JSON WebAPI relay control

Back to Main Power Relay Module (PRM3) Product Page.

> WebAPI documentation may be found at https://egauge.net/support/webapi. The `/ctrl` endpoint is used for controlling the PRM3.

> The eGauge Python Library contains helper functions for authentication and requests and is available via Bitbucket and PyPi. Demonstration code provided by eGauge Systems typically requires this library.

> To interact with the WebAPI control service, the authenticated user must have permission "Allowed to view all data, change settings, and control devices from anywhere"

Starting in firmware v4.4, the PRM3 may be controlled with EG4xxx series meters by using the eGauge JSON-based WebAPI /ctrl endpoint. The following is a Python script showing some possible miscellaneous interactions with an eGauge PRM3.

```
#!/usr/bin/env python3

# Example script interacting with an eGauge PRM3 power relay module through a
# meter's WebAPI. The PRM3 must be connected to the eGauge via USB.
#
# This script uses the eGauge Python library, available from bitbucket or pip:
# https://bitbucket.org/egauge/python/src/master/egauge/
# https://pypi.org/project/egauge-python/
# WebAPI documentation: https://egauge.net/support/webapi

# Requires firmware version 4.4 or greater

from egauge import webapi

URI = "https://device-url"
USR = "my_meter_username"
```

```
PWD = "my_meter_password"

dev = webapi.device.Device(URI, webapi.JWTAuth(USR, PWD))

USB_PORT = "USB1" # what USB port the relay is connected to



# get the connected relay(s) information
relays = dev.get("/ctrl/device")
"""e.g.,
{
   'result':
   [
      {
         'path': ['net.egauge.slowd', 'USB1'],
         'mfg': 'eGauge',
         'model': 'PRM3',
         'sn': '00000004',
         'prot': 'SCPI',
         'link': 'serial',
         'quality': '1',
         'interface': ['relay', 'scpi']
      }
   ]
}
"""

# get the serial number of the PRM3 connected to port USB_PORT
# we use the SN to identify which relay to send commands to
for relay in relays["result"]:
   if USB_PORT in relay["path"]:
      relay_sn = relay["sn"]



# get the control interfaces and available methods for the PRM3 relay
# e.g., open_mask, set_mask, get_mask
methods = dev.get("/ctrl/interface")["result"]



# get the relay mask
# https://egauge.net/support/m/prm3/mask
```

```
payload = {
    "attrs": {"sn": relay_sn},
    "method": "relay.get_mask",
    "args": []
}

# store the transaction ID of the request
# e.g., {'result': {'tid': 897412938}}
tid = dev.post("/ctrl/call", payload)["result"]["tid"]

# get the response. e.g., {'result': 3} means relays 0 and 1 are closed, and
# 2 is open https://egauge.net/support/m/prm3/mask
print(dev.get(f"/ctrl/call/{tid}")["result"])



# set the mask to 5 (close relay 0 and 2, open relay 1).
# Use relay.open_mask and relay.close_mask to only open OR close relays
payload = {
    "attrs": {"sn": relay_sn},
    "method": "relay.set_mask",
    "args": [5]
}

# make the request, store the transaction ID
tid = dev.post("/ctrl/call", payload)["result"]["tid"]

# check the transaction response. This should generally be {'result': False}
print(dev.get(f"/ctrl/call/{tid}")["result"])



# we can also send arbitrary SCPI commands, such as configurations.
# here we set the Modbus baud rate to 9600:
payload = {
    "attrs": {"sn": relay_sn},
    "method": "scpi.exec",
    "args": ["MODBus:BAUD 9600"]
}

tid = dev.post("/ctrl/call", payload)["result"]["tid"]

# should be "OK"
```

```
print(dev.get(f"/ctrl/call/{tid}")["result"])



# verify the baud
payload = {
   "attrs": {"sn": relay_sn},
   "method": "scpi.exec",
   "args": ["MODBus:BAUD?"]
}

tid = dev.post("/ctrl/call", payload)["result"]["tid"]

# {'result': '9600\r\n'}
print(dev.get(f"/ctrl/call/{tid}")["result"])
```
