QSUSB API V1.2.2 Aug 2016

The QSUSB server listens on port 2020.
It uses a http REST based protocol as defined below and most commands can be tested in the address bar of a standard web browser (Chrome or FireFox recommended).
Your application can be written in HTML/CSS/Javascript and placed in the www folder, which acts as a simple web server.
You can also write an app in any other language (c,vb.net etc) and just make http calls to QSUSB server to communicate with QwikSwitch devices.

Commands

SET LEVEL

Set the output of a QwikSwitch device.

*http://host:port/***@id=level** (level = 0 - 100)

eg. http://127.0.0.1:2020/**@123450=100** (Set level of device @123450 to 100%)
eg. http://127.0.0.1:2020/**@123450=0** (Set level of device @123450 to 0%)

The server will return a JSON formatted string:
On Success: {id":"@123450","cmd":"STATUS.ACK","data":"ON,RX1REL,V40","rssi":"71%"}
On No Reply: {"id":"@123450","cmd":"ERROR","data":"NO REPLY","rssi":""}

Note: cmd = Command (ACK = Acknowledge)
data = " Output State , Model , Firmware Version "
Output State = ON/OFF for Relay's or 0% - 100% for Dimmer's
"rssi" = Received Signal Strength Indicator between the device and the USB Modem

GET LEVEL

Get the output status of a QwikSwitch device

*http://host:port/***@id=?**
eg. http://127.0.0.1:2020/**@123450=?**

Returns:
The server will return a JSON formatted string:
On Success: {"id":"@123450","cmd":"STATUS.ACK","data":"ON,RX1REL,V40","rssi":"71%"}
On No Reply: {"id":"@123450","cmd":"ERROR","data":"NO REPLY","rssi":""}

LISTEN

Listen for the next packet received by the QSUSB server. This is a blocking call, and will wait for the next packet before closing the connection (also known as “long polling”)

*http://host:port/***&listen**
eg. http://127.0.0.1:2020/**&listen**

Returns:
The server will return a JSON formatted string:
On Success: {"id":"@123456","cmd":"STATUS.ACK","data":"ON,RX1REL,V40","rssi":"62%"}

http://en.wikipedia.org/wiki/Push_technology

OFF TIMER

Set a delay off on a QwickSwitch device.
If the device is ON it will turn OFF after x seconds.
If the device it OFF it will turn ON then turn OFF after x seconds.
The timer is in the QS device and does not require the PC to run.

*http://host:port/***@id+seconds (seconds = 0 - 65535)**
eg. http://127.0.0.1:2020/**@123450+60** (Turn OFF after 60 sec)

The server will return a JSON formatted string:
On Success: {"id":"@123450","cmd":"STATUS.ACK","data":"ON,RX1REL,V40","rssi":"71%"}
On No Reply: {"id":"@123450","cmd":"ERROR","data":"NO REPLY","rssi":""}

Device List

Store a list of device ID and Names in the QSUSB Server. The list is stored in a file called “devices” in the same folder as the qsusb.exe binary and is loaded on startup.

ADD/EDIT A DEVICE

To add or edit a device in the device list:

*http://host:port/***&device/id/type/name**

id = Device ID eg. @123450
type = Device type rel = relay | dim = Dimmer | tmp = Temperature Sensor
name = Any choosen name eg. Lounge Light (Max 30 Chars)

eg. http://127.0.0.1:2020/**&device/@123450/rel/Bedroom Light**
Returns: DEVICE ADD (Device Successfully Added)
or: DEVICE EDIT (Device already exists and was edited)

REMOVE A DEVICE

To remove a device from the device list:

*http://host:port/***&device/id-**

eg. http://127.0.0.1:2020/**&device/@123450-**
Returns: DEVICE DELETE (Device removed)

VIEW DEVICE LIST

To view all devices in the list:

*http://host:port/***&device?**

eg. http://127.0.0.1:2020/**&device?**

The server will return a JSON formatted string:
On Success eg. :

[{"id":"@021d60","name":"Light","type":"rel","val":"ON","time":"1339074575","rssi":"73%"},{"id":"@0392b0","name":"Office Dimmer","type":"dim","val":"0%","time":"1339074362","rssi":"82%"}]

note: “time” = Epoch of when last update was received
Or: [ ] (No Device List available)

QSUSB Scheduler

The schedule “engine” runs in the QSUSB Server. The schedules are stored in a text file called “cron” in the same folder as the qsusb.exe binary and is loaded on startup.

SET A SCHEDULE

To set a schedule in the QSUSB Server use the following command:

*http://host:port/***&cron/index/days/hh:mm/@id=level@id=level**

&cron = Scheduler command
index = 001 to 100
days = Days of Week [M|m T|t W|w T|t F|f S|s S|s]
Capital Letter = day enabled Small Letter = Day disabled
eg. “MTWTFSS” = everyday
eg. “Mtwtfss” = only Monday
eg. “mtwtfSS” = Saturday and Sunday enabled
“mtwtfss” is not allowed, at least 1 day of week must be enabled.
hh:mm = Trigger time (hour : minute 00:00 - 23:59)
@id=level Device ID = output value (0 - 100) ...Max 20 ID’s

eg. **http://127.0.0.1:2020/&cron/001/MTWTFss/01:30/@123450=0@11aa20=100**
At 1:30am on Mon,Tue,Wed,Thu,Fri turn device @123450 OFF and device @11aa20 to 100%

Returns: CRON SAVED (Schedule 001 saved)
or ERROR (Invalid or incorrect data entered)

DELETE A SCHEDULE

To remove a schedule in the QSUSB Server use the following command:

*http://host:port/***&cron/index-**

eg. **http://127.0.0.1:2020/&cron/001-**
Delete schedule 001

Returns: CRON SAVED (Schedule 001 Removed)
or ERROR (Invalid or incorrect data entered)

VIEW ALL SCHEDULES

To view all schedules in the QSUSB Server use the following command:

*http://host:port/***&cron?**

eg. **http://127.0.0.1:2020/&cron?**
View all Schedules

The server will return a JSON formatted string:
Returns:
On Success example:
[{"key":"001","run":1,"days":"SmTWtfs","time":"00:00","cmd":["@0392b0=0"]},{"key":"002","run":1,"days":"SMTWTFS","time":"23:00","cmd":["@074330=0"]}]

Or: [ ] (No Schedules Saved)

Data Storage

The QSUSB server allows your app to POST and GET data to the “mem” folder in simple text files.

Save Data (POST Only)

*http://host:port/***&save/filename**

filename = file name to save to (Max 16 chars alphanumeric only)

*NB The header ‘Content-Length’ **MUST** be set to the length of the data being posted,
otherwise the socket will hang up on you, and no data will be saved.

Read Data (GET)

*http://host:port/***&get/filename**

Other Commands

Get QSUSB Version

*http://host:port/***&version?**

Returns: QSUSB V1.70 (Server Version)
QwikSwitch USB V1.0 (Modem Firmware Version, if USB is connected)

USB Modem Connected

*http://host:port/***&usb?**

Returns: USB:OK (USB Modem is connected)
or USB:NONE (No USB Modem Connected)

GEYSER UNIT (GU)

Turn OFF heating: SET LEVEL 0 http://host:port/@xxxxxx=0
PING Failsafe: SET LEVEL 2 http://host:port/@xxxxxx=2
if GU does not receive a “ping” within 60 mins it will heat water to 50 °C. This is to rather have hot water if controller fails then cold.

Set Temperature 10 - 70 °C: SET LEVEL 10 ….70
GU Will keep water at this temperature.
e.g SET LEVEL 45 will heat water to 45 °C
http://host:port/@xxxxxx=45

Boost Now: SET LEVEL (128 + temp) temp = 10 to 70 °C
GU will heat to this value then return to last temperature
e.g SET LEVEL 178 = boost to 50 °C
To stop boost and return to last temperature: SET LEVEL 0

Get GU Status: http://host:port/@xxxxxx=?
return data e.g:
{"id":"@09c820","cmd":"STATUS.ACK","data":"370d1f1e071d","rssi":"45%"}

data: eg. 370d1f1e071d (hex)
[HW][SW][(boost),current temp][(heating?),setpoint][0000]

HW - Hardware version [byte]
SW - Software version [byte]
boost- whether geyser is currently boosting (MSB bit)
current temp - current geyser temperature (7 bits)
heating - whether geyser is currently heating (MSB bit)
setpoint - temperature to which the geyser is heating (7 bits)
[0000] unused

GEYSER UNIT (GU) [Continued]

Turn OFF ping requirement: http://host:port/@xxxxxx/set/0800

By default, the GU needs to receive a SETLEVEL command every 30 minutes,
otherwise it will return to it's idle temperature.

Turn ON ping requirement: http://host:port/@xxxxxx/set/0828

This will return the GU to it's default state of needing to receive a ping every 30 minutes.

Special Notes

RX1-30A

This device returns a special data packet. Here is how to decode.

Decoding Data packet:

Relay **ON:** 3002**8**000000000

Relay **OFF**: 3002**0**000000000

Character in bold should be switched, where 8 is **ON**, and 0 is **OFF**.