Poco External Switch API Guide

Network Control and Monitoring of the Lumitec Poco Digital Lighting Controller

Version History 2
Introduction and Terminology 3
  Power Line Instruction (PLI) 3
  Poco Virtual Switches 3
  Poco External Switches 3
  Poco External Switch API (ExtSw-API) 3
Poco on an IP Network 4
  Discover the Poco's IP address via mDNS 5
  Connecting to the Poco via HTTP 5
Create an External-Switch via the Poco UI 5
Poco External Switch API 6
WebSocket Notifications 9
Appendix A—API Specification 10
Appendix B— Remote access Poco via the Internet.... 10
Appendix C – Other useful APIs 11
  Poco Device Information 11

Version History

Document Version	Date	Changes
3.3.0	2023-06-09	Added Semantic Versioning, Error-codes, Heartbeat, and WebSocket notification of Extsw state changes.
3.2	2023-03-31	Added new actions T2HSB, T2HS, T2B, T2BD, Patterns. Sync doc version to API spec version.
1.3	3/17/2022	Changed formatting, updated "Setup an External-Switch on the Poco UI" and created "Appendix E"
1.2	2/15/2022	Added Appendices
1.1	10/13/2021	Added WebSocket information
1.0	5/4/2021	Initial Version

Introduction and Terminology

The purpose of this document is to introduce the Poco External Switch API, explain the uses and benefits, and provide examples of how to use the API.

Poco Digital Lighting Control delivers powerful, low-cost on-board lighting control of Lumitec PLI-enabled lights. Lumitec is pleased to give third party integrators the ability to interface with Poco through an API. This document will explain how to integrate with the Poco Digital Lighting Controller, how to search and connect to a Poco, and examples of using the API.

Power Line Instruction (PLI)

PLI is Lumitec's proprietary 2-wire technology that will allow color change, dimming, addressing, and advanced patterns of PLI-enabled lights using just the power and ground wires. The PLI electrical specs and command set are not publicly available and not covered by this document. The Poco will take care of generating and sending the PLI signals to lights.

Poco Virtual Switches

Poco allows grouping of individual lights into **Virtual Switches** which can be controlled together. Users can interact with Poco **Virtual Switches** via an easy-to-use HTML5 user interface (**Poco UI**) on compatible marine Multi-Function-Displays (MFD), and Android/iOS mobile devices.

Poco External Switches

Virtual Switches are only directly available on the Poco UI, as most of the logic and state is implemented on the HTML5 client's machine. To make a Virtual Switch available without using the Poco UI, you must export it from the UI to be available to the Poco's internal firmware as an **External Switch** which can execute the "Actions" under the Automation tab.

External Switches are available to be:

- Set as the Poco Start-up Switch. It is automatically activated on Poco boot-up and does not depend on a connected UI.
- Controlled by the Poco discrete input (blue wire on Poco 3+).
- Controlled via physical switches wired to Poco via the **Pico S8** module on the Poco Accessory Bus (blue wire on Poco 3+).
- Controlled and monitored by a 3rd-party system via this API (i.e., a Home Automation System).

Poco External Switch API (ExtSw-API)

Poco provides an Application Programming Interface (API) for control and monitoring of Poco External Switches over an IP network called the Poco External Switch API (ExtSw-API).

Poco on an IP Network

The Poco has three IP network interfaces, each can have its own IP address and can function simultaneously and independently (WiFi AP, WiFi STA, and Ethernet. Poco also has a BLE interface, which is completely different and not discussed here).

Poco's WiFi AP, WiFi STA, and Ethernet interfaces are essentially identical at the IP and HTTP level, except that they will have different IP addresses. That means that once you have established a connection to the network, the rest of the communication is agnostic to the physical medium. A command (i.e., http://192.168.2.119/v3/extsw?id=1&act=2) will look and act the same on all three IP networks, except that the IP address will be different.

WiFi AP:

Poco's WiFi Access Point (AP) interface is special because it has a DHCP Server (dhcps) attached. Assuming you connected to the Poco's AP from your computer, the Poco's DHCP Server assigned your computer's WiFi interface an IP in the range of 192.168.4.2~254. Then you can reach the Poco via it's static IP of http://192.168.4.1/ (and http://poco.local/ if your client supports mDNS).

WiFi STA and Ethernet:

Poco's WiFi Station (STA) and Ethernet and interfaces don't have a DHCP Server (dhcps) enabled, so they may not be as easy to setup. STA and Ethernet are typically used on networks that already provide a DHCP Server (i.e., a home or business router). If Poco provided a duplicate DHCP Server on these networks, it could cause IP address conflicts. Instead, Poco acts as a DHCP Client (dhcpc) and requests an IP from the network.

If there is an existing DHCP Server on the network, Poco will request and receive an IP in the range configured for the network. If your client is also on the same network, then it should also have a unique IP in the same range, and thus IP communications should be possible.

If there is no DHCP Server on the network, then Poco will self-assign an IP in the range of *169.254.0.0* through *169.254.255.255*. Since this behavior is a standard implemented on most systems, then your client will likely also have self-assigned an IP in the same range, but unique, and thus IP communications should be possible.

Now, the tricky part is finding Poco's IP address, since it may be randomly picked from the available range. And Poco doesn't have any way to display it's IP address directly (i.e., an LCD display, etc.). There are many approaches to solve this problem, I'll outline a few:

- mDNS is implemented on Poco, and it advertises itself on any IP network. If your client also supports mDNS (a.k.a. Bonjour, Zeroconf, Avahi), then you can find the Poco by looking for http://poco.local.
- Log-in to the web interface of your router to view all devices on the network. Since your router is the DHCP server it keeps a list of all the devices that have requested an IP. You should see the Poco listed by its hostname (i.e., "poco-1234") along with its IP address.
- Statically assign IP address by DHCP Server (router): You should be able to configure your router to assign a fixed IP address to the Poco. Every router make/model is different, see your router's documentation for setup details.

- Scan the local network using a tool like arp-scan, nmap, etc. (Caution, this may trigger a security warning on your network)

Note: The ability to assign a Static IP address to a Poco interface is not implemented but may be added in a future update.

Discover the Poco's IP address via mDNS

Poco advertises itself via mDNS with hostname "poco.local".

Poco also advertises an mDNS service of type "_http._tcp". To be sure it is a Poco and not another device with a web interface, look for the txt record "poco=true". (This is how the Poco iOS/Android App finds Pocos on the local network.)

For example, on Linux: You can discover Poco on your local network with:

```
avahi-browse _http._tcp --resolve -t
```

From those results, filter by text record = "poco=true".

Note: If there are multiple Pocos on the same network, then additional Pocos may be advertised with **Hostnames** "poco-1.local", "poco-2.local", etc. You should not rely on the hostname to uniquely identify a particular Poco. For that purpose, you may use the mDNS **Instance Name** (i.e., "poco-12AB"). The instance name is unique to a particular Poco and persistent, unlike the hostname.

Connecting to the Poco via HTTP

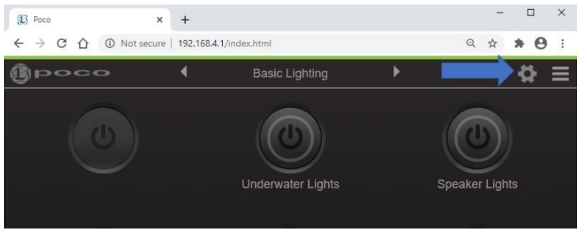
Poco supports HTTP and WebSocket on port 80. The main Poco UI is an HTML5/JS application which is accessed at http://poco.local/index.html. Users can control and configure the Poco using this UI from a web browser or via the Poco Android/iOS app.

For Machine-to-Machine control and automation, you should use the ExtSw-API to manipulate and query the state of Poco: http://poco.local/v3/extsw.

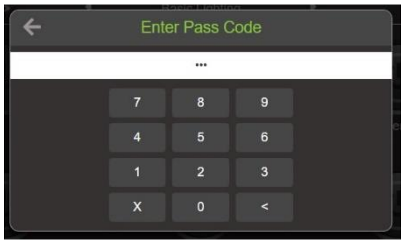
Note: poco.local can be replaced with the IP address of Poco (I.e. http://192.168.4.1/). It is recommended to use Poco's IP address instead of the mDNS name for robust M2M communication.

Create an External-Switch via the Poco UI

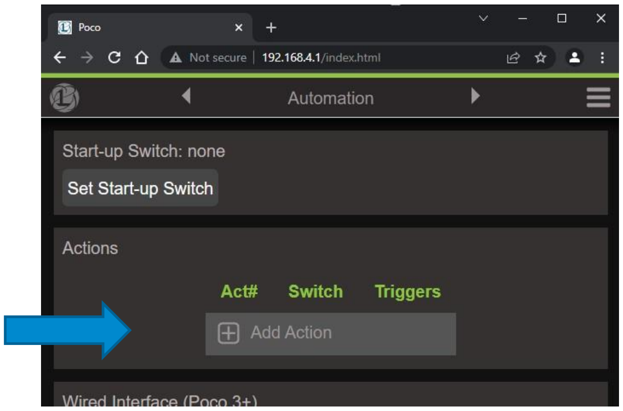
Step 1 Connect to the Poco UI using an Internet browser or the Poco Android/iOS App.



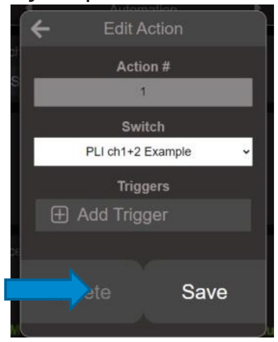
Step 2 Enter the Poco configuration menu by clicking the gear icon in the top right. You will be prompted to enter a pass code, and the default code is “0000”.



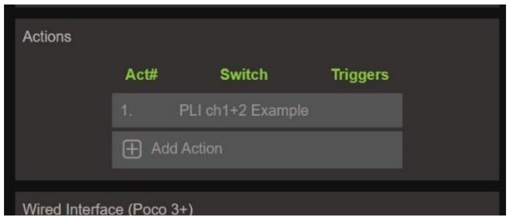
Step 3 Navigate to the “Automation” tab and select “+ Add Action”.



Step 4 Select the Virtual Switch to be activated by requests to this ExtSw-API ID#. Remember to hit save.



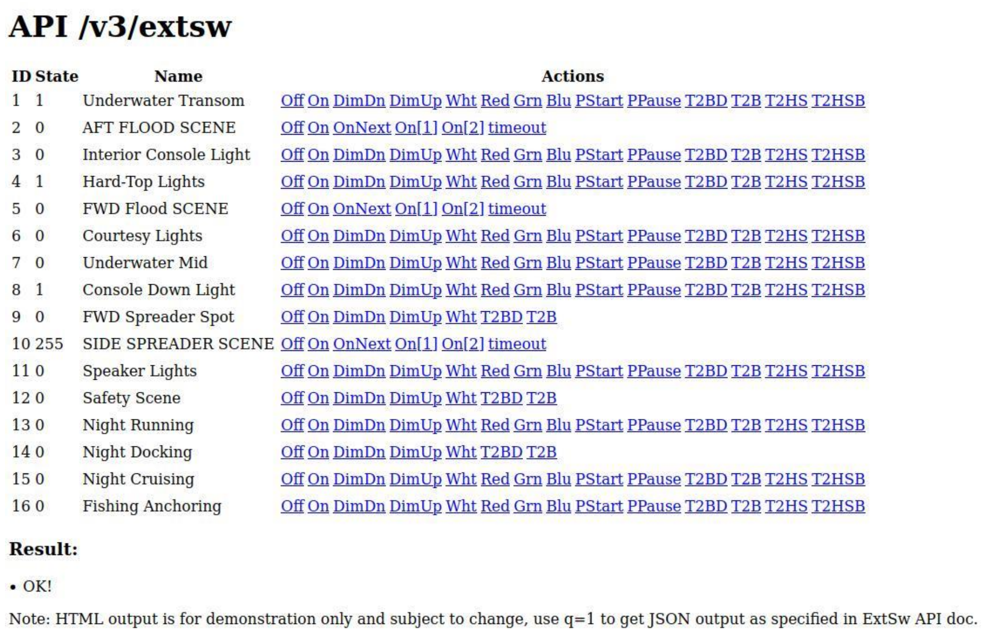
Step 5 See that the newly created External-Switch appears on the list of the Poco’s “Actions”.



Poco External Switch API

After configuring some external switches in the main UI, the External Switch API-UI can be tested at http://192.168.4.1/v3/extsw.

- The default view is rendered as HTML for easy evaluation. The HTML format is subject to change and should not be used in a program.
- A machine-readable JSON output is enabled with HTTP query “q=1”. This API document defines the JSON schema and will try to maintain backward compatibility in future versions.



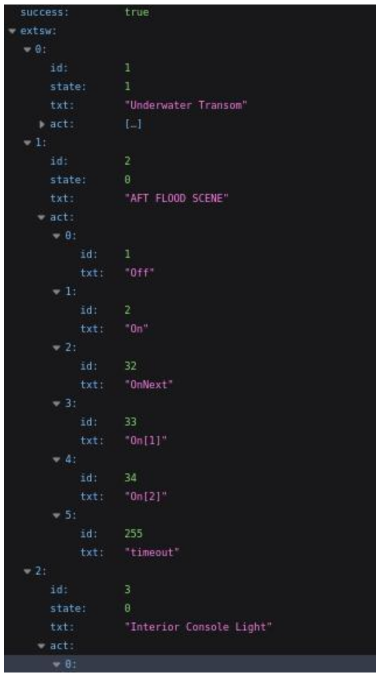
API /v3/extsw

ID	State	Name	Actions
1	1	Underwater Transom	Off On DimDn DimUp Wht Red Grn Blu PStart PPause T2BD T2B T2HS T2HSB
2	0	AFT FLOOD SCENE	Off On OnNext On[1] On[2] timeout
3	0	Interior Console Light	Off On DimDn DimUp Wht Red Grn Blu PStart PPause T2BD T2B T2HS T2HSB
4	1	Hard-Top Lights	Off On DimDn DimUp Wht Red Grn Blu PStart PPause T2BD T2B T2HS T2HSB
5	0	FWD Flood SCENE	Off On OnNext On[1] On[2] timeout
6	0	Courtesy Lights	Off On DimDn DimUp Wht Red Grn Blu PStart PPause T2BD T2B T2HS T2HSB
7	0	Underwater Mid	Off On DimDn DimUp Wht Red Grn Blu PStart PPause T2BD T2B T2HS T2HSB
8	1	Console Down Light	Off On DimDn DimUp Wht Red Grn Blu PStart PPause T2BD T2B T2HS T2HSB
9	0	FWD Spreader Spot	Off On DimDn DimUp Wht T2BD T2B
10	255	SIDE SPREADER SCENE	Off On OnNext On[1] On[2] timeout
11	0	Speaker Lights	Off On DimDn DimUp Wht Red Grn Blu PStart PPause T2BD T2B T2HS T2HSB
12	0	Safety Scene	Off On DimDn DimUp Wht T2BD T2B
13	0	Night Running	Off On DimDn DimUp Wht Red Grn Blu PStart PPause T2BD T2B T2HS T2HSB
14	0	Night Docking	Off On DimDn DimUp Wht T2BD T2B
15	0	Night Cruising	Off On DimDn DimUp Wht Red Grn Blu PStart PPause T2BD T2B T2HS T2HSB
16	0	Fishing Anchoring	Off On DimDn DimUp Wht Red Grn Blu PStart PPause T2BD T2B T2HS T2HSB

Result:

- OK!

Note: HTML output is for demonstration only and subject to change, use q=1 to get JSON output as specified in ExtSw API doc.



```
success:        true
▾ extsw:
  ▾ 0:
      id:       1
      state:    1
      txt:      "Underwater Transom"
    ▸ act:      […]
  ▾ 1:
      id:       2
      state:    0
      txt:      "AFT FLOOD SCENE"
    ▾ act:
      ▾ 0:
          id:   1
          txt:  "Off"
      ▾ 1:
          id:   2
          txt:  "On"
      ▾ 2:
          id:   32
          txt:  "OnNext"
      ▾ 3:
          id:   33
          txt:  "On[1]"
      ▾ 4:
          id:   34
          txt:  "On[2]"
      ▾ 5:
          id:   255
          txt:  "timeout"
  ▾ 2:
      id:       3
      state:    0
      txt:      "Interior Console Light"
    ▾ act:
      ▾ 0:
```

FIGURE 1- EXAMPLE OF EXTSW API HTML UI (LEFT) AND JSON OUTPUT (RIGHT)

The current state of each External Switch can be read as the value of the variable “state”. A value of 0 means the switch is “Off”, any other value means switch is not off. A non-zero value may indicate additional information depending on the type of switch.

An External Switch command is sent to Poco using a HTTP GET request. i.e.:

```
http://192.168.4.1/v3/extsw?q=1&id=1&act=2
```

Where 192.168.4.1 is the I.P. address of Poco, ‘id=1’ specifies the External Switch with id #1, and ‘act=2’ specifies the Action id of 2, which turns “On” the switch.

The “DimDn” and “DimUp” actions can adjust brightness up or down. The default increment is +/-10%. But you can specify a different increment with the “T2BD” action. In addition to the “T2BD” action you must specify the “delta” parameter. The full scale of delta is -128/+127. For example, to increase the brightness by 25%, take the positive section of the scale and divide by 4 (+127/4) = 32. To find a decreasing delta, take the negative end of the scale and divide it by the brightness factor. For example, 10% decrease in brightness would have the calculation -128/10 = -12.

Decrease Brightness by 10% (DimDn)	http://poco.local/v3/extsw?id=1&act=3

Increase Brightness by 10% (DimUp)	`http://poco.local/v3/extsw?id=1&act=4`
Increase Brightness by 25% (T2BD)	`http://poco.local/v3/extsw?id=1&act=5&delta=32`
Decrease Brightness by 10% (T2BD)	`http://poco.local/v3/extsw?id=1&act=5&delta=-12`

Table 2. BRIGHTNESS COMMAND EXAMPLES

Changes to the External Switch's hue, saturation, and absolute brightness can be achieved by using one of the "T2HSB", "T2HS", or "T2B" actions with optional parameters- hue, sat, and bright. The hue parameter is an integer that controls the color of the light. The primary colors are- 0: Red, 85: Green, and 170: Blue. The max hue is 255 which is close to red. Sat controls the saturation of the light by sending an integer on a 0-255 scale to allow anywhere from 0: white to 255: full color saturation. Bright is sent as an integer on a 0-255 scale to control the absolute brightness of the light, where 0: no brightness and 255: 100% brightness. If a parameter is not specified, the default value defined by the switch can be used.

"T2HSB" allows changes to the light's hue, saturation, and absolute brightness in one action. For example: if the "T2HSB" action was sent with hue=0&sat=255&bright=255, the light will respond by changing the hue to red and setting the saturation and brightness at max. For another example: if only hue and saturation were specified as hue=85&sat=255, then the light will change its hue to green at full saturation but use the default brightness of the switch. Since the action modifies all three: hue, saturation, and brightness, it will use the default values for any of those not specified.

"T2HS" accepts the hue and sat parameters. For example: if the T2HS action is sent with hue=170&sat=255, then the light will respond by changing its hue to blue at full saturation. If the previous state was On, then the brightness will not change. But if the switch state was off before the command, then it will also set the default brightness of the switch. For another example: if the T2HS action only specifies sat=255 and no hue, then the switch's default hue will be set.

"T2B" will allow change to just the absolute brightness of the light without modifying the hue or saturation. "T2B" only accepts the bright parameter. By sending bright=128, the light will change its brightness to 50% and if previously on the hue and saturation will stay at whatever the user selected, but if this is on turn-on then the light will have the default hue and saturation based on the switch specification.

Set Full Brightness, Color Red, Full Saturation (T2HSB)	`http://poco.local/v3/extsw?id=1&act=8&hue=0&sat=255&bright=255`
Set Hue to Green & Full Saturation (T2HS)	`http://poco.local/v3/extsw?id=1&act=9&hue=85&sat=255`
Set Brightness 50%, Leave Hue & Saturation as-is (T2B)	`http://poco.local/v3/extsw?id=1&act=10&bright=128`

Table 3. HSB COMMAND EXAMPLES

Some Lumitec lights come with built-in patterns, such as Spectrum/Color-Cycle and White/Blue Cross-Fade. These patterns can be controlled with the "Pattern Start" and "Pattern Pause/Resume" actions. If the switch state was previously on, the pattern will start in the current brightness, but if the state was off, then the pattern will start in the switch's default brightness. To override the default pattern, you can specify the pattern start index as parameter "pid".

Start Default pattern	`http://poco.local/v3/extsw?id=1&act=6`
Start Spectrum (Color-cycle) pattern	`http://poco.local/v3/extsw?id=1&act=6&pid=4`
Start White/Blue Cross-Fade pattern	`http://poco.local/v3/extsw?id=1&act=6&pid=5`
Pause or resume the current pattern	`http://poco.local/v3/extsw?id=1&act=7`

Table 4. PATTERN COMMAND EXAMPLES

A switch of the type "Scene Select" is a mutually exclusive group of child-switches. The actions include an "On[n]" for each child-switch, where n is an index from 1 of the child-switch within the Scene Select switch. For example, if a Scene Select has 3 child-switches, then the actions will be "On[1]", "On[2]", "On[3]". By specifying an action of "On[n]", that child-switch will be turned "On", and all other child-switches within that Scene Select are turned "Off". The "On" and "OnNext" actions will cycle between the child-switches and "Off". ("On" is just an alias for "OnNext" in this case.)

The "state" variable of a Scene Select switch can represent the index of the child-switch which was just activated with "On" or "OnNext", but it will "Timeout" after a few seconds to the value of 255. The state value of 255 indicates that the Scene Select switch is "on", but a subsequent action of "On" or "OnNext" will turn off the switch "Off" instead of advancing to the next child-switch.

See Appendix for the full API specification.

WebSocket Notifications

As an alternative to polling this API for changes to External Switch states, other systems can subscribe to notification of changes to switch states and configuration via WebSocket API (https://websockets.spec.whatwg.org/#network).

For status updates (push) you may connect to the WebSocket broadcast at:

ws://poco.local/websocket/ws.cgi

Poco will broadcast JSON messages on WebSocket when the state of one or more switches has changed. The "extsw" array contains an object for each changed switch state. Below is an example:

```
{"typ":3,"sid":0,"extsw":[{"id":1,"state":1},{"id":11,"state":1},{"id":16,"state":1}]}
```

Note: Poco actions are not atomic, and state changes may be broken up into multiple messages.

When the Poco configuration has changed (i.e., added/removed/changed switches), Poco will broadcast a message like the following:

```
{"typ":2,"sid":7991,"confc":"refresh"}
```

Poco will periodically broadcast an uptime status message on WebSocket. Other systems may use this to verify connectivity to Poco. It is sent every 10 seconds and gives the Poco server's uptime in milliseconds.

```
{"typ":0,"sid":0,"uptime":11091}
```

Note: For development or debugging, WebSocket traffic can be seen at: http://poco.local/altui/websocket/index.html.

Note: The Poco server has a limited number of TCP/IP sockets. These must be shared among all HTTP and WebSocket clients. Please close unused connections.

Note: Other options in conjunction with WebSocket that are in consideration but not implemented yet: Webhook, MQTT, Server-Sent-Events (SSE), IP multicast, etc.

Appendix A—API Specification

Below is the API spec in OpenAPI 3 YAML format.

Tip: use a OpenAPI compatible viewer to view and test this specification. I.e., https://editor.swagger.io/?url=https://update.dev.lumitec.io/releases/poco-fw-ota/poco-fw_3p6-rc2/extsw_API_HTTP_3.3.0.yaml

To-do: insert copy of spec here.
Here's a link to it:
https://update.dev.lumitec.io/releases/poco-fw-ota/poco-fw_3p6-rc2/extsw_API_HTTP_3.3.0.yaml

Appendix B— Remote access Poco via the Internet

Note*: Lumitec Poco FW does not connect to the internet or any remotely hosted service. This section offers some general ideas to achieve remote access using common internet technology. It is up to the installer to implement this and Lumitec cannot provide direct customer assistance.*

Several methods for remote access to your Poco:

1. VPN connection to your local network which contains Poco.
    - Instructions will vary with the type of VPN software and mobile OS.
2. Forward a port through your Internet router/firewall to poco port 80. Then you can access Poco via your Internet IP address (or DNS name), i.e., http://8.8.4.4:8880 or http://myserver.net:6780 .
    - It's not secure by default, it is up to the installer to secure it.
    - Instructions will vary with router/firewall model. I can provide an example for Asus...

3. Reverse-Proxy with SSL/TLS and authentication. Then you can access Poco as a subdomain or subdirectory of your website, i.e., https://poco.mysite.net or https://mysite.net/poco
    - Instructions will vary with HTTP server. I can provide Apache2 and Nginx examples...
4. Integrate Poco with a 3rd party home automation system (i.e., HomeAssistant). The 3rd-party HA system is responsible for remote access and security.
    - Instructions will vary with HA system type.
5. Use a cloud service (MQTT—part of potential future improvements).
    - We can host our own service called Poco-Cloud or Poco-365. It's essentially just a MQTT server that Poco and Poco App connect to communicate.
    - Poco should be able to connect to other MQTT servers for 3rd-party integration. See ESPHome.

Appendix C – Other useful APIs

These other API commands may be useful but aren't formally specified for 3rd party use. Please inform Lumitec if you plan to use any of these.

Poco Device Information

GET /v2/info

Returns a JSON object:

```
{
  "sys_id": "84:cc:a8:0d:5c:f4",
  "hw": 33,
  "hw_ver": "3.E",
  "manuf": "Lumitec LLC",
  "model": "poco",
  "name": "poco-F45C",
  "serial": "123456",
  "date": "11/18/2021",
  "fw_ver": "3.6-rc1 ",
  "confc": "6d0df310"
}
```

"sys_id" value is derived from the Poco's MAC address, so it is unique and can't be changed. You could use this to uniquely identify a Poco on the network.

"name" is the Poco's friendly name. This is typically unique but can be changed by the user. You could use this to present in a user-friendly list of discovered Pocos.

"confc" value is a hash of the Poco's configuration file contents. You could use this to determine if Poco's configuration has been changed.