Messaging Protocol Version 0.4

The Waggle Team

2014-2016

For all the messages sent using protocol Version 0.4, the following fields will be standard across all headers, and across all devices.

Byte Field	Field Name	Value	Notes
0	Protocol Version	$0x04$	The Current Version of Protocol is Major 0, Minor 4.
10	Extended Header	$0x01$	Set if *Optional Key* is presented.
11	Optional Keys	$0xzz$	8–bit flags.

Notes:

- This version of protocol provides data transmission between beehive server and nodes. This supports three communication scenarios: 1) data transmission, 2) request/response, and 3) potentially a request initiated from the beehive server.
- The major differences in this version are –
  - The size of message header is 40 Bytes including CRC16 and is reduced by resizing the sender and responder's unique IDs by 2 bytes each.
  - A plugin in both server and node sides can be revealed by looking at the plugin ID in message.
  - Additional message types such as response of a request.
  - Data are serialized and compressed in message body.
  - Sending a large message exceeding 1 kB is supported using the *Optional_key* field.

1 Message Packet

<table>
<tr><th>0</th><th>1</th><th>2</th><th>3</th></tr>
<tr><td>Prot_Ver: Maj_N:Min_N (0)</td><td>Flag: Dev_P:Msg_P:Pref (1)</td><td colspan="2">Length of Message Body: Len_byte 2 (2), Len_byte 1 (3)</td></tr>
<tr><td colspan="4">Message Time Stamp: Time 4 (4), Time 3 (5), Time 2 (6), Time 1 (7)</td></tr>
<tr><td>Msg_Mj_Type (8)</td><td>Msg_Mi_Type (9)</td><td>Ext_header (10)</td><td>Optional_Key (11)</td></tr>
<tr><td colspan="4">S_UniqID_byte 8 (12), S_UniqID_byte 7 (13), S_UniqID_byte 6 (14), S_UniqID_byte 5 (15)</td></tr>
<tr><td colspan="4">S_UniqID_byte 4 (16), S_UniqID_byte 3 (17), S_UniqID_byte 2 (18), S_UniqID_byte 1 (19)</td></tr>
<tr><td colspan="4">R_UniqID_byte 8 (20), R_UniqID_byte 7 (21), R_UniqID_byte 6 (22), R_UniqID_byte 5 (23)</td></tr>
<tr><td colspan="4">R_UniqID_byte 4 (24), R_UniqID_byte 3 (25), R_UniqID_byte 2 (26), R_UniqID_byte 1 (27)</td></tr>
<tr><td colspan="2">Snd Session Number: Session_No_Hi (28), Session_No_Lo (29)</td><td colspan="2">Resp Session Number: Session_No_Hi (30), Session_No_Lo (31)</td></tr>
<tr><td colspan="3">Snd_Seq 3 (32), Snd_Seq 2 (33), Snd_Seq 1 (34)</td><td>Resp_Seq 3 (35)</td></tr>
<tr><td colspan="2">Resp_Seq 2 (36), Resp_Seq 1 (37)</td><td colspan="2">CRC_16_byte1 (38), CRC_16_byte2 (39)</td></tr>
</table>

40 Byte Packet Header

<table>
<tr><td>Payload</td></tr>
<tr><td>Payload</td></tr>
<tr><td>⋮</td></tr>
<tr><td>Payload</td></tr>
</table>

N Byte Message Payload

<table>
<tr><td>CRC_32 (39+Len(Data))</td></tr>
</table>

4 Byte Packet Footer

Parameter Description:

- **Header:**
  - *Prot_Ver: Maj_N:Min_N* — Major and minor version of the communication protocol used. They can take any value from 0 to 16 ($0xf$) each.
  - *Flag: Dev_P:Msg_P:Pref* — Indicator of priorities. Dev_P represents device priority, Msg_P indicates message priority, and Pref is a preference. Pref can be set to 'True' when the message refers to Msg_P for priority or 'False' when the message priority needs to get the highst priority.
  - *Length of Message Body* — Two bytes to describe the length of the payload only. Maximum size limited to 65 KB. However actual message packets will be limited to 1 kB.
  - *Message Time Stamp* — 4 byte Epoch time in seconds when the message was created.
  - *Msg_Mj_Type* — The 1 byte major type of the message. We envision 256 major message types, each with 256 minor types allowed.
  - *Msg_Mi_Type* — The 1 byte minor type of the message. We envision 256 minor message types, for each of 256 major types allowed.
  - *Ext_Header* — If it is set, refer to the *Optional_key* field.
  - *Optional_Key* — When *Ext_header* is set this field provides additional information in the message body.

7	6	5	4	3	2	1	0
S_PUID	R_PUID	Reserved for the future use					MMSG

- ∗ S_PUID: If set the first four bytes of the message body tell sender's plugin unique identifier (PUID), 4 bytes.
- ∗ R_PUID: If set the four bytes of the message body tell recipient's plugin unique identifier (PUID), 4 bytes.
- ∗ MMSG: Multiple message (MMSG) provides 6 bytes of chunk information in the message body. When it is set the 6 bytes consists of two parts: the next three bytes (if PUIDs exist; otherwise the first three bytes) provide chunk number of this message, the other three bytes represent the total number of chunks. This supports sending data with a size of up to 16 GB (ideally).

- *S_UniqID* — 8 byte unique ID of the sender. The assignment of the S_UniqID is separately documented. The 8 byte ID uniquely describes every endpoint, including the cloud.

  - *R_UniqID* — 8 byte unique ID of the receiver. The assignment of the R_UniqID is separately documented. The 8 byte ID uniquely describes every endpoint, including the cloud.
  - *Session Number* — 2 byte unique session ID of the sender. This ID changes when the sender goes through a power cycle of if all the sequence numbers of a session are consumed.
  - *Session Number* — 2 byte unique session ID of the receiver. This ID changes when the sender goes through a power cycle of if all the sequence numbers of a session are consumed.
  - *Snd_Seq* — 3 byte increasing number identifying the message for the sender-reciever pair, sent from the sender.
  - *Resp_Seq* — 3 byte sequence number if any of the message sent by the receiver to which the current message is a response to.
  - *CRC_16* — 2 byte CRC-16 of the message header.

- Payload
- Footer
  - CRC_32 — 4 byte CRC-32 of the Payload.

2 Database Queries

- List sensors:
  - all sensors available at *X mile* radius about *location* between *time start* and *time end*
  - all sensors available at *X mile* radius about *location* now
  - all sensors available at *X mile* radius about *location* at *time*
- All sensor data:
  - all sensor values from a *X mile* radius about *location* between *time start* and *time end*
  - all sensor values from a *X mile* radius about *location* now
  - all sensor values from a *X mile* radius about *location* at *time*
- Data from a particular sensor:
  - *sensor_uniq_id* values from a *X mile* radius about *location* between *time start* and *time end*
  - *sensor_uniq_id* values from a *X mile* radius about *location* now
  - *sensor_uniq_id* values from a *X mile* radius about *location* at *time*
  - *sensor_uniq_id* dataset time range
- Data from a particular type of sensor:
  - *Parameter* values from a *X mile* radius about *location* between *time start* and *time end*
  - *Parameter* values from a *X mile* radius about *location* now
  - *Parameter* values from a *X mile* radius about *location* at *time*
  - *Parameter* dataset time range

3 Message Types

Major Type	**Description**	**Minor Types**	**Notes**
$0x72$ (‘r’)	Registration	$0x69$ (‘i’), $0x75$ (‘u’), $0x72$ (‘r’), $0x6e$ (‘n’), $0x64$ (‘d’), $0x61$ (‘a’)	This message type will evolve to offer several registration based functions
$0x70$ (‘p’)	Heartbeat / Ping	$0x72$ (‘r’), $0x61$ (‘a’)	
$0x61$ (‘a’)	Acknowledgement	Any	
$0x74$ (‘t’)	Time	$0x72$ (‘r’), $0x61$ (‘a’), $0x75$ (‘u’)	
$0x73$ (‘s’)	Sensor Data	$0x64$ (‘d’)	
$0x6c$ (‘l’)	Location	$0x72$ (‘r’), $0x6d$ (‘m’), $0x65$ (‘e’), $0x6c$ (‘l’), $0x80$, $0x81$, $0x82$, $0x83$, $0x90$, $0x91$, $0x92$, $0x93$, $0xa0$, $0xa1$,$0xa2$,$0xa3$,$0xb0$,$0xb1$,$0xb2$,$0xb3$	
$0x64$ (‘d’)	Command / Response	$0x63$ (‘c’), $0x72$ (‘r’), $0x6f$ (‘o’), $0x73$ (‘s’)	
$0x63$ (‘c’)	Combined Message	$0x66$ (‘f’)	

3.1 Registration

3.1.1 Initial Registration Message — Deprecated

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x72$ (‘r’)	This is normally the first message sent by a new instance.
9	Message Minor Type	$0x69$ (‘i’)	
23–27	Reference Sequence ID	$0xzz$ $0xzz$ $0xzz$	Present if sent as a response to ‘rr’ message.

Payload:

In this protocol version, we will use the current registration message format in the body.

3.1.2 Registration Update — Not Implemented

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x72$ (‘r’)	A node can update its Registration with this message.
9	Message Minor Type	$0x75$ (‘u’)	

Payload:

In this protocol version, both initial and updated registrations will contain the full registration message. The update only alerts the cloud to look for an existing registration. The cloud may update or overwrite the old registration.

3.1.3 Request for Registration

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x72$ (‘r’)	
9	Message Minor Type	$0x72$ (‘r’)	

Payload:

Meta data of the requestor (e.g., name, version, instance, etc.).

3.1.4 Request for Configuration Registration

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	0x72 (‘r’)	
9	Message Minor Type	0$x6e$ (‘n’)	

Payload:

Configuration information of the sender.

3.1.5 Request for De-registration

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	0x72 (‘r’)	
9	Message Minor Type	0x64 (‘d’)	
2–3	Length of Message Body	0x00 0x00	Empty Message.

Payload:

None.

3.1.6 Registration Response

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	0x72 (‘r’)	
9	Message Minor Type	0x61 (‘a’)	

Payload:

Responder’s message.

3.2 Alive Heartbeat

3.2.1 Onetime Heartbeat Ping Request

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	0x70 (‘p’)	
9	Message Minor Type	0x72 (‘r’)	
2–3	Length of Message Body	0x00 0x00	Empty Message.

Payload:

None.

3.2.2 Onetime Heartbeat Pong

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x70$ (‘p’)	
9	Message Minor Type	$0x61$ (‘a’)	
2–3	Length of Message Body	$0x00$ $0x04$	4 Byte message.

Payload:

“Pong”

3.3 Acknowledgement

3.3.1 Message Receipt – Not Implemented

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x61$ (‘a’)	
9	Message Minor Type	$0xzz$	Type of the message being acknowledged.
2–3	Length of Message Body	$0x00$ $0x00$	Empty Message.
35–37	Reference Sequence ID	$0xzz$ $0xzz$ $0xzz$	Sequence number of the message being acknowledged.

Payload:

None.

3.3.2 Completion of Task

This message will change on a case by case basis. I am not sure if we will have to classify this under the acknowledgement case, but I think we should have some way of Acknowledging that.

3.4 Time

`http://tools.ietf.org/html/rfc958`

3.4.1 Request Current Time

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x74$ (‘t’)	
9	Message Minor Type	$0x72$ (‘r’)	
2–3	Length of Message Body	$0x00$ $0x00$	Empty Message.

Payload:

None.

3.4.2 Current Time Response

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x74$ (‘t’)	
9	Message Minor Type	$0x61$ (‘a’)	
17–19	Length of Message Body	$0x00$ $0xzz$	N–bytes message.

Payload:

A floatting point number of the current time since the epoch.

3.4.3 Time Update – Not Implemented

Header:

Byte Field	Field Name	Value	Notes
23	Message Major Type	$0x74$ (‘t’)	
24	Message Minor Type	$0x75$ (‘u’)	
17–19	Length of Message Body	$0x00$ $0x00$ $0x04$	4 Byte message.

Payload:

0	1	2	3
Time 4 (0), Time 3 (1), Time 2 (2), Time 1 (3)			

Time Representation:
Current Epoch Time_Sec = (Time 4 << 24) + (Time 3 << 16) + (Time 2 << 8) + (Time 1)

3.5 Sensor Data

The data is serialized and compressed before sending.

When size of data exceeds the maximum limit of data length (or limit that system designer set) multiple Waggle messages can be used to send the data. For this case, both *Ext_header* and *MMSG* in *Optional_key* fields will be set. In addition, there are 6 bytes of information added in the message body of each message (See parameter description in **??**). *Snd_Seq* will be the same for all the sub messages.

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x73$ (‘s’)	
9	Message Minor Type	$0x64$ (‘d’)	

Payload:

Serialized data that are compressed.

3.6 Location

`http://dev.w3.org/geo/api/spec-source.html#api_description`
`http://resources.arcgis.com/en/help/main/10.1/index.html#//009t0000023w000000`

3.6.1 Request Current Location – Not Implemented

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x6c$ (‘l’)	
9	Message Minor Type	$0x72$ (‘r’)	
2–3	Length of Message Body	$0x00$ $0x00$	Empty Message.

Payload:

None.

3.6.2 Location Standardized – Not Implemented

We will use WGS84 geodetic datum.

- **Latitude:** Latitude and Longitude of point. Northern latitudes are positive, southern latitudes are negative. eastern longitudes are positive, western longitudes are negative. Valid formats include (Max 16 allowed):
  - N4338’19.39”, W11614’28.86” - **LatLon Type:** $0x00$
  - 4338’19.39”N, 11614’28.86”W - **LatLon Type:** $0x01$

- 43 38 19.39, -116 14 28.86 - **LatLon Type:** $0x02$
- 43.63871944444445, -116.2413513485235 - **LatLon Type:** $0x03$

- **Altitude:** Elevation of the point. Valid formats include (Max 8 allowed):
  - Orthometric in Meters - **Elevation Type:** $0x00$
  - Geoid in Meters - **Elevation Type:** $0x01$
  - Ellipsoidal in Meters - **Elevation Type:** $0x02$
  - Non-standard (10 m above pedestal) in Meters - **Elevation Type:** $0x03$

There are 16 different formats for representation of position, which are derived using a combination of the 4 LatLon and 4 Elevation types. The minor type uses the upper nibble to state the Latitude and Longitude type and the lower nibble for the Elevation type.

$$Message\ Minor\ Type\ =\ 0x80\ |\ Elevation\ Type\ <<4\ |\ LatLon\ Type$$

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x6c$ ('l')	
9	Message Minor Type	*Generated*	16 different representations.

Payload:

We will have a delimited payload of the following kind -
*Latitude*_[0]_*Longitude*_[0]_*Elevation*

3.6.3 Location Meta – Not Implemented

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x6c$ ('l')	
9	Message Minor Type	$0x6d$ ('m')	Meta Data of the location.

Payload:

The payload of the message will be considered as a text string describing the location. This is for human readable description like - Under a tree, In the shade, on the top of the HVAC unit and so on.

3.6.4 Get Location Estimator Type – Not Implemented

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x6c$ (‘l’)	
9	Message Minor Type	$0x65$ (‘e’)	
2–3	Length of Message Body	$0x00$ $0x00$	Empty Message.

Payload:

None.

3.6.5 Location Estimator Type – Not Implemented

This message will both provide location estimator type and also the error bounds for the Latitude, Longitude and Elevation.

Estimator Types:

- Preset Static - **Position Type:** $0x00$
- Satellite - **Position Type:** $0x01$
- Dead Reckoning or Software Estimation - **Position Type:** $0x02$
- Ranging - **Position Type:** $0x03$

Errors Types:

- Linear Error (LE90) in meters - **Error Type:** $0x00$
- Circular Error (CE90) in meters - **Error Type:** $0x01$

$$\mathit{Estimation\ Type} \;=\; \mathit{Position\ Type} \;<<\; 4 \,|\, (\mathit{Error\ Type})$$

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x6c$ (‘l’)	
9	Message Minor Type	$0x6c$ (‘l’)	

Payload:

Delimited payload -
Latitude_Estimation_Type_[1]_Latitude_Error_[0]_
Longitude_Estimation_Type_[1]_Longitude_Error
_[0]_Elevation_Estimation_Type_[1]_Elevation_Error

3.6.6 Set Location Interrupt

The following will depend on the Location Engine. What are the features and facilities we want to include in this engine? Can we set location alerts? i.e. alert me when you have reached location x,y,z, alert me when you have moved further than 10 ft from base (a floating platform?). May not be a core waggle feature, but something the particular instance will implement.

3.6.7 Get Current Location Interrupt

The following will depend on the Location Engine. What are the features and facilities we want to include in this engine? Can we set location alerts? i.e. alert me when you have reached location x,y,z, alert me when you have moved further than 10 ft from base (a floating platform?). May not be a core waggle feature, but something the particular instance will implement.

3.7 Direct Command Message

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x64$ (‘d’)	
9	Message Minor Type	$0x63$ (‘c’)	

Payload:

Specific Payload

3.8 Direct Request Message

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x64$ (‘d’)	
9	Message Minor Type	$0x72$ (‘r’)	

Payload:

Specific payload.

3.9 Direct OS Specific Request Message

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x64$ (‘d’)	
9	Message Minor Type	$0x6f$ (‘o’)	

Payload:

Specific payload.

3.10 Direct Shell Command

Header:

Byte Field	Field Name	Value	Notes
8	Message Major Type	$0x64$ (‘d’)	
9	Message Minor Type	$0x73$ (‘s’)	

Payload:

Specific payload.

3.11 Combined Message – Needs revision...

- Packet Encapsulation
  - Check individual messages being forwarded together for integrity.
  - Arrange individual messages one after another in the forwarding message body.
  - Compute packet CRC32.
- Packet Decoding
  - Check Header and Message CRC.
  - Read first 32 bytes in body for the header of the first encapsulated message.
  - Extract first message body and CRC32.
  - Process the encapsulated message as a regular message.
  - Continue until the end of all encapsulated messages.

Byte Field	Field Name	Value	Notes
8	Message Major Type	0*x*63 (‘c’)	
9	Message Minor Type	0*x*66 (‘f’)	

<table>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th></th></tr>
<tr><td>Prot_Ver: Maj_N:Min_N (0)</td><td>Flag: Dev_P:Msg_P:Pref (1)</td><td colspan="2">Length of Message Body: Len_byte 2 (2), Len_byte 1 (3)</td><td rowspan="10">40 Byte
Packet
Header</td></tr>
<tr><td colspan="4">Message Time Stamp: Time 4 (4), Time 3 (5), Time 2 (6), Time 1 (7)</td></tr>
<tr><td>Msg_Mj_Type (8)</td><td>Msg_Mi_Type (9)</td><td>Ext_Header (10)</td><td>Optional Key (11)</td></tr>
<tr><td colspan="4">S_UniqID_byte 8 (12), S_UniqID_byte 7 (13), S_UniqID_byte 6 (14), S_UniqID_byte 5 (15)</td></tr>
<tr><td colspan="4">S_UniqID_byte 4 (16), S_UniqID_byte 3 (17), S_UniqID_byte 2 (18), S_UniqID_byte 1 (19)</td></tr>
<tr><td colspan="4">R_UniqID_byte 8 (20), R_UniqID_byte 7 (21), R_UniqID_byte 6 (22), R_UniqID_byte 5 (23)</td></tr>
<tr><td colspan="4">R_UniqID_byte 4 (24), R_UniqID_byte 3 (25), R_UniqID_byte 2 (26), R_UniqID_byte 1 (27)</td></tr>
<tr><td colspan="2">Snd Session Number: Session_No_Hi (28), Session_No_Lo (29)</td><td colspan="2">Resp Session Number: Session_No_Hi (30), Session_No_Lo (31)</td></tr>
<tr><td colspan="3">Snd_Seq 3 (32), Snd_Seq 2 (33), Snd_Seq 1 (34)</td><td>Resp_Seq 3 (35)</td></tr>
<tr><td colspan="2">Resp_Seq 2 (36), Resp_Seq 1 (37)</td><td colspan="2">CRC_16_byte1 (38), CRC_16_byte2 (39)</td></tr>
</table>

<table>
<tr><td>[Header1][Body1][Footer1]</td><td rowspan="4">N Byte
Message
Payload</td></tr>
<tr><td>[Header2][Body2][Footer2]</td></tr>
<tr><td>⋮</td></tr>
<tr><td>...</td></tr>
</table>

<table>
<tr><td>CRC_32 (39+Len(N–Bytes Data))</td><td>4 Byte
Packet
Footer</td></tr>
</table>