Cisco Meraki Dashboard API

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 that can be used for many purposes. 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(s)
- Build your own dashboard for store managers, field techs, or unique use cases

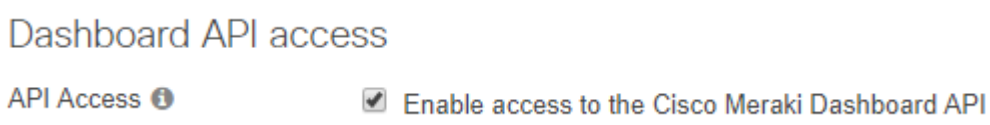
Note: API call volume is rate-limited to ten calls per second, per organization.

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 be 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.**



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.

Note: there is a limit of only two API keys per profile. After generating two keys, the "Generate API key" disappears from the dashboard view. You need to revoke one of the generated keys before you can generate a new one.

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

Using the Dashboard API

API Requests

Every request must specify an API key via a request header. Further, the API key must be specified in the URL. The API will return 404 (rather than a 403) in response to a request with a missing or incorrect API key in order to prevent leaking the existence of resources to unauthorized users.

> X-Cisco-Meraki-API-Key: <secret key>

The API version must be specified in the URL:

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

Note: For organizations hosted in the China dashboard, use **api.meraki.cn**

Once an API version is released, we will make only backwards-compatible changes to it. Backwards-compatible changes include:

- Adding new API resources
- Adding new optional request parameters to existing API methods
- Adding new properties to existing API responses
- Changing the order of properties in existing API responses

Identifiers in the API are opaque strings. A ***<network_id>***, for example, might be the string “126043”, whereas an ***<order_id>*** might contain characters, such as “4S1234567”. Client applications must not try to parse them as numbers. Even identifiers that look like numbers might be too long to encode without loss of precision in Javascript, where the only numeric type is IEEE 754 floating point.

Verbs in the API follow the usual REST conventions:

- GET returns the value of a resource or a list of resources, depending on whether an identifier is specified.
  - GET of ***/v1/organizations*** returns a list of organizations

    - GET of ***/v1/organizations/<org_id>*** returns a particular organization
- POST adds a new resource
    - POST to ***/v1/organizations/<org_id>/admins***
    - POST performs other non-idempotent changes
- PUT updates a resource
    - PUT is idempotent; it updates a resource, creating it first if it does not already exist
    - PUT should specify all the fields of a resource; the API will revert omitted fields to their default value
- DELETE removes a resource

> **Note**: Call volume is limited to ten calls per second, per organization.

Status and Error Codes

Responses from the API generally use standard HTTP status codes. Some examples:

- **400: Bad Request** - You did something wrong, e.g. a malformed request or missing parameter
- **403: Forbidden** - You don't have permission to do that
- **404: Not found** - No such URL or you don't have access to the API or organization at all
- **429: Too Many Requests** - You submitted more than ten calls in one second to an organization, triggering rate limiting. This also applies for API calls made across multiple organizations that triggers rate limiting for one of the organizations.

If the response code is not specific enough to determine the cause of the issue, error messages will be included in the response in JSON format, for example:

```
{
    "errors": [
       "VLANs are not enabled for this network"
    ]
}
```

Examples

Example requests and responses follow. Please note that these are not comprehensive; for a comprehensive list of API endpoints and their details, please see the online documentation URL above.

Organizations

Retrieve the list of organizations for the API key specified in X-Cisco-Meraki-API-Key:

```
GET https://api.meraki.com/api/v1/organizations
Response code: 200
Response body: [{"id":"<org_id>","name":"<org_name>" }]
```

Retrieve metadata about a specific organization:

```
GET https://api.meraki.com/api/v1/organizations/<org_id>
```

```
Response code: 200
Response body: { "id": <org_id>, "name": "Test Organization" }
```

Note also that most HTTP libraries and clients will not follow a redirect with an HTTP verb other than GET, and may instead follow the redirect but substitute GET for the HTTP verb. As a result, you may find a redirect on DELETE, POST, or PUT will look as though a GET was requested. To avoid this, first perform a GET on the organization you are working with and store the URL you are working with, then use that URL for subsequent requests (particularly those that modify state).

Create a new organization:

```
POST https://api.meraki.com/api/v1/organizations
Request body: { "name": "Second Test Organization" }
Response code: 201
Response body: { "id": <new_org_id>, name: "Second Test Organization" }
```

The new organization will be unconfigured with the exception that the admin who created it will have full organization write access on the new organization. Clients of the API can add additional admins and then remove the creating admin from the new organization, if desired.

Note: The API will not allow clients to remove the last admin with full organization write access, as doing so could lead to an unconfigurable organization.

Networks

List the networks in an organization:

```
GET https://api.meraki.com/api/v1/organizations/<new_org_id>/networks
Response code: 200
Response body: [ { "id": <network_id>, "name": "Test Network", "organization_
id": <new_org_id>, "type": "wireless",
   "timeZone": "America/Los_Angeles", "tags": " test "}]
```

Create a new network:

```
POST https://api.meraki.com/api/v1/organizations/<org_id>/networks
Request body: { "name": "Test Network 2", "organizationId": <org_id>, "type":
"appliance"}
Response code: 201
Response body: { "id": <network_id>, "name": "Test Network 2", "organization_
id": <org_id>, "type": "appliance", "tags": null }
```

Valid network types are **appliance, switch, wireless, systemsManager** or **camera**. Use a combination of these to create a combined network. For example, **"type": "appliance wireless"**.

Licenses and Devices

Claim a device, license key, or order into an organization. When claiming by order, all devices and licenses in the order will be claimed. Licenses will be added to the organization and devices will be placed in the organization's inventory. These three types of claims are mutually exclusive and cannot be performed in one request. The **licenseMode** param can be either **renew** or **addDevices**. **addDevices** will increase the license limit while **renew** will extend the amount of time until expiration. This parameter is required when claiming by licenseKey. Please see this article for more information.

```
POST https://api.meraki.com/api/v1/organizations/<org_id>/ claim
Request body: { "order": "4CXXXXXXX" }
Response code: 200
```

```
Response body: <empty>
```

SNMP Settings

Set the SNMP settings for an organization. Note that the community string for SNMPv2 is in the response. It can also be retrieved via a GET to this same URL.

```
GET https://api.meraki.com/api/v1/organizations/<org_id>/snmp
Request body: { "v2cEnabled": true, "v3Enabled": true, "v3AuthMode": "SHA",
"v3AuthPass", <secret>, "v3PrivMode": "AES128", "v3PrivPass": <secret>,
"allowedIPs": ["5.6.7.8"] }
Response code: 200
Response body: { "v2Enabled": true, "v2CommunityString": <secret>,
"v3Enabled": true, "authMode": "SHA", "authPass", <secret>, "privMode":
"AES128", "privPass": <secret>, "allowedIPs": ["5.6.7.8"] }
```

Administrators

List the administrators for an organization:

```
GET https://api.meraki.com/api/v1/organizations/<org_id>/admins
Response code: 200
Response body: [ { "name": "Jane Doe", "email": "jane@doe.com", "orgAccess":
"full","id": <admin_id> } ]
```

Organization access can be **full, read-only,** or **none**.

Add a new administrator with rights to only one network:

```
POST https://api.meraki.com/api/v1/organizations/<org_id>/admins/
Request body: { "name": "John Doe", "email": "john@doe.com", "orgAccess":
"none", "networks": [{ "id": <network_id>, "access": "full" }
Response code: 201
Response body: { "id": <admin_id>, "name": "John Doe", "email": "john@doe.
com", "orgAccess": "none", "networks": [{ "id": <network_id>, "access": "full"
}]
```

Valid access levels for networks are **full, read-only, monitor-only,** and **guest-ambassador**. Clients can also give administrators access by tag by substituting the **networks** field for **tags** such as:

```
"tags": [{ "tag": "foo", "access": "read-only" }],
```

Valid access levels for tags are the same as for networks. Clients can update administrators using PUT and specifying an **admin_id**.

Delete an administrator:

```
DELETE https://api.meraki.com/api/v1/organizations/<org_id>/admins/<admin_id>
Response code: 204
Response body: <empty>
```