to beta firmware by using the firmware upgrade tool. Beta firmware can be considered analogous to “Early Deployment” firmware seen in other products in the industry.

The latest beta firmware is fully supported by our Support and Engineering teams. Older betas are supported with best effort; an upgrade to the latest beta will ensure full support.

#### Stable Release Candidate

As a new firmware version matures from beta, it has the opportunity to graduate into a stable release candidate. A formal review of the beta firmware’s success is conducted by our software and product teams. Key performance indicators (KPIs) for quantifying firmware quality are analyzed including open support cases & engineering issues, firmware adoption, and stability metrics. After the formal review, a beta may be reclassified as a "Stable Release Candidate." At this point the firmware version will be indicated as such in the [firmware upgrade tool](https://documentation.meraki.com/General_Administration/Firmware_Upgrades/Managing_Firmware_Upgrades). Once a new stable release candidate is available, Engineering will begin scheduling a limited set of customers for upgrade. These upgrades can be canceled, modified, or reverted using the firmware upgrade tool as well.

The latest stable release candidate firmware is fully supported by our Support and Engineering teams. Older stable release candidates are supported with best effort; an upgrade to the latest beta, stable release candidate, or stable will ensure full support.

#### Stable Release

A stable release candidate matures into a stable version over time as it is slowly rolled out to devices globally. When the Meraki install-base hits a specified threshold for a major version (roughly 20% of nodes), that firmware revision will be promoted to stable, pending a final formal review. For point releases, the determination will be made on a case-by-case basis.

Again, the same KPIs are analyzed as used in the stable release candidate review. Upon completion of these processes the firmware can be promoted to "Stable." After promotion, stable versions can be applied by any customer via the [firmware upgrade tool](https://documentation.meraki.com/General_Administration/Firmware_Upgrades/Managing_Firmware_Upgrades) on dashboard. The latest stable version is also the version that is used for all newly created dashboard networks for a particular device.

# 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](https://developer.cisco.com/meraki/explore/) section for open source projects, or browse the [Marketplace](https://apps.meraki.io/) for partner solutions

## **What's New in v1**

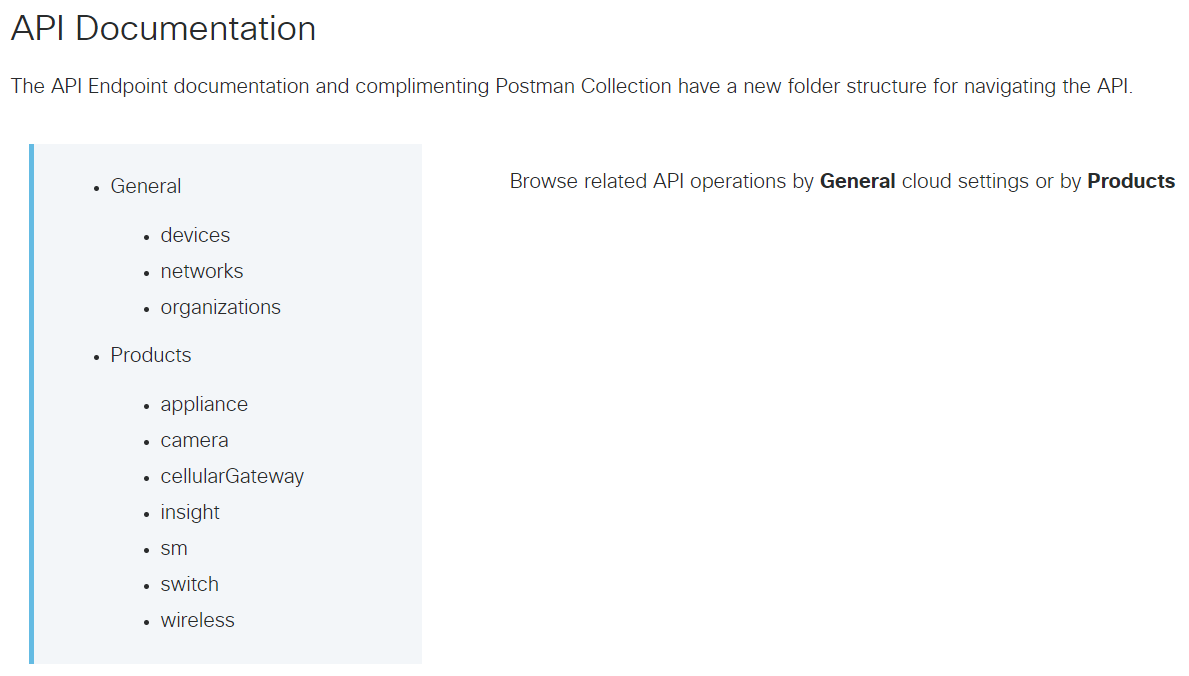
The Dashboard API has evolved significantly, providing **hundreds of endpoints** to manage your Meraki networks!

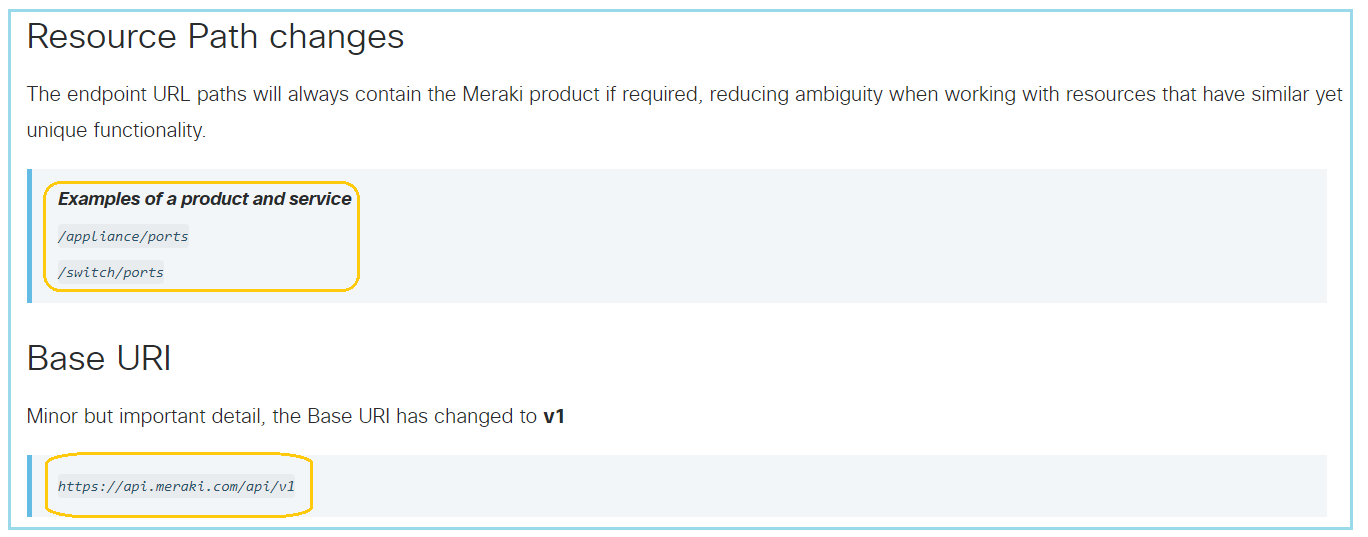
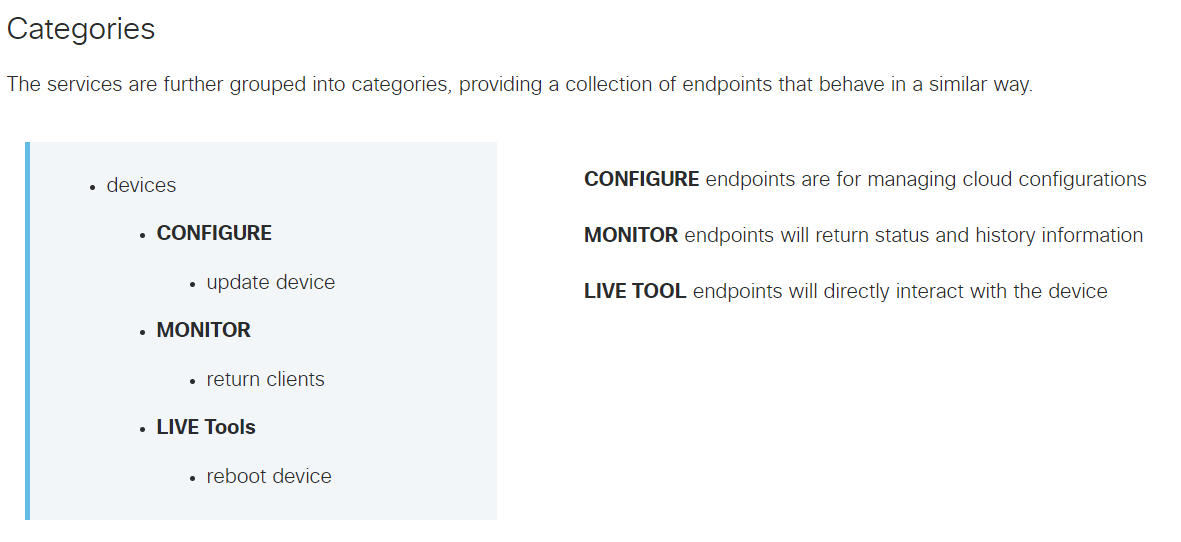
We want to do so much more. But in order for us to include many of these new features or improvements, we need to break a few things.

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

The API documentation, Postman collection, and Python library will remain synced and up-to-date with improved navigation and features.

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





SDKs

Going forward, the custom Meraki [Python library](https://developer.cisco.com/meraki/api-v1/#python) will be the recommended SDK for simplified API scripting. The previously auto-generated Python, Node.js, and Ruby SDKs for v0 will remain in the Meraki GitHub but will no longer be maintained

# Global Meraki Path Parameter IDs

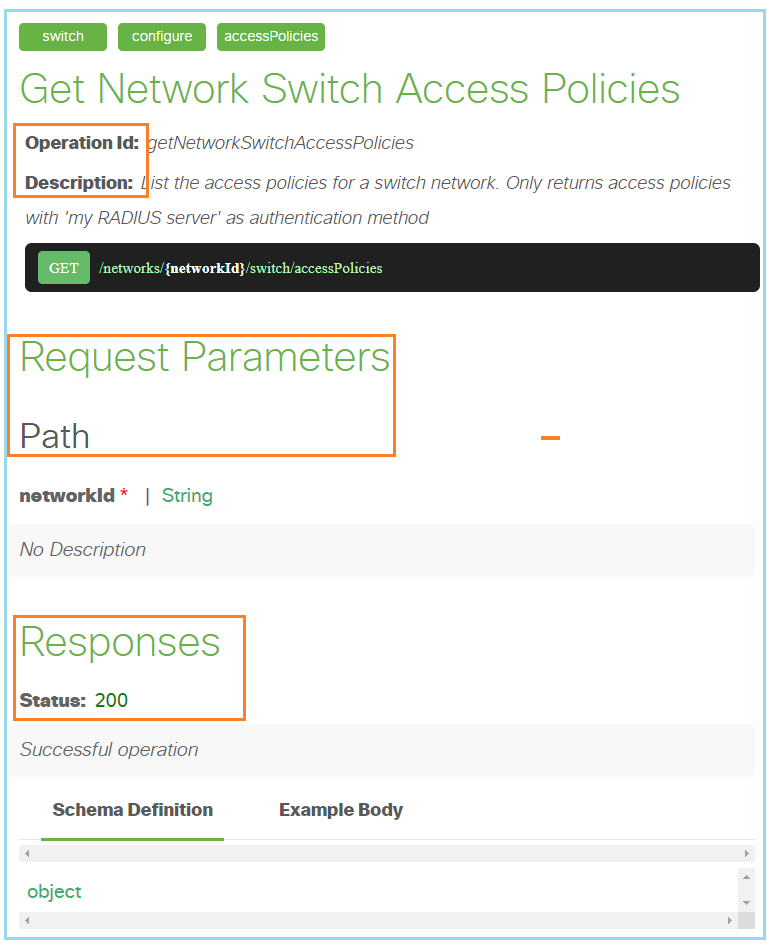
# <https://developer.cisco.com/meraki/api-v1/#!path-parameters/global-meraki-path-parameter-ids>

Each service API path parameter ID has related API endpoint operations which can be used to get the ID's value.

For example, the following API path requires an organizationId for its scope and an adminId for its service parameter.

**/organizations/{organizationId}/admins/{adminId}**

Some endpoints will return the parameter named as id in the body, whereas other operations may return the named ID such as groupId or floorPlanId. This table provides a helpful map of the global Meraki path parameters and their respective operations. It also includes the ID parameter name you should expect to use for related API operations.



**Meraki Dashboard API Python Library**

The Meraki Dashboard API [Python library](https://github.com/meraki/dashboard-api-python/) provides all current Meraki [Dashboard API](https://developer.cisco.com/docs/meraki-api-v1) calls to interface with the Cisco Meraki cloud-managed platform. The library is supported on Python 3.7 or above, and you can install it via [PyPI](https://pypi.org/project/meraki/" \t "_blank):

CODE SNIPPET

pip install meraki

Features

While you can make direct HTTP requests to dashboard API in any programming language or REST API client, using a client library can make it easier for you to focus on your specific use case, without the overhead of having to write functions to handle the dashboard API calls. The Python library can also take care of error handling, logging, retries, and other convenient processes and options for you automatically.

* Support for all API endpoints, as it uses the [OpenAPI specification](https://api.meraki.com/api/v1/openapiSpec" \t "_blank) 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](https://developer.cisco.com/docs/meraki-api-v1/#!rate-limit) 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

## **Setup**

1. Enable API access in your Meraki dashboard organization and obtain an API key ([instructions](https://documentation.meraki.com/zGeneral_Administration/Other_Topics/The_Cisco_Meraki_Dashboard_API))
2. **Keep your API key safe and secure, as it is similar to a password for your dashboard**. If publishing your Python code to a wider audience, please research secure handling of API keys.
3. Install the latest version of [Python 3](https://wiki.python.org/moin/BeginnersGuide/NonProgrammers)
4. Use pip (or an alternative such as easy\_install) to install the library from the Python [Package Index](https://pypi.org/project/meraki/):
   * **pip install meraki**
   * If you have both Python3 and Python2 installed, you may need to use pip3 (so pip3 install meraki) along with python3 on your system
   * If meraki was previously installed, you can upgrade to the latest non-beta release with **pip install --upgrade meraki**
5. If you clone this repository and want to use v1 locally, rename the folder "meraki\_v1" to "meraki", replacing the v0 contents there. You can also specify the version of the library when installing with pip:
   * See the full [release history](https://pypi.org/project/meraki/#history) to pick the version you want, or use pip install meraki== without including a version number to display the list of available versions
   * v0 versions of the Python library begin with 0 (0.**x**.**y**), and v1 versions begin with 1 (1.0.0b**z** for beta)
   * Specify the version you want with the install command; for example: pip install meraki==0.x.y for v0 or pip install meraki==1.0.0bz for v1 beta
   * You can also see the version currently installed with pip show meraki

## **Usage**

1. Export your API key as an [environment variable](https://www.twilio.com/blog/2017/01/how-to-set-environment-variables.html), for example:

SHELL

export MERAKI\_DASHBOARD\_API\_KEY=093b24e85df15a3e66f1fc359f4c48493eaa1b73

1. Alternatively, define your API key as a variable in your source code; this method is not recommended due to its inherent insecurity.
2. Single line of code to import and use the library goes at the top of your script:

import meraki

1. Instantiate the client (API consumer class), optionally specifying any of the parameters available to set:

dashboard = meraki.DashboardAPI()

1. Make dashboard API calls in your source code, using the format client.scope.operation, where client is the name you defined in the previous step (**dashboard** above), scope is the corresponding scope that represents the first tag from the OpenAPI spec, and operation is the operation of the API endpoint. For example, to make a call to get the list of organizations accessible by the API key defined in step 1, use this function call:

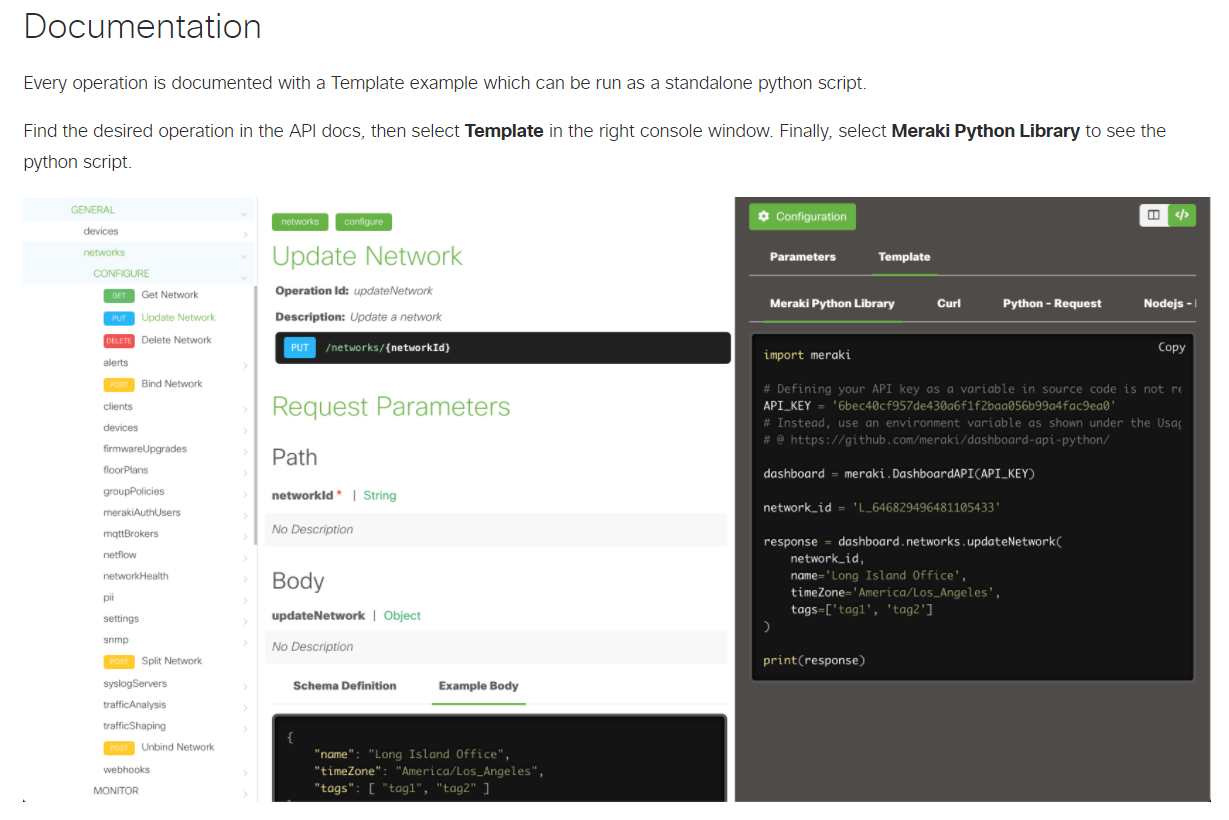
my\_orgs = dashboard.organizations.getOrganizations()

1. If you were using this module versions 0.34 and prior, that file's functions are included in the legacy.py file, and you can adapt your existing scripts by replacing their from meraki import meraki line to **import meraki**

[**https://developer.cisco.com/meraki/api-v1/#!update-network**](https://developer.cisco.com/meraki/api-v1/#!update-network)

[**https://developer.cisco.com/meraki/api-v1/#!getting-started/base-uri**](https://developer.cisco.com/meraki/api-v1/#!getting-started/base-uri)

For a full working script that demos this library, please see and run the **org\_wide\_clients\_v1.py** file included (in **examples** folder). That code collects the clients of all networks, in all orgs to which the key has access. No changes are made, since only GET endpoints are called, and the data is written to local CSV output files.



# Certificate Requirements for TLS

<https://documentation.meraki.com/General_Administration/Other_Topics/Certificate_Requirements_for_TLS>

<https://api.meraki.com/api/v1/openapiSpec>

<https://developer.cisco.com/meraki/explore/>

<https://developer.cisco.com/meraki/>