# Meraki-API-V1

This module allows you to interact and manage your Meraki organization using Powershell. This module uses the Version 1 REST API.

[!WARNING] I have enabled the some of the writable functions that have been tested (New-, Add-, Set- Remove-). Please make sure you are 100%, absolutely, without a doubt, sure you know what you are doing! I hold no responsibility if you damage your Meraki Organization/Networks. You are warned!

[Module Reference](https://clifra-jones.github.io/Meraki-API-V1/docs/reference.html)

[Release notes](https://clifra-jones.github.io/Meraki-API-V1/docs/releaseNotes.md)

This module aims to follow Powershell best practices.

Powershell best practices discourage the use of pluralized function names, I have made the effort to remove most of the pluralized functions that were in this module. I have created aliases to the previous pluralized functions. This should prevent any scripts you already have from breaking. You should make the effort to modify your scripts to use the new non-pluralized functions. There are still some pluralized function, primarily in the Organization functions. The organization functions use different endpoint URIs.

Example:

* Get-MerakiNetworks which is an Organization endpoint.
* Get-MerakiNetwork which is a Network endpoint.

Many of the objects returned by the functions in this module provide additional properties that facilitate piping the results to other commands. Properties such as NetworkID, Serial, InterfaceId, etc are added to the results from the API methods that do not contain them.

Piping is not supported for certain function. These include all Remove- functions and some Set- functions that the module creates unique identifiers for each item returned. This is done for safety reasons on the Remove- functions. For module provided unique identifiers I cannot guarantee that the item referred to by the identifier is the same item configuration across different networks, devices, etc. Again this is for safety and data integrity reasons.

As stated above, writable function are YOUR responsibility! There is no UNDO, there is no recovery! If you delete a network, it is gone, you will have to rebuild it manually! The same applies to any configuration in your organization! While this module provides a convenient method of doing something like cleaning up old networks that are no longer is use much faster than doing it manually through the dashboard, you MUST use care to make sure all the network IDs you are providing are in fact meant to be removed! You can seriously damage your Meraki organization and possibly find yourself unemployed! Neither myself, or Cisco can be held responsible for your actions!

## Secure API Key storage

The module now supports storing your API keys in Secure Storage. This requires the following modules to be installed on your system:

* Microsoft.Powershell.SecretsManagement
* Microsoft.Powershell.SecretStore

These are now required modules for this module. You will need to install them even if you do not use the secure key storage feature.

You will need to create a vault to store your keys. SecretStore does not support multiple named vaults. Doing so just duplicates the vault! See: [SecretStore issue 58](https://github.com/PowerShell/SecretStore/issues/58" \l "issuecomment-824216690) If you currently have a secret store vault on your system you do not need to create a new one. This vault will be used to store your API key.

### Configure the Secret Store

This setting must be done under the user profile of the user that will run the scripts. Secret Stores and Vaults ONLY exist in the User Profile. If you are configuring for Automation you must be logged in (or run as) as the user who will run the scripts.

You need to configure the Secret stores authentication method. We do this with the Set-SecretStoreConfiguration function. There are two parameters that must be set:

* Authentication: Either None or Password
* Interaction: Either None or Prompt

If you are configuring the store for your personal use you should set Authentication to Password and Interaction to Prompt.

If you are configuring the vault for automation to execute schedules tasks set Authentication and Interaction to None. This should only be done on a server under a service account with a secure password.

Example, Configure for personal use:

Set-SecretStoreConfiguration -Authentication Password -Interaction Prompt

You will be prompted for enter a password to unlock the vault and required to confirm it.  
  
Example, Configure for automation:

Set-SecretStoreConfiguration -Athentication: None -Interaction None

You will still be prompted to create a password then prompted to not use the password to unlock the vault.

### Register a new vault

As stated above you should only create 1 vault. The name of the vault doesn’t matter but it should be set as the default.

Example:

Register-SecretVault =Name ‘MyVault’ -ModuleName Microsoft.Powershell.SecretStore -DefaultVault

See the Usage section below for instruction for creating you API key and configuring your profile.

## Examples

There are a few examples in the Examples folder under the module folder. You can refer to these examples for various techniques used with this module. The example DocumentMerakiNetwork.ps1 requires the module Import-Excel to function. You will need to install this module to use this example.

## INSTALLATION

The module is now available from the Powershell Gallery. This will always be the latest stable version of this module.

Install-Module Meraki-API-V1

There is a test branch you can clone if you want to test out any new features not in the current production version. The branch name is 2.0\_test.

## USAGE

API Access must be enabled on your Meraki Dashboard.

You will need to have a Meraki API key. You can get your key by logging into your Meraki Dashboard, go to your profile and generate your API key. Save this key in a safe place.

Once you have your API key you need to obtain the Organization ID for the Organizations you have access to. You can do this with the Get-MerakiOrganizations function.

Open Powershell

Import-Module Meraki-API-V1  
Get-MerakiOrganizations -APIKey '{key string}'

Configure your user profile to use the API.

You must configure your profile to use the API module. To do this use the Set-MerakiAPI function. This function creates a file in the folder .meraki named config.json

API Keys can be used across organizations. You API key identifies you, not your organization. Many functions will retrieve data independently of the organization.

Some command are organization dependent and will only retrieve data for a specific organization. Either the default organization or one provided by the -ProfileName or -OrgId parameters.

Set the API Key and organization Id. This sets the API Key and the Organization Id in the default profile.

Set-MerakiAPI -APIKey '{key string}' -OrgID 'XXXXXX'

Set the API Key, Organization Id and name the profile. If there is not a default profile this will also be the default profile.

Set-MerakiAPI -APIKey '{key string}' -OrgId 'XXXXXX' -ProfileName 'ProfileName'

Set the API key and Organization ID and use a Secure Key.  
You must have previously setup a vault and configured the Secret Store.

Set-MerakiAPI -APIKey '{key string}' -OrgId 'XXXXXX' -SecureKey

This will create the file .meraki/config.json file in your user profile.

Create additional profiles. Profiles are used to identify the Organization you wish to operate on. Profiles and the parameter -OrgId are only required on functions that operate at the organization level. For appliances, devices, etc. you do not need to provide a profile or use the -OrgId parameter.

When creating an additional profile DO NOT include the -APIKey parameter. You only need the -OrgId and the -ProfileName parameters.

Set-MerakiAPI -OrgId '987456' -ProfileName "MyOtherOrg"

See the module reference for additional information on functions, syntax, and examples.