This PowerShell script aims to monitor the status of network devices in a Meraki network and switch between two different network paths (original and alternate) based on the devices' status.

Here's a step-by-step explanation of each part of the script:

A screenshot of a computer program

Description automatically generated with medium confidence

A picture containing text, screenshot, software, multimedia software

Description automatically generated

**Set your Meraki API key and network ID:**

**A picture containing text, font, screenshot, line

Description automatically generated**

The $apiKey and $networkId variables store your **Meraki API key** and **network ID**, which are required for API authentication and specifying the target network.

**Set the endpoint URL for retrieving network status information:**

A screen shot of a computer

Description automatically generated with low confidence

The $url variable is assigned the endpoint URL for the Meraki API, which retrieves network status information.

**Set the alternate path IP address:**

A picture containing text, font, screenshot

Description automatically generated

The $altPath variable stores the IP address of the alternate network path the script will switch to in case of an outage.

**Set the original path IP address:**

**A picture containing text, font, screenshot

Description automatically generated**

The $originalPath variable stores the IP address of the original network path.

**Set the headers for the API request:**

A picture containing text, screenshot, font, line

Description automatically generated

The $headers variable is a **hashtable** that stores the API key under the "X-Cisco-Meraki-API-Key" key, which is required for authenticating API requests.

**Loop until the script is stopped:**

**A screen shot of a computer program

Description automatically generated with low confidence**

The while ($true) loop is an infinite loop that will continue executing until the script is manually stopped.

**Make the API request:**

**A picture containing text, screenshot, font

Description automatically generated**

The Invoke-RestMethod cmdlet sends an HTTP request to the API endpoint URL specified in $url, passing the headers from $headers. The response is stored in the $response variable.

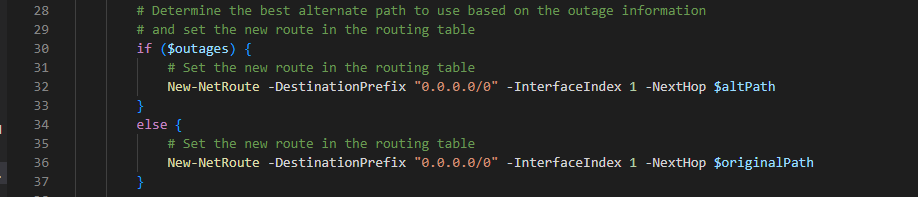
**Parse the response to extract outage information:**

**A picture containing text, screenshot, font, line

Description automatically generated**

The $outages variable holds the outage information. It filters the $response using the Where-Object cmdlet to include only the objects with a status that is not **"ok.”**

**Determine the best alternate path to use based on the outage information and set the new route in the routing table:**



If any outages (the $outages variable is not empty), the script sets the alternate path IP address as the next hop in the routing table using the New-NetRoute cmdlet. Otherwise, it sets the original path IP address as the next hop.

**Loop until the original path becomes available again:**

**A screen shot of a computer program

Description automatically generated with low confidence**

Another while ($true) loop is used to continuously monitor the original path's availability.

**Determine if the original path is available:**

****

The Test-Connection cmdlet sends a ping to the original path IP address. The result is stored in the $pingResult variable.

**If the original path is available, switch back to it and exit the loop:**

A screen shot of a computer

Description automatically generated with low confidence

If the original path is available (the $pingResult is true), the script sets the actual path IP address as the next hop in the routing table using the New-NetRoute cmdlet and breaks out of the inner loop.

**Wait for a short period before checking again:**

**A picture containing text, screenshot, font

Description automatically generated**

The Start-Sleep cmdlet causes the script to pause for 5 seconds before retesting the availability of the original path.

**Handle any errors that occur during the script execution:**

A picture containing text, screenshot, font

Description automatically generated

The catch block handles any errors that may occur during the script execution and writes the error message to the console using the **Write-Host** cmdlet.