GreenEye Monitor

GEM Packet Format

Document Code: ***GEM-PKT***

Ver 2.1

2013-07-16

TABLE OF CONTENTS

GreenEye Monitor

Terms of Use and Copyright Notice

This document provides intellectual information which is the property of **Brultech Research Inc.** This document is subject to the terms and conditions listed below. This document has been provided to you by **Brultech Research Inc.** on condition of your acceptance of the following terms and conditions:

Personal and Non-Commercial Use Limitation

This document is for your personal and non-commercial use only. You may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell any information obtained from this document without the expressed written consent of **Brultech Research Inc**.

Brultech Research Inc. reserves the right to terminate your access to this information at any time without notice for any reason whatsoever.

Liability Limitations

The information included in this document may include inaccuracies or typographical errors.

Agreement of the Terms

Proceeding to the next page indicates that you agree to the terms set out in the above paragraphs.

You must agree to the terms and conditions described on this page before reading the following pages!

INTRODUCTION

This section describes the method used to acquire data from the GEM. The GEM requires a “data host” to collect, store and display the acquired measurements.

PACKETS

Data from the GEM is assembled into a group of bytes called a packet. Various packet format options are available depending on the means of communication and host requirements.

The two general type of byte types used by the GEM are “Binary” and “ASCII” bytes.

Binary

Binary data is a very basic data type and allows a packet to be compact. This provides the most efficient data transfer requiring minimal overhead included in the packet.

The downfall of binary packets is that it cannot readily be transferred over networks or the internet. This format is typically limited to communication via serial port, ZigBee module and sometime Wifi or Ethernet to serial modules.

Parsing the binary data is also a bit more complicated as opposed to ASCII data.

ASCII

ASCII data uses standard characters and numbers exclusively. This means that the data is readily viewable using text editors. This type of data is used exclusively when sending data to a web server via HTTP. The GEM can send ASCII data using HTTP “GET”, “PUT” or “POST” methods. It can also send a packet in the form of a standard ASCII string.

DATA PUSH OR PULL

Push

Data from the GEM can be “Pushed” at a pre-set time interval. This means that after “x” number of seconds, the GEM sends an updated packet and repeats this every interval. This is the most common method used by the GEM. With this method, the GEM typically operates as a client and the data host as a server.

Pull

The GEM can be configured to stop sending packets until a packet is requested by the host. Using this method, the data communication line is quiet. When the host requests an update, a command is sent to the GEM and it then sends a data packet. Optionally, the GEM may be configured to send a “keep alive” character to maintain the TCP/IP connection.

PACKET FORMATS

GEM PACKETS

The GEM current has the following packet formats. Each format is explained in this document.

List Format (0)

This format is restricted for viewing real-time data values. It should not be used by a host for data collection.

Multi ECM-1240 Simulation (1)

This binary format was created early in the development of the GEM to allow the ECM-1240's Engine Software to accommodate data from the GEM. This is accomplished by treating the first seven GEM channels (CH1 to CH7) as the seven channels of ECM-1240 device-1 and the next seven channels (CH8 to CH14) as ECM-1240 device-2 and so on.

When using this format, the number of ECM-1240 devices to be simulated may be set to a value from 1-5. This is configured under the "Packet Send" menu in the "Packet Format Option 1" section.

ASCII With Wh (2)

Regular ASCII data is used in this format in the form of:

Key1=Value&Key2=Value&………

This format may be the simplest to implement for those wishing to develop a custom application.

HTTP GET (3)

This format is used to post data using HTTP which is typical when sending data over a network or the internet to a web server.

BIN48-NET-Time (4)

Sends binary data for 48 channels (32 channels plus future 16 channel expansion), "watt-second" data for NET metering and a time stamp.

This is the format used by the DashBox when a direct serial connection is used or when communicating via XBee module.

BIN48-NET (5)

Sends binary data for 48 channels (32 channels plus future 16 channel expansion), "watt-second" data for NET metering. This packet resembles "Bin48-NET-Time" but excludes the time stamp.

.

SEG Format (6)

This this the old method used to post data to "Smart Energy Groups" data host (smartenergygroups.com). This has been replaced by an improved version "New SEG Format (14)" which is preferred for new installations. This format should only be used for early GEM installations which used this format.

BIN48-ABS (7)

This binary packet send 48 channel data (32 channels plus future 16 channel expansion). This packet does not include polarized watt-seconds required for net metering.

BIN32-NET (8)

Sends binary data for 32 channels and includes polarized "watt-second" data for NET metering.

BIN32-ABS (9)

Sends binary data for 32 channels but **excludes** polarized "watt-second" data.

Not Implemented (10)

Reserved for future format implementation.

Universal Device ISY (11)

Sends data to Universal Device's automation controller via XBee module.

Not Implemented (12)

Reserved for future format implementation.

COSM (13)

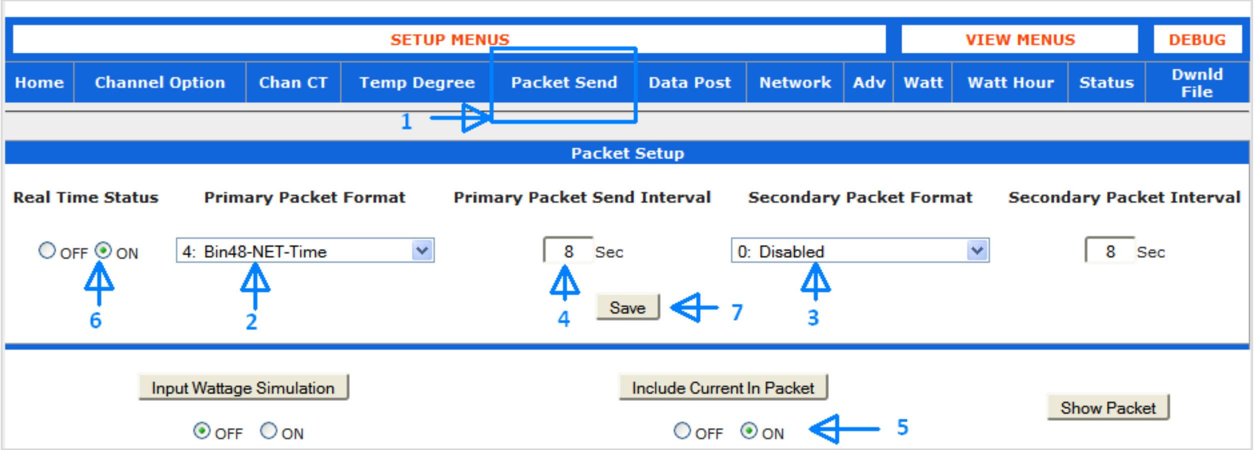
"Cosm" (cosm.com) was a free data hosting site which has recently been acquired by a different company and now charges for hosting. It uses HTTP "PUT" method and may be implemented to "PUT" data to other sites.

New SEG Format (14)

This format is used to post data to "Smart Energy Groups" data host (smartenergygroups.com). This is the replacement for the older version "SEG Format" option 6.

PACKET SETUP

Packet setup is done via the GEM’s setup page by clicking the “Packet Send” menu Figure 1 (1).



Real Time Status	Primary Packet Format	Primary Packet Send Interval	Secondary Packet Format	Secondary Packet Interval
○ OFF ● ON	4: Bin48-NET-Time	8 Sec	0: Disabled	8 Sec

Input Wattage Simulation: ● OFF ○ ON

Include Current In Packet: ○ OFF ● ON

Figure 1

Refer to Figure 1.

Select Packet Format

Select the desired packet format using the drop-down list (2). If the “Secondary Packet Format” (3) is set to disabled, the “Primary Format” will be sent via Com1 and Com2 ports. The GEM has the ability to send a different packet format to each Com port however the ”Secondary Packet” which sends data to Com2 port can only send non-HTTP data. The “Secondary Packet” is sent via Com2.

Send Interval

Presently the Primary Packet Send Interval (4) controls the interval for both “Primary” and “Secondary” packets. This represents how many seconds between each packet send. CAUTION! It is recommended not to set this interval to a value lower than five seconds.

Enable Packet Send

Enable Real-Time packet send (6) by selecting “ON”.

Save

Save changes by clicking the “Save” button (7).

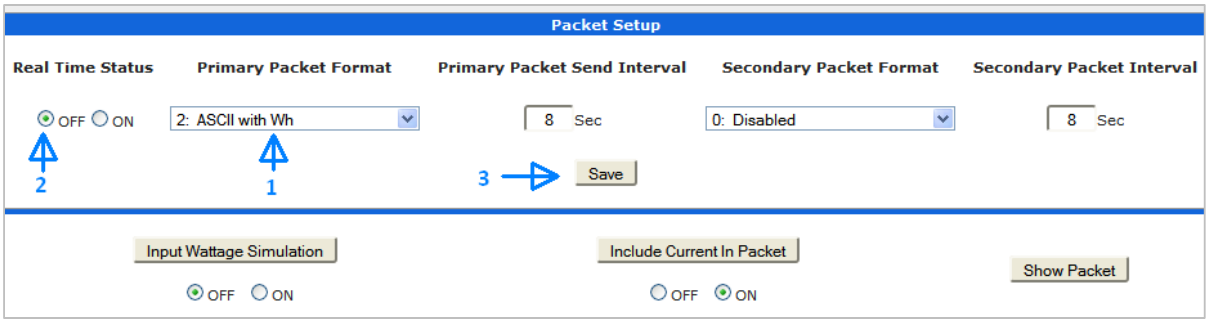
Include Current (Amps)

“Include Current in Packet” was added in COM firmware version 2.28 and ENG version 1.44. Starting with these versions, true RMS current was implemented. In order to preserve backward compatibility, the option has been added to include/exclude the current in certain packet formats. After selecting the desired radio button, click the “Include Current in Packet” button to save.

PULLING PACKETS

When pulling packets, the GEM communication remains silent until a command is sent from the host requesting a single packet.

SETUP FOR PULLING DATA



Packet Setup

Real Time Status	Primary Packet Format	Primary Packet Send Interval	Secondary Packet Format	Secondary Packet Interval
● OFF ○ ON	2: ASCII with Wh	8 Sec	0: Disabled	8 Sec

Save

Input Wattage Simulation ● OFF ○ ON

Include Current In Packet ○ OFF ● ON

Show Packet

Figure 2

Select packet format

Referring to Figure 2 (1), select the desired packet format.

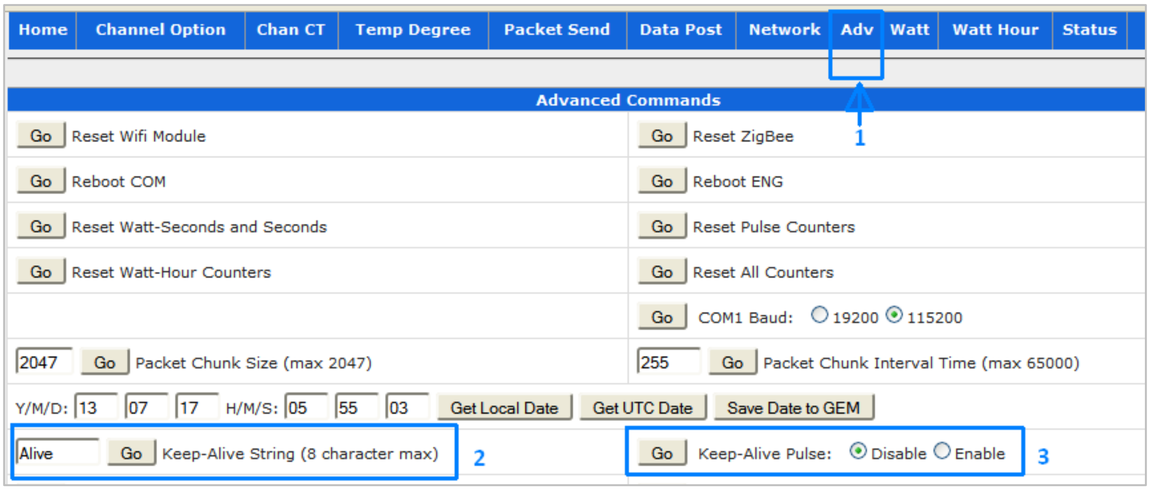
Turn OFF Real-Time

Set “Real Time Status” to OFF. See Figure 2 (2).

Enable/Disable Keep-Alive

If the GEM is setup as a TCP Client, the TCP server may close the socket connection after a period of time. Once this happens, the server cannot send a request for a packet because the there is no TCP connection. The TCP connection needs to be initiated by the Client which is the GEM in this case.

A technique which sends a “Keep-Alive” (or Heartbeat) byte or bytes may be used to keep the connection open. This is accomplished by sending the Keep-Alive at a given interval. The interval is set using the specified “Primary Packet Send Interval”. The character used for keep-alive is specified in the “Advanced” menu section of the GEM setup page. Figure 3 (1)



Advanced Commands

Go Reset Wifi Module	Go Reset ZigBee
Go Reboot COM	Go Reboot ENG
Go Reset Watt-Seconds and Seconds	Go Reset Pulse Counters
Go Reset Watt-Hour Counters	Go Reset All Counters
	Go COM1 Baud: ○ 19200 ● 115200
2047 Go Packet Chunk Size (max 2047)	255 Go Packet Chunk Interval Time (max 65000)
Y/M/D: 13 07 17 H/M/S: 05 55 03 Get Local Date	Get UTC Date Save Date to GEM
Alive Go Keep-Alive String (8 character max)	Go Keep-Alive Pulse: ● Disable ○ Enable

Figure 3

By default the keep-alive string is “Alive” but may be changed to any string between 1 to 8 characters. See Figure 3 (2). Remember to click “Go” to save the change.

Enable the keep-alive string using the section shown in Figure 3 (3). Again remember to click “go” to save the selection.

Request Packet Command

When using the “PULL” method, the host needs to send a command to request a packet. The command is:

“^^^APISPK”

Sending this command will cause the GEM to send a single packet if “Real Time” option is set to OFF.

LIST FORMAT (0)

This format is restricted for viewing real-time data values. It should not be used by a host for data collection.

Example

```
C 1: 2994 W / 27.58 A / 3155 VA          C 2: 84 W / 1.16 A / 132 VA
C 3: 3 W / 0.02 A / 2 VA                 C 4: 68 W / .82 A / 93 VA
C 5: 0 W / 0 A / 0 VA                    C 6: 1 W / 0.04 A / 4 VA
C 7: 0 W / 0 A / 0 VA                    C 8: 0 W / 0 A / 0 VA
C 9: 467 W / 4.18 A / 478 VA             C 10: 0 W / 0 A / 0 VA
C 11: 1 W / .10 A / 11 VA                C 12: 0 W / 0 A / 0 VA
C 13: 1582 W / 14.46 A / 1654 VA         C 14: 45 W / .54 A / 61 VA
C 15: 0 W / 0 A / 0 VA                   C 16: 0 W / 0 A / 0 VA
C 17: 0 W / 0 A / 0 VA                   C 18: 208 W / 2.68 A / 306 VA
C 19: 0 W / 0 A / 0 VA                   C 20: 18 W / .16 A / 18 VA
C 21: 7 W / 0.08 A / 9 VA                C 22: 0 W / 0 A / 0 VA
C 23: 101 W / 1.26 A / 144 VA            C 24: 257 W / 2.46 A / 281 VA
C 25: 0 W / 0 A / 0 VA                   C 26: 0 W / 0 A / 0 VA
C 27: 0 W / 0 A / 0 VA                   C 28: 0 W / 0 A / 0 VA
C 29: 0 W / 0 A / 0 VA                   C 30: 0 W / 0 A / 0 VA
C 31: 0 W / 0 A / 0 VA                   C 32: 0 W / 0 A / 0 VA

T1= 21.0 C  T2= 27.0 C  T3= 27.5 C  T4= x  T5= x  T6= x  T7= x  T8= x
P1= 1    P2= 0    P3= 0    P4= 0
Volt: 114.4        SEC: 5956900
F:60Hz
<EOP>
```

MULTI ECM-1240 SIMULATION (1)

This binary format was created early in the development of the GEM to allow the ECM-1240's Engine Software to accommodate data from the GEM. This is accomplished by treating the first seven GEM channels (CH1 to CH7) as the seven channels of ECM-1240 device-1 and the next seven channels (CH8 to CH14) as ECM-1240 device-2 and so on.

When using this format, the number of ECM-1240 devices to be simulated may be set to a value from 1-5. This is configured under the "Packet Send" menu in the "Packet Format Option 1" section.

ASCII WITH WH (2)

Regular ASCII data is used in this format in the form of:

Key1=Value&Key2=Value&………

This format may be the simplest to implement for those wishing to develop a custom application.

Key	Value
“n”	GEM serial number
“m”	Number of elapsed minutes corresponding to the elapsed watt-hour value
“wh_x”	Watt-hour consumption. “x” is the channel number represented.
“p_x”	Instantaneous power in “Watt” “x” is the channel number represented.
“a_x”	Instantaneous current in “Amp” “x” is the channel number represented.
“t_x”	Temperature in degree “C” or “F” depending on the selected option. “x” is the temperature channel number represented.
“c_x”	Pulse counter accumulated value “x” is the pulse counter channel number represented.
“v”	Line Voltage in “Volt”

Example

```
n=01000010&m=4&wh_1=223.19&p_1=3065&a_1=28.28&wh_2=6.22&p_2=84&a_2=1.14&wh_3=.28&
p_3=3&a_3=0.02&wh_4=5.06&p_4=69&a_4=.82&wh_5=.03&p_5=0&a_5=0&wh_6=.08&p_6=1&a_6=0.
04&wh_7=.01&p_7=0&a_7=0&wh_8=.01&p_8=0&a_8=0&wh_9=37.51&p_9=530&a_9=4.76&wh_10=.0
1&p_10=0&a_10=0&wh_11=.11&p_11=1&a_11=.10&wh_12=.01&p_12=0&a_12=0&wh_13=115.89&p_
13=1581&a_13=14.48&wh_14=3.37&p_14=46&a_14=.56&wh_15=.01&p_15=0&a_15=0&wh_16=.01&p
_16=0&a_16=0&wh_17=.01&p_17=0&a_17=0&wh_18=15.46&p_18=209&a_18=2.70&wh_19=.02&p_1
9=0&a_19=0&wh_20=1.35&p_20=18&a_20=.16&wh_21=.57&p_21=7&a_21=0.08&wh_22=.01&p_22=0
&a_22=0&wh_23=7.47&p_23=101&a_23=1.24&wh_24=19.20&p_24=267&a_24=2.58&wh_41=.00&wh
p_41=.00&p_41=0&a_41=17.20&t_1=21.0&t_2=26.5&t_3=27.0&v=114.4
```

HTTP GET (3)

This format is used to post data using HTTP which is typical when sending data over a network or the internet to a web server.

The full HTTP Format sends everything via GET statement, including expansion channels (expansion add-on is a planned future addition to the GEM system).

The structure of the GET statement looks like:

SN=######&SC=#&V=#&c1=#,#&c2=#,#&…&c48=#,#&PL=#,#,#,#&T=#,#,#,#,#,#,#,&Resp=

The table below explains the fields and their values.

Field	Type	Description
SN	Serial Number	The SN field contains your serial number.
SC	Seconds Counter	A counter that counts the amount of seconds elapsed.
V	Voltage	The field V has the current voltage; this value must be divided by 10 as voltage has 1 decimal point resolution.
C1..C48	Channels 1 through 48	Each individual channel with absolute wattseconds and polarized wattseconds separated by a comma. Includes future expansion board channels. As of firmware version COM ver 2.30 and ENG ver 1.44, the option to include current (Amp) has been made available. If enabled, an additional comma and "Amp" value is appended to each channel. The packet example in this chapter shows a packet which includes the load current.
PL	Pulse values	Up to 4 pulse values separated by commas.
T	Temperature Sensor values	The field T has up to 8 temperature sensors separated by commas. If a temperature sensor isn't being used, a null value will be inserted instead of the temperature value.
Resp	Response	

Example

HTTP GET example with **“Include Current in Packet”** option enabled

```
GET
sites//?SN=01000010&SC=5956977&V=1149&c1=356108415191,0,27.62&c2=474374184783,0,1.16&c3
=267233220038,0,0.02&c4=258836305667,0,.84&c5=465639463448,0,0&c6=219586310554,0,0.04&c7
=1641234,0,0&c8=2432143,0,0&c9=1578123495,0,3.88&c10=1296949,0,0&c11=63518439,0,.10&c12=7
43931336,0,0&c13=2666185110,0,14.50&c14=336662841,0,.56&c15=82205955,0,0&c16=218101154,0,
0&c17=4270608,0,0&c18=590448354,0,2.72&c19=2335935,0,0&c20=120208829,0,.16&c21=130404319
,0,0.08&c22=2086382,0,0&c23=4543725523,0,1.36&c24=1142327047,0,2.58&c25=51285,0,0&c26=389
84,65281,0&c27=51884,1,0&c28=37785,0,0&c29=51094,0,0&c30=44409,0,0&c31=69278,0,0&c32=3843
8,0,0&c33=0,0,445.44&c34=0,0,18.64&c35=0,0,154.26&c36=0,0,499.20&c37=0,0,655.50&c38=0,0,573.3
2&c39=0,0,302.08&c40=0,0,655.44&c41=0,0,17.20&c42=0,0,970.24&c43=0,0,655.36&c44=0,0,118.27&
c45=0,0,34.70&c46=0,0,690.04&c47=0,0,509.44&c48=0,0,655.48&PL=1,0,0,0&T=21.0,27.0,27.0,,,,,&Res
p= HTTP/1.1
Host: datahost.com
```

BIN48-NET-TIME (4)

Sends binary data for 48 channels (32 channels plus future 16 channel expansion), "watt-second" data for NET metering and a time stamp.

This is the format used by the DashBox when a direct serial connection is used or when communicating via XBee module.

Byte #	Type	# of Bytes	Description
1	FE	1 Byte	The header bytes are used to differentiate between the different binary packet formats and other Brultech Research Inc. devices.
2	FF	1 Byte	
3	05	1 Byte	
4-5	Voltage	2 Bytes Hi to Lo	The value given by the two bytes must be divided by 10 as voltage is provided with 1 decimal point resolution.
6-245	Absolute Wattseconds	5 Bytes x 48 Lo to Hi	Absolute Wattseconds counter for each channel. 1kW = 1,000 Watts and 1 Hour = 3600 Seconds 1KWh = 3,600,000 Watt Seconds
246-485	Polarized Wattseconds	5 Bytes x 48 Lo to Hi	Polarized Wattseconds counter for each channel. 1kW = 1,000 Watts and 1 Hour = 3600 Seconds 1KWh = 3,600,000 Watt Seconds
486-487	Serial Number	2 Bytes Hi to Lo	Pre-programmed Serial Number.
488	Reserved	1 Byte	Reserved in case of future use.
489	Device ID	1 Byte	Pre-programmed ID byte.
490-585	Current (Amp)	96 Bytes Lo to Hi	Current (Amp) for each channel The two-byte value must be divided by 50 to obtain the "Amp" value giving a resolution of ".02 Amp"
586-588	Seconds	3 Bytes Lo to Hi	Three byte continuous counter incrementing every second.
589-600	Pulse Counters	3 Bytes x 4 Lo to Hi	Three bytes for each pulse counter.
601-616	Temperature	2 Bytes x 8 Lo to Hi	Two bytes for each temperature sensor
617-622	Date/Time	6 Bytes	A single byte for year, month, day, hour, minute,

			second
623	FF	1 Byte	The footer bytes are used to show the end of the packet. They match the first 2 bytes in the packet.
624	FE	1 Byte	
625	Checksum	1 Byte	Checksum is used to validate that the packet is correct

BIN48-NET (5)

Sends binary data for 48 channels (32 channels plus future 16 channel expansion), “watt-second” data for NET metering. This packet resembles “Bin48-NET-Time” but excludes the time stamp.

Byte #	Type	# of Bytes	Description
1	FE	1 Byte	The header bytes are used to differentiate between the different binary packet formats and other Brultech Research Inc. devices.
2	FF	1 Byte	
3	05	1 Byte	
4-5	Voltage	2 Bytes Hi to Lo	The value given by the two bytes must be divided by 10 as voltage is provided with 1 decimal point resolution.
6-245	Absolute Wattseconds	5 Bytes x 48 Lo to Hi	Absolute Wattseconds counter for each channel. 1kW = 1,000 Watts and 1 Hour = 3600 Seconds 1KWh = 3,600,000 Watt Seconds
246-485	Polarized Wattseconds	5 Bytes x 48 Lo to Hi	Polarized Wattseconds counter for each channel. 1kW = 1,000 Watts and 1 Hour = 3600 Seconds 1KWh = 3,600,000 Watt Seconds
486-487	Serial Number	2 Bytes Hi to Lo	Pre-programmed Serial Number.
488	Reserved	1 Byte	Reserved in case of future use.
489	Device ID	1 Byte	Pre-programmed ID byte.
490-585	Current (Amp)	96 Bytes Lo to Hi	Current (Amp) for each channel The two-byte value must be divided by 50 to obtain the “Amp” value giving a resolution of “.02 Amp”
586-588	Seconds	3 Bytes Lo to Hi	Three byte continuous counter incrementing every second.
589-600	Pulse Counters	3 Bytes x 4 Lo to Hi	Three bytes for each pulse counter.
601-616	Temperature	2 Bytes x 8 Lo to Hi	Two bytes for each temperature sensor
617	FF	1 Byte	The footer bytes show the end of the packet. They match the first 2 bytes in the packet.
618	FE	1 Byte	
619	Checksum	1 Byte	Checksum is used to validate that the packet is correct.

SEG FORMAT (6)

This this the old method used to post data to “Smart Energy Groups” data host (smartenergygroups.com). This has been replaced by an improved version “New SEG Format (14)” which is preferred for new installations. This format should only be used for early GEM installations which used this format.

Example

```
PUT /sites/f9992346fcb9b4d HTTP/1.1
Host: api.smartenergygroups.com
Accept: */*
Content-Length: 565

(site f9992346fcb9b4d (node myhome ? (p_1 3139)(a_1 28.90)(p_2 85)(a_2 1.16)(p_3 3)(a_3 0.02)(p_4
69)(a_4 .82)(p_5 0)(a_5 0)(p_6 1)(a_6 0.04)(p_7 0)(a_7 0)(p_8 0)(a_8 0)(p_9 586)(a_9 5.26)(p_10
0)(a_10 0)(p_11 1)(a_11 .10)(p_12 0)(a_12 0)(p_13 1590)(a_13 14.56)(p_14 46)(a_14 .56)(p_15 0)(a_15
0)(p_16 0)(a_16 0)(p_17 0)(a_17 0)(p_18 209)(a_18 2.70)(p_19 0)(a_19 0)(p_20 18)(a_20 .18)(p_21
8)(a_21 0.08)(p_22 0)(a_22 0)(p_23 100)(a_23 1.24)(p_24 273)(a_24 2.64)(p_41 0)(a_41
17.20)(temperature_1 21.0)(temperature_2 27.0)(temperature_3 27.5)(voltage 114.1)))
```

BIN48-ABS (7)

This binary packet send 48 channel data (32 channels plus future 16 channel expansion). This packet does not include polarized watt-seconds required for net metering. **Requires COM firmware 2.42 or greater**.

Byte #	Type	# of Bytes	Description
1	FE	1 Byte	The header bytes are used to differentiate between the different binary packet formats and other Brultech Research Inc. devices.
2	FF	1 Byte	
3	05	1 Byte	
4-5	Voltage	2 Bytes Hi to Lo	The value given by the two bytes must be divided by 10 as voltage is provided with 1 decimal point resolution.
6-245	Absolute Wattseconds	5 Bytes x 48 Lo to Hi	Absolute Wattseconds counter for each channel. 1kW = 1,000 Watts and 1 Hour = 3600 Seconds 1KWh = 3,600,000 Watt Seconds
246-247	Serial Number	2 Bytes Hi to Lo	Pre-programmed Serial Number.
248	Reserved	1 Byte	Reserved in case of future use.
249	Device ID	1 Byte	Pre-programmed ID byte.
250-345	Current (Amp)	96 Bytes Lo to Hi	Current (Amp) for each channel The two-byte value must be divided by 50 to obtain the "Amp" value giving a resolution of ".02 Amp"
346-348	Seconds	3 Bytes Lo to Hi	Three byte continuous counter incrementing every second.
349-360	Pulse Counters	3 Bytes x 4 Lo to Hi	Three bytes for each pulse counter.
361-376	Temperature	2 Bytes x 8 Lo to Hi	Two bytes for each temperature sensor
377	FF	1 Byte	The footer bytes are used to show the end of the packet. They match the first 2 bytes in the packet.
378	FE	1 Byte	
379	Checksum	1 Byte	Checksum is used to validate that the packet is correct.

BIN32-NET (8)

Sends binary data for 32 channels and includes polarized "watt-second" data for NET metering.

Byte #	Type	# of Bytes	Description
1	FE	1 Byte	The header bytes are used to differentiate between the different binary packet formats and other Brultech Research Inc. devices.
2	FF	1 Byte	
3	07	1 Byte	
4-5	Voltage	2 Bytes Hi to Lo	The value given by the two bytes must be divided by 10 as voltage is provided with 1 decimal point resolution.
6-165	Absolute Wattseconds	5 Bytes x 32 Lo to Hi	Absolute Wattseconds counter for each channel. 1kW = 1,000 Watts and 1 Hour = 3600 Seconds 1KWh = 3,600,000 Watt Seconds
166-325	Polarized Wattseconds	5 Bytes x 32 Lo to Hi	Polarized Wattseconds counter for each channel. 1kW = 1,000 Watts and 1 Hour = 3600 Seconds 1KWh = 3,600,000 Watt Seconds
326-327	Serial Number	2 Bytes Hi to Lo	Pre-programmed Serial Number.
328	Reserved	1 Byte	Reserved in case of future use.
329	Device ID	1 Byte	Pre-programmed ID byte.
330-393	Current (Amps)	64 Bytes Lo to Hi	Current (Amp) for each channel The two-byte value must be divided by 50 to obtain the "Amp" value giving a resolution of ".02 Amp"
394-396	Seconds	3 Bytes Lo to Hi	Three byte continuous counter incrementing every second.
397-408	Pulse Counters	3 Bytes x 4 Lo to Hi	Three bytes for each pulse counter.
409-424	Temperature	2 Bytes x 8 Lo to Hi	Two bytes for each temperature sensor
425-426	Spare Bytes	2 Bytes	
427	FF	1 Byte	The footer bytes are used to show the end of the packet. They match the first 2 bytes in the packet.
428	FE	1 Byte	
429	Checksum	1 Byte	The checksum is used to validate the packet.

BIN32-ABS (9)

This binary packet send 48 channel data (32 channels plus future 16 channel expansion). This packet does not include polarized watt-seconds required for net metering. **Requires COM firmware 2.42 or greater**.

Byte #	Type	# of Bytes	Description
1	FE	1 Byte	The header bytes are used to differentiate between the different binary packet formats and other Brultech Research Inc. devices.
2	FF	1 Byte	
3	08	1 Byte	
4-5	Voltage	2 Bytes Hi to Lo	The value given by the two bytes must be divided by 10 as voltage is provided with 1 decimal point resolution.
6-165	Absolute WattSeconds	5 Bytes x 32 Lo to Hi	Absolute Watt-Second counter for each channel. 1kW = 1,000 Watts and 1 Hour = 3600 Seconds 1KWh = 3,600,000 Watt Seconds
166-167	Serial Number	2 Bytes Hi to Lo	Pre-programmed Serial Number.
168	Reserved	1 Byte	Reserved in case of future use.
169	Device ID	1 Byte	Pre-programmed ID byte.
170-233	Current (Amps)	64 Bytes Lo to Hi	Current (Amp) for each channel. The two-byte value must be divided by 50 to obtain the "Amp" value giving a resolution of ".02 Amp"
234-236	Seconds	3 Bytes Lo to Hi	Three byte continuous counter incrementing every second.
237-248	Pulse Counters	3 Bytes x 4 Lo to Hi	Three bytes for each pulse counter.
249-264	Temperature	2 Bytes x 8 Lo to Hi	Two bytes for each temperature sensor
265-266	Spare Bytes	2 Bytes	
267	FF	1 Byte	The footer bytes are used to show the end of the packet. They match the first 2 bytes in the packet.
268	FE	1 Byte	
269	Checksum		Checksum is used to validate that the packet is correct

NOT IMPLEMENTED (10)

Reserved for future format implementation.

Emon Packet Format

```
GET sites//?apikey=f9998636fcb9b4d&json={SN:01000010,SC:5959577,V:1144,E1:356116610095,P1:3031,E2:474374403823,P2:85,E3:267233402876,P3:54,E4:258836485041,P4:70,E5:465639463637,P5:0,E6:219586312634,P6:1,E7:1641246,P7:0,E8:2432160,P8:0,E9:1579540072,P9:434,E10:1296973,P10:0,E11:63521162,P11:1,E12:743931343,P12:0,E13:2670299918,P13:1586,E14:336781717,P14:46,E15:82205973,P15:0,E16:218101185,P16:0,E17:4270636,P17:0,E18:590997736,P18:213,E19:2335958,P19:0,E20:120255749,P20:18,E21:130423390,P21:7,E22:2086394,P22:0,E23:4543990674,P23:101,E24:1143012292,P24:260,E41:0,P41:0,T1:21.0,T2:26.5,T3:27.0,X:0} HTTP/1.1
Host: hostwebsite.com
```

UNIVERSAL DEVICE ISY (11)

Packets used to send GEM data to ISY controllers is sent in binary form and can only be transferred via ZigBee module.

NOT IMPLEMENTED (12)

Reserved for future implementation.

COSM (13)

“Cosm” (cosm.com) was a free data hosting site which has recently been acquired by a different company and now charges for hosting. It uses HTTP “PUT” method and may be implemented for use with other sites.

```
PUT /v2/feeds/double_packet.csv HTTP/1.1
Host: api.smartenergygroups.com
X-PachubeApiKey:
Content-Length: 450

E1,36942
P1,3020
E2,1020
P2,86
E3,42
P3,4
E4,840
P4,68
E5,2
P5,0
E6,11
P6,1
E7,0
P7,0
E8,0
P8,0
E9,6395
P9,477
E10,0
P10,0
E11,13
P11,1
E12,0
P12,0
E13,19034
P13,1587
E14,550
P14,46
E15,0
P15,0
E16,0
P16,0
E17,0
P17,0
E18,2525
P18,210
E19,0
P19,0
E20,216
P20,18
E21,87
P21,7
E22,0
P22,0
E23,1211
P23,100
E24,3153
P24,265
E41,0
P41,0
T1,21.0
T2,CTtype,CTrange,Phase
T3,CTtype,CTrange,Phase
V,1144
```

NEW SEG FORMAT (14)

```
PUT /sites/f9992346fcb9b4d HTTP/1.1
Host: api.smartenergygroups.com
Accept: */*
Content-Length: 567

(site f9992346fcb9b4d (node mygem ? (p_1 2828)(a_1 26.62)(p_2 86)(a_2 1.18)(p_3 150)(a_3 1.34)(p_4
68)(a_4 .80)(p_5 0)(a_5 0)(p_6 1)(a_6 0.04)(p_7 0)(a_7 0)(p_8 0)(a_8 0)(p_9 96)(a_9 1.02)(p_10 0)(a_10
0)(p_11 1)(a_11 0.08)(p_12 0)(a_12 0)(p_13 1650)(a_13 15.30)(p_14 44)(a_14 .52)(p_15 0)(a_15 0)(p_16
0)(a_16 0)(p_17 0)(a_17 0)(p_18 204)(a_18 2.72)(p_19 0)(a_19 0)(p_20 17)(a_20 .16)(p_21 7)(a_21
0.06)(p_22 1)(a_22 0)(p_23 100)(a_23 1.26)(p_24 257)(a_24 2.52)(p_41 0)(a_41 17.20)(temperature_1
21.0)(temperature_2 31.5)(temperature_3 31.5)(voltage 112.1)))
```

```
PUT /sites/f9992346fcb9b4d HTTP/1.1
Host: api.smartenergygroups.com
Accept: */*
Content-Length: 826

(site f9992346fcb9b4d (node mygem ? (e_1 278.23)(p_1 3155)(a_1 29.10)(e_2 7.68)(p_2 85)(a_2
1.16)(e_3 3.48)(p_3 153)(a_3 1.34)(e_4 6.24)(p_4 68)(a_4 .82)(e_5 .03)(p_5 0)(a_5 0)(e_6 .10)(p_6
1)(a_6 0.04)(e_7 .01)(p_7 0)(a_7 0)(e_8 .01)(p_8 0)(a_8 0)(e_9 45.80)(p_9 466)(a_9 4.18)(e_10 .01)(p_10
0)(a_10 0)(e_11 .14)(p_11 1)(a_11 .10)(e_12 .01)(p_12 0)(a_12 0)(e_13 143.12)(p_13 1584)(a_13
14.50)(e_14 4.16)(p_14 45)(a_14 .54)(e_15 .01)(p_15 0)(a_15 0)(e_16 .02)(p_16 0)(a_16 0)(e_17
.01)(p_17 0)(a_17 0)(e_18 19.10)(p_18 211)(a_18 2.74)(e_19 .02)(p_19 0)(a_19 0)(e_20 1.67)(p_20
18)(a_20 .16)(e_21 .70)(p_21 8)(a_21 0.08)(e_22 .01)(p_22 0)(a_22 0)(e_23 9.22)(p_23 100)(a_23
1.24)(e_24 23.72)(p_24 261)(a_24 2.52)(e_41 .00)(p_41 0)(a_41 17.20)(temperature_1
21.0)(temperature_2 26.5)(temperature_3 27.0)(voltage 114.2)))
```

BINARY PACKET FIELDS

PACKET HEADER

The packet header is unique to the binary packet. It is used to identify the start of the packet, and the type of Brultech device. In the case of the GreenEye Monitor, the final byte is 05. The ECM-1240, for example, ends with 03.

VOLTAGE

The voltage value is the sensed voltage taken from the PT. This value is sent with 1 decimal point resolution. Therefore, when parsing the voltage you must divide it by 10.

ABSOLUTE WATT-SECONDS

The absolute watt-seconds counters are granular representation of kilowatt-hour. They are continuously incrementing counters representing the watt-second value of energy. Each counter will increment regardless of whether the power is generated or consumed.

The counter begins at “zero” when the GreenEye Monitor is reset. If the energy monitor is never reset, this 5 byte counter will increment to a value equivalent to 305,419.896 KWh (years for a typical home). At this time the counter will roll-over to zero and start again.

POLARIZED WATT-SECONDS

The polarized watt-seconds counters are the same size as the absolute counters. The only difference is that the count will increment only if the energy is flowing in one direction; therefore, they may be used to record generated only or consumed only energy.

These counters are only required for applications such as solar or wind renewable energy systems connected to an inverter, producing power which is put back on the electrical power grid.

The energy direction which will cause this counter to increment may be set using one of the following methods:

- Send a “polarity” command to the GreenEye Monitor to change direction via setup software (or raw command). There are separate commands for each channel.
- Remove the CT for the channel in question and re-install so that the arrow on the CT points in the opposite direction.

SERIAL NUMBER

In terms of the binary packet format, the serial number contains the first 5 characters of the serial number. The entire serial number is attained by appending the ID byte to the beginning. For example, if your serial number was “0100013”, the Serial Number bytes will return “00013” and your Device ID will be “01”, appending “01” onto “00013” gives the full serial number.
The HTTP packet sends the full serial number with the ID byte pre-appended.

Device ID

The Device ID byte is unique to the binary packet; it is the start of the serial

Seconds Counter

The seconds counter starts at 0, and increments every second. The accumulated value is sent out each packet. The counter will roll over after 16777216 seconds. Provisions to deal with roll-over will be explained in section .

Pulse Counters

The pulse counters are similar to the watt-seconds counter.

Temperature

The temperature value is taken from its assigned sensor
If using an HTTP format, the temperature value outputted is the raw value from the sensor.
If you're using Binary, the **v**alue is **multiplied by 2** due to the 0.5 C resolution of the DS18B20. When the value is parsed via software, it must be **divided by 2** to obtain its value.

Date/Time

The six bytes represent year, month, day, hours, minutes, seconds respectively. Date/Time **must** be set prior to logging using a local computer.
Assembling the date format is dependent on the user.

End Bytes

The end bytes are unique to the binary packets and signify the end of the packet.

Checksum

This is the last byte of the binary packets which is the sum of all the previous bytes. This is represented by a single byte only which is the LSB of the sum. For example, if the value for the sum of all byte came to 513 (dec), the checksum value would be:

513 – 256 – 256 = 1

PARSING VALUES

Generating kWh and Watt values

Pseudocode

```
If prevSec > currSec
        secDiff = 256^3 - prevSec
        secDiff += currSec
Else
        secDiff = currSec - prevSec
Function GenerateValues (prevWS, currWS, secDiff)
        If prevWS > currWS
                wsDiff = 256^5 - prevWS
                wsDiff += currWS
        Else
                wsDiff = currWS − prevWS
        watt = wsDiff/secDiff
        kWh = wsDiff/3600000
        Return watt, kWh
```

VERSION HISTORY

2.1

- Added 48-ABS packet format.
- Updated the amperage section for each packet.