mySQM+ PROTOCOL 032

May 2022

APPLICABLE TO FIRMWARE 128-35+

INTERFACE COMMANDS - TCP/IP, PORT 2121

- All commands are a string of char characters
- All send commands begin with a colon character :
- All commands have a minimum of three digits (0-9).
- All commands end with a hash character #
- Commands are zero padded from the left to three places, for example, command 2 is written as :002#

RETURN VALUES

- All return values are terminated with a #
- All return values are string characters, numbers are converted to strings before sending

Send	**Meaning**
:000#	Get System up-time
:001#	Reboot controller
:002#	Get MAC Address
:003#	Get SQM
:004#	Get LUX
:005#	Get NELM
:006#	Get make hay
:007#	Get services: returns Controllermode, Webserver state, TCPIP Server state, OTA state, MQTT client state, mqtt connect state, NTP client state, Management server state
FIRMWARE	
:008#	Get firmware version number
:009#	Get firmware filename
:010#	Get firmware hash value
DISTANCE MODE	
:011#	Get metric/imperial mode 0 = metric 1 = Imperial
:0120#	Set metric
:0121#	Set imperial

TEMPERATURE MODE

:013#	Get temperature mode (0=Celsius 1=Fahrenheit)
:0140#	Set temperature mode Celsius
:0141#	Set temperature mode Fahrenheit

TLS2591

:017#	Get tlscorrectionfactor
:018#	Get TSL2591 Gain
:019#	Get TSL2591 Integration time
:020#	Get TSL2591 sample size
:021#	Get TSL2591 averaging (0=Disabled, 1= Enabled)
:022n.n#	Set TLS sensor correction factor
:023n#	Set TSL2591 Gain
:024n#	Set TSL2591 Integration time
:025n#	Set TSL2591 sample size
:0260#	Set TSL2591 averaging OFF
:0261#	Set TSL2591 averaging ON

:027# UNUSED

BME280

:028#	Get Relative Humidity
:029#	Get Pressure
:030#	Get Ambient temperature
:031#	Get Dewpoint
:032#	Get Altitude (in meters)
:033#	Get Pressure adjusted to sea-level
:034x.x#	Set Altitude (float in meters)

MLX90614

:035#	Get IR Sensor object temperature
:036#	Get IR Sensor ambient temperature
:037#	Get Sky-State, 0=clear 1=partly cloudy 2=cloudy
:038#	Get % Cloud Cover
:039#	Get corrected sky value
:040#	Get cloud sensor k1 - k7 values k1,k2,..k7#
:041#	Get Temp Clear
:042#	Get Temp Overcast
:043#	Get Cloud-Flag-Percent
:044x,x#	Set cloud model k1..k7, separated by ,
:045n#	Set Temp Clear (-8)
:046n#	Set Temp Overcast (0)
:047n#	Set Cloud-Flag-Percent (30)

RAIN

:048#	Get rain total in mm over previous hour
:049#	Get Raining (Boolean)
:050#	Get RVout - raining analogue value (int)
:051#	Get cumulative rainfall for this current hour
:052#	Get cumulative rainfall for this current day
:053#	Do Sync Rain Bucket Gauge min/hour to RTC

WIND

:054#	Get wind speed (in meters per second)
:055#	Get wind direction
:056#	Get Wind Chill Factor
:057#	Get windspeed sensor name and wind direction sensor name
:058#	Get average windspeed last 30s
:059#	Get maximum wind speed gust last 30s
:015#	Get wind direction offset
:016n#	Set wind direction offset (unsigned int 0-359)

GPS

:060#	Get local date
:061#	Get local time
:062#	Get latitude
:063#	Get longitude
:064#	Get altitude (height in meters)
:065#	Get number of satellites
:066#	Get GPS fix
:067#	Get if GPS using static co-ordinates [0=No 1=Yes]
:068#	Get if GPS using truncated co-ordinates [0=No 1=Yes]
:069#	Get GPS Fix 'es, gga_quality_indicator, gll_fix_status, gll_position_mode_indicator, rmc_mode_indicator, gsa gsa_smode, gsa_navmode
:070#	Get GPS Static Latitude setting
:071#	Get GPS Static Longitude setting
:072#	Get GPS Format (0=DD, 1=DMS)
:0730#	Set GPS Format DD
:0731#	Set GPS Format DMS
:0740#	Set GPS use static co-ordinates OFF
:0741#	Set GPS to use static co-ordinates ON
:0750#	Set GPS truncated state OFF
:0751#	Set GPS truncated state ON
:076ss#	Set GPS Static Latitude setting (string)
:077ss#	Set GPS Static Longitude setting (string)

DISPLAY

:078#	Get display enabled state
:079#	Get page display option [String]
:080#	Get page display time [unsigned int]
:0810#	Set Display enabled state OFF
:0811#	Set Display enabled state ON
:082ss#	Set page display option [String]
:083nnnn#	Set page display time [unsigned int 2000-10000] in milliseconds

RTC

:084#	Get RTC date/time string dd/mm/yr,hh:mm:ss]
:085#	Get SYNC RTC to NTP, if NTP should sync date/time to RTC
:086ss#	Set RTC date/time mm,dd,yyyy,hr,mn,ss
:0870#	Set SYNC RTC to NTP state OFF
:0871#	Set SYNC RTC to NTP state ON

MANAGEMENT SERVER

:088#	Get Management Server state (0=Stopped=0, 1=Running)
:089#	Get Management Server Port number
:090nn#	Set Management Server Port number
:0910#	Set Management Server state OFF
:0911#	Set Management Server state ON

MQTT

:092#	Get MQTT Subscribe topic
:093#	Get MQTT Server state (Start=1, Stop=0)
:094#	Get MQTT Broker IP address
:095#	Get MQTT Publish Topic
:096#	Get MQTT Publish Interval
:097#	Get MQTT Subscribe Enable state
:098ss#	Set MQTT Subscribe topic
:099ss#	Set MQTT Broker IP address
:100ss#	Set MQTT Publish Topic
:101n#	Set MQTT Publish Interval
:1020#	Set MQTT Server state OFF
:1021#	Set MQTT Server state ON
:1030#	Set MQTT Subscribe state OFF
:1031#	Set MQTT Subscribe state ON

NTP

:104#	Get NTP Time one string
:105#	Get NTP Client state (0=Stopped=0, 1=Running)
:106#	Get NTP UTC date/time string [dd/mm/yr,hh:mm:ss]
:107#	Get NTP LOC [UTC adj by TZ] date/time string [dd/mm/yr,hh:mm:ss]
:108#	Get NTP Servername
:109#	Get NTP Update interval (in minutes)
:110ss#	Set NTP Servername
:111n#	Set NTP Update interval (in minutes,)
:112s#	Set NTP Time zone (this will also restart the NTP Client
:113#	Restart NTP Client (Reserved for future use)
:1140#	Start NTP Client (Reserved for future use)
:1141#	Stop NTP Client (Reserved for future use)

WEBSERVER

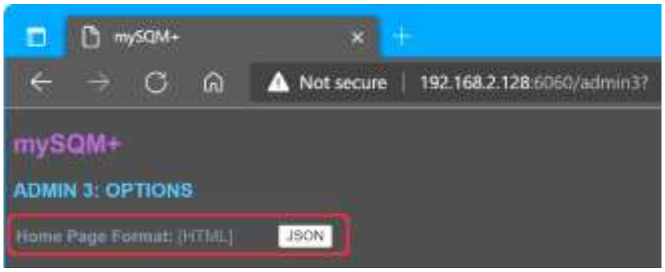
:115#	Get webserver port
:116#	Get webserver state (0=Stopped, 1=Running)
:117#	Set webserver port
:1180#	Set Webserver state OFF
:1181#	Set Webserver state ON

WEB SERVER

There is significant JSON support built into the web server functionality.

HOME PAGE FORMAT: HTML OR JSON

Management Server **/admin3** page provides the option of sending the Home Page of the Web Server in either HTML format or in JSON format.



mySQM+

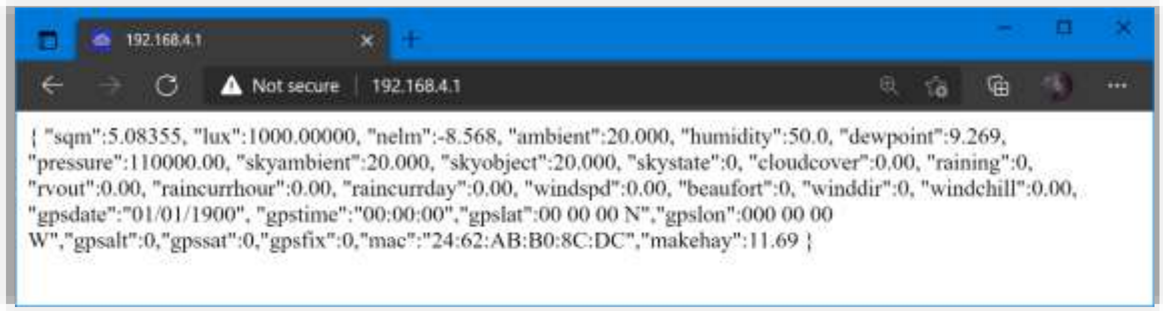
ADMIN 3: OPTIONS

Home Page Format: [HTML] JSON

In the screenshot, the current format setting for the home page is HTML (the current format state is shown between [].

The button shows what the home page format state will change to when clicked by the user.

Clicking on the JSON button will change the home page format to JSON, and any request for the home page will be sent a JSON string.



```
{ "sqm":5.08355, "lux":1000.00000, "nelm":-8.568, "ambient":20.000, "humidity":50.0, "dewpoint":9.269,
"pressure":110000.00, "skyambient":20.000, "skyobject":20.000, "skystate":0, "cloudcover":0.00, "raining":0,
"rvout":0.00, "raincurrhour":0.00, "raincurrday":0.00, "windspd":0.00, "beaufort":0, "winddir":0, "windchill":0.00,
"gpsdate":"01/01/1900", "gpstime":"00:00:00","gpslat":00 00 00 N","gpslon":000 00 00
W","gpsalt":0,"gpssat":0,"gpsfix":0,"mac":"24:62:AB:B0:8C:DC","makehay":11.69 }
```

This can be useful if you want to write your own applications to communicate with the controller. The client application is responsible for connecting to the controller and sending a GET root page **/** request to the Web Server.

All parameters are lowercase and values are returned in CELSIUS and METRIC. The client application is responsible for converting any values to FAHRENHEIHT or IMPERIAL units of measurement.

This call obeys the rules for truncate gps and static gps. The gps co-ordinates are derived from the static gps co-ordinates, use static gps is enabled, and truncate gps is enabled on the Management Server page ***/admin2***

Using the Home Page customizations, you can decide what values/settings appear on the Webserver's Home Page, for both HTML and JSON formats.

USING THE WEB SERVER WITH CLIENT APPLICATIONS

Client applications (such as Curl) can establish a connection to the Web Server on port 80. Using this port, the application can request the groups of data corresponding to various functions of the controller. A request returns a JSON formatted string. All requests and responses are lowercase.

WEBSERVER GROUPS AND JSON RESPONSES

The home page of the Web Server lists values under the main headings of Temperature, Dark Sky, Weather and GPS. These groups can be retrieved using the following requests

/d1

An application sending **/d1** receives a JSON string response of the "*DARKSKY*" variables.

```
{ "sqm":25.08356,"nelm":7.536,"lux":0.00,"skystate":1,"cloudcover":100.00 }
```

/g1

An application sending **/g1** t receives a JSON string response of the "*GPS*" variables. This call obeys the rules for truncate gps and static gps. To convert UTC to LOC, firmware uses the *TimeZone* and *TZ* string values from the */admin6* Management server page (NTP settings).

```
{ "gpsdate":"01/01/1900","gpstime":"00:00:00","gpslat":"00 00.000000 N","gpslon":"00 00.000000
E","gpsalt":0, "gpssat":0,"gpsfix":0 }
```

/rd

An application sending **/rd** receives a JSON string response of ALL dynamic values except GPS.

```
{
 "sqm":12.21815,"nelm":-1.461,"ambient":20.000,"humidity":50.0,"dewpoint":9.269,
"pressure":1100.00,"skyambient":20.000,"skyobject":20.000,"lux":1.40011,"skystate":0,
"cloudcover":0.00,"raining":0,"rvout":0.00,"windspd":0.00,"beaufort":0,"winddir":0
}
```

/t1

An application sending **/t1** receives a JSON string response of the "*TEMPERATURE*" variables.

```
{ "ambient":20.000,"dewpoint":9.269,"skyambient":23.850,"skyobject":23.690 }
```

/w1

An application sending **/w1** receives a JSON string response of the “*WEATHER*” variables.

{ "humidity":50.0,"pressure":110000.00,"raining":0,"rvout":0.00,"raincurrhour":0.00, "raincurrday":0.00,"windspd":0.00,"beaufort":0,"winddir":0 }

WEBSERVER REQUESTS AND JSON RESPONSES

The following requests return a JSON string of the requested values.

/about

An application sending **/**about receives a JSON string response of the *controller* details.

{ "product":"mySQM+","Author":"R. Brown, 2020-2021.","ver":"120" }

/date

An application sending **/date** receives a JSON string response for the *local date and time*. Which date and time is sent to the client is subject to the LOCALDATETIMESOURCE on the /admin web page.

{ "local":"02/12/2021,21:28:35","source":"rtc" }

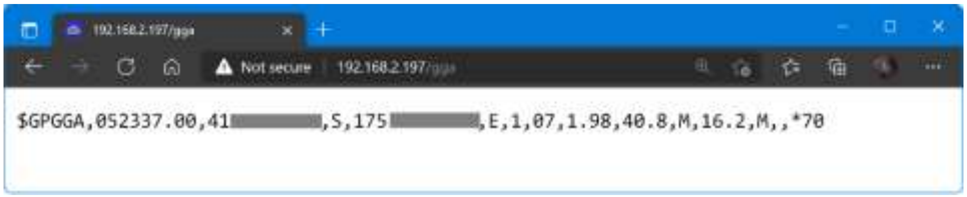
/gps

An application sending **/gps** receives a JSON string response for the *gps values and settings*.

{ "gpsdate":"01/01/1900","gpstime":"00:00:00","gpslat":"-77 00 00 S","gpslon":"166 00 00 E","gpsalt":0.0,"gpssat":0,"gpsfix":0 }

/ gpsgga

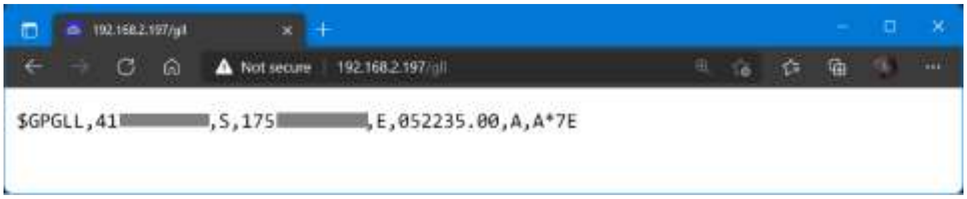
An application sending **/gpsgga** receives a string response for the *last received NMEA message*.



```
$GPGGA,052337.00,41[redacted],S,175[redacted],E,1,07,1.98,40.8,M,16.2,M,,*70
```

/gpsgll

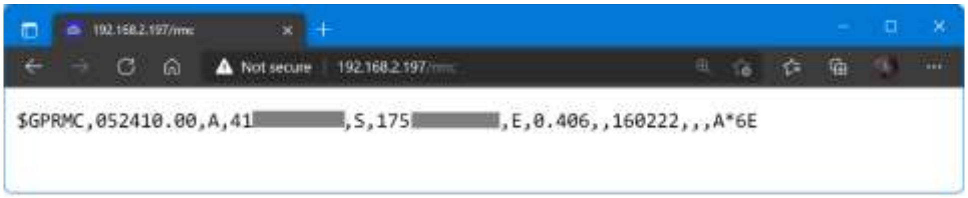
An application sending **/gpsgll** receives a string response for the *last received NMEA message*.



```
$GPGLL,41[redacted],S,175[redacted],E,052235.00,A,A*7E
```

/gpsrmc

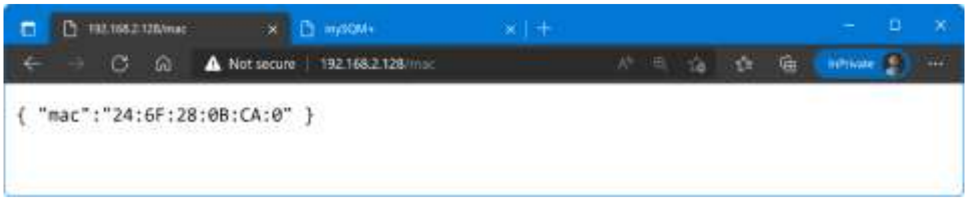
An application sending **/gpsrmc** receives a string response for the *last received NMEA message*.



```
$GPRMC,052410.00,A,41[redacted],S,175[redacted],E,0.406,,160222,,,A*6E
```

/mac

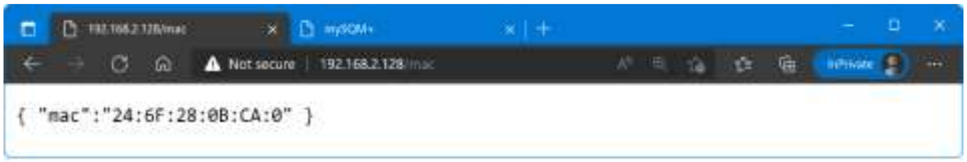
An application sending **/mac** receives a JSON string response of the *MAC address* value.



```
{ "mac":"24:6F:28:0B:CA:0" }
```

/makehay

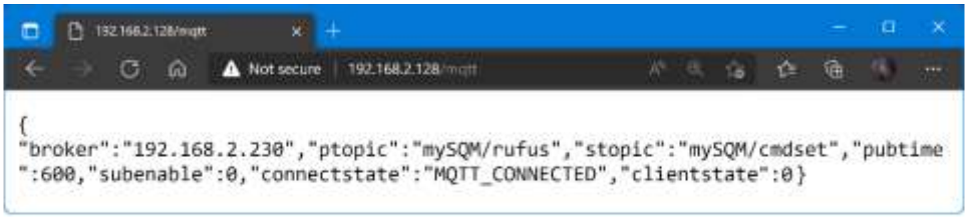
An application sending **/makehay** receives a JSON string response of the *vpd value* (make hay < 10 is good) value.



```
{ "mac":"24:6F:28:0B:CA:0" }
```

/mqtt

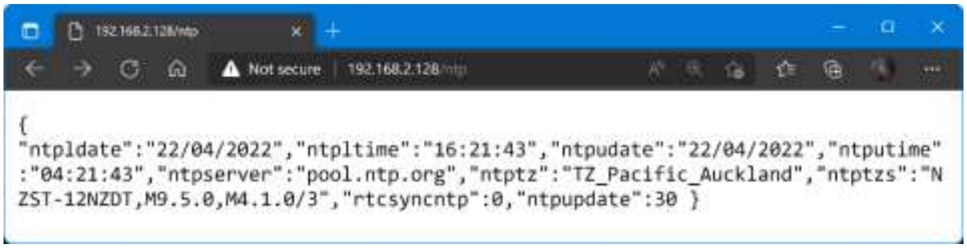
An application sending **/mqtt** receives a JSON string response of the *MQTT broker IP and connection details* [if enabled].



```
{
"broker":"192.168.2.230","ptopic":"mySQM/rufus","stopic":"mySQM/cmdset","pubtime
":600,"subenable":0,"connectstate":"MQTT_CONNECTED","clientstate":0}
```

/ntp

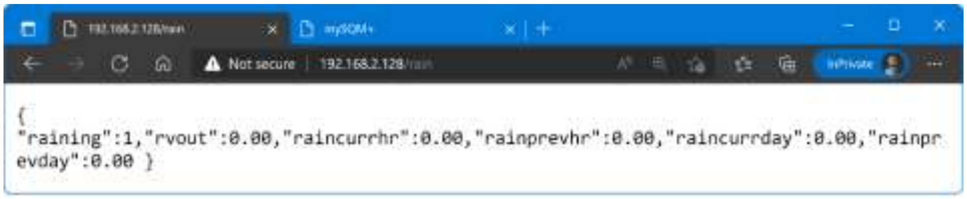
An application sending **/ntp** receives a JSON string response of the *NTP local date and time* dynamic values [if enabled].



```
{
"ntpldate":"22/04/2022","ntpltime":"16:21:43","ntpudate":"22/04/2022","ntputime"
:"04:21:43","ntpserver":"pool.ntp.org","ntptz":"TZ_Pacific_Auckland","ntptzs":"N
ZST-12NZDT,M9.5.0,M4.1.0/3","rtcsyncntp":0,"ntpupdate":30 }
```

/rain

An application sending **/rain** receives a JSON string response of the *rain information* [if enabled].



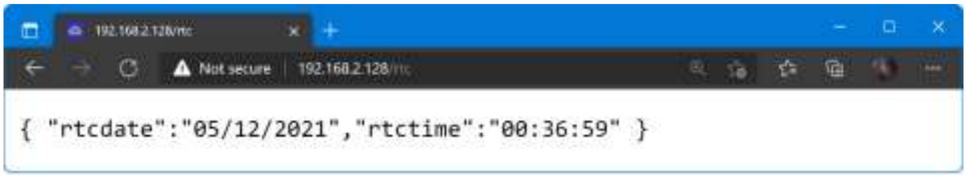
```
{
"raining":1,"rvout":0.00,"raincurrhr":0.00,"rainprevhr":0.00,"raincurrday":0.00,"rainpr
evday":0.00 }
```

/rbgsync

An application sending **/rbgsync** receives a JSON string response (Ok or nok) of the controllers attempt to synchronise the *rain bucket gaug*e to the *Real Time Clock*.

/rtc

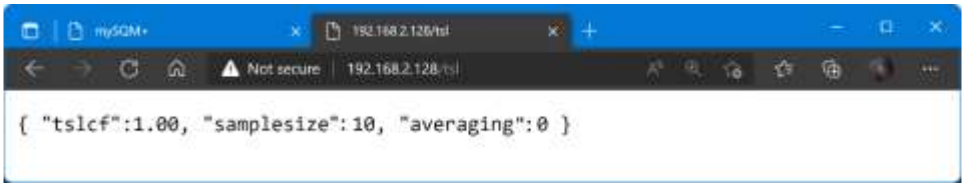
An application sending **/rtc** receives a JSON string response of the *Real Time Clock* dynamic values [if enabled].



```
{ "rtcdate":"05/12/2021","rtctime":"00:36:59" }
```

/tsl

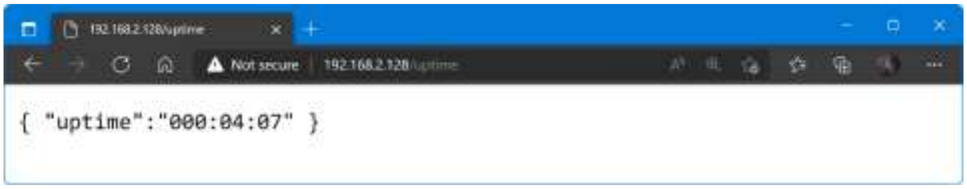
An application sending **/tsl** receives a JSON string response of the *TLS2591 sensor* settings



```
{ "tslcf":1.00, "samplesize":10, "averaging":0 }
```

/uptime

An application sending **/uptime** receives a JSON string response of the *system uptime.* The format is *days:hours:minutes*



```
{ "uptime":"000:04:07" }
```

CUSTOM JSON RESPONSES (from file customjson.h)

/cj1

{ "windspeed":0.00,"winddirection":0 }

/cj2

{ "sqm":10.38279,"nelm":-3.280,"lux":2.29704 }

WEB SERVER XHTML SUPPORT

The Web Server supports direct XJTL requests, where the value of the requested item is returned. The home page of the webserver as an HTML page has embedded scripts in the page that send XHTML requests to the server, and the responses are used to dynamically update the web page contents.

This means that you do not need to use REFRESH to get an updated page with new values, because the home page does this dynamically without having to request the home page again.

XHTML Request	Value returned
/ba	*ambient*
/bh	*humidity*
/bd	*dewpoint*
/bp	*pressure*
/ma	*skyambient*
/mo	*skyobject*
/sq	*sqm*
/ne	*nelm*
/lu	*lux*
/ss	*skystate*
/cc	*cloudcover*
/ra	*raining*
/rv	*rvout*
/rh	*raincuurhr*
/ry	*raincurrday*
/ws	*windspd*
/bs	*beaufort*
/wd	*winddir*
/wc	*windchill*
/ut	*system_up_time*
/mh	*makehay*

As an example of a **sq** request, the web server responds with the value of sqm at the time of the request

10.43037

MANAGEMENT SERVER

The Management Server [port 6060] provides the following JSON queries.

/reboot

An application sending **/reboot** to the controller will cause the controller to reboot. There is no response to this request. After 6s, the page will refresh and directed to the / page of the Management server.

/reboottcp

An application sending **/reboottcp** to the controller will cause the controller to stop and then restart the TCP/IP-server service [used by TCP/IP applications in Windows and LINUX on port 2121].

/rebootws

An application sending **/rebootws** to the controller will cause the controller to stop and then restart the web-server service.

/rbgsync

An application sending **/rbgsync** to the controller will cause the controller to update the **minute** and **hour** values used by the Rain Bucket Gauge to that of the Real Time Clock. There is no response to this request.