Open Oceans Partner Training
Detailed Software Interfaces

Payload Protocol

Mike Cookson
Director of Sensor Engineering

LIQUID ROBOTICS®

Module Agenda

Objectives

- Understand the communication protocol (**Payload Protocol**) that runs over the SV3 Sensor port (this was called a payload port on SV2 Wave Gliders)
- Get a feel for the scope of the implementation task
- Gain knowledge to decide the approach to take when developing payload software
- Prepare for implementation

Approach

- Review the message formats and structures
- Walk through the primary use cases
- Discuss reference implementation

Note that this presentation is not intended as a reference manual: please see "Recommended Reading" on the last page.

©2016. Company Confidential.

Definitions

PIB: Payload Interface Board, a controller often used in sensor port-based payloads to implement simple interfaces or to act as a communication multiplexer

S2S: Ship-to-Shore, a designation for the messaging between WGMS and the Wave Glider

SMC: Sensor Management Computer, a programmable controller used for sensor interfaces and customer applications

VMC: Vehicle Management Computer, the navigation computer on SV3 Wave Gliders

WCS: Write Console Script, a message that carries arbitrary text commands from WGMS to modules on the Wave Glider

WGMS: Wave Glider Management System, the shore-side web application and infrastructure used for piloting Wave Gliders, sending commands and recording telemetry messages

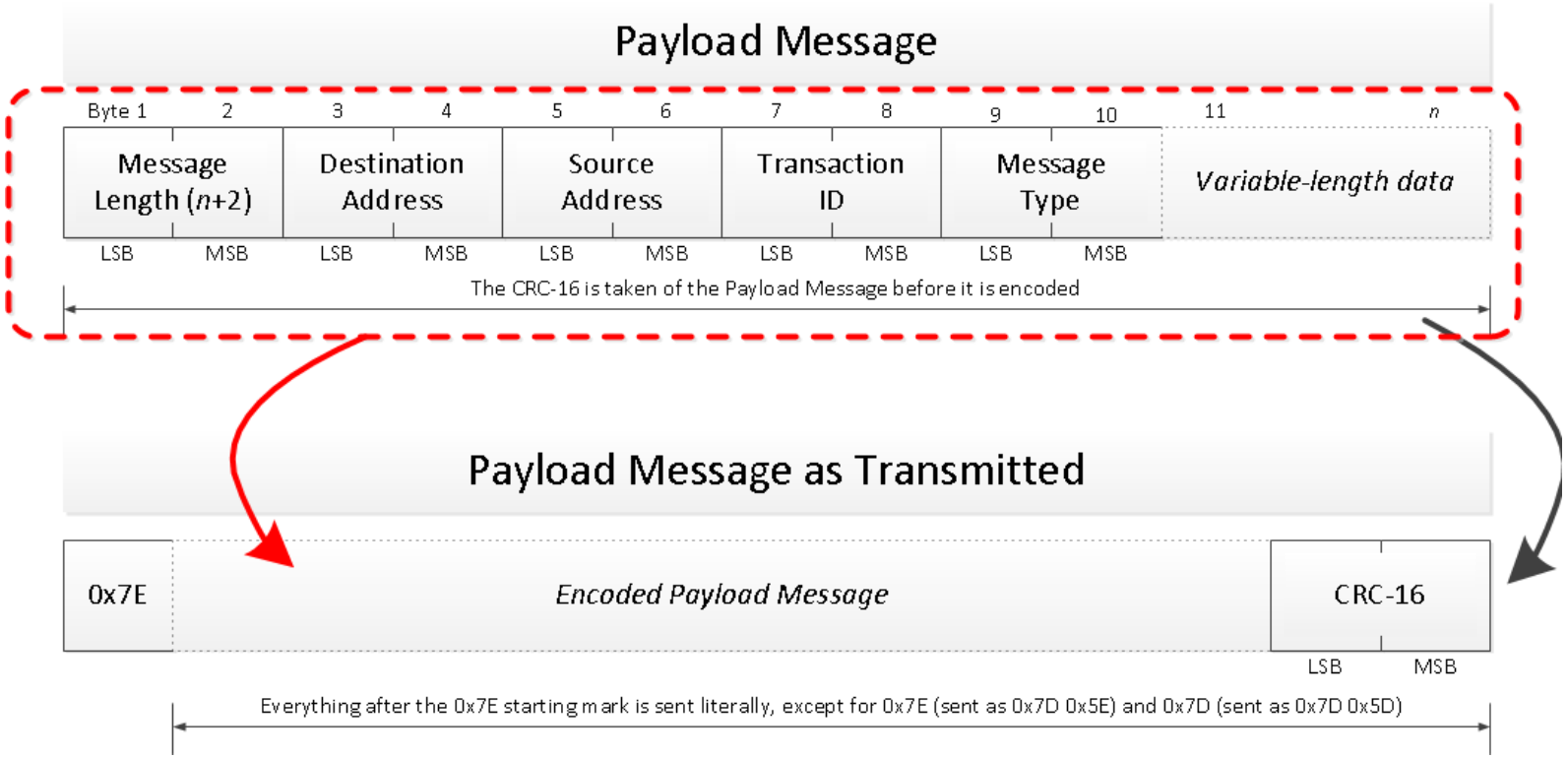
©2016. Company Confidential.

Payload Software Interface Requirements

- Pass commands from shore-side infrastructure and navigation computer to payload and return responses
- Allow telemetry messages and alarm notifications to be sent from payload to navigation computer and shore-side infrastructure
- Allow for description of payload to be propagated to shore
- Allow payload access to navigation computer resources and potentially to other payloads
- Coordinate startup, registration and shutdown/power-off

©2016. Company Confidential.

Payload Protocol Formatting/Encoding



©2016. Company Confidential.

What is a CRC?

- CRC (cyclic-redundancy check) is an algorithm for detecting and sometimes correcting errors in transmitted data
- The result of a CRC calculation is a number (Payload Protocol uses a 16-bit integer) that is sent along with the data as a mathematical summary. The receiver re-runs the CRC calculation and compares the transmitted CRC number.
- There are a number of CRC calculators available on-line; the following is an example:

http://www.lammertbies.nl/comm/info/crc-calculation.html

©2016. Company Confidential.

Payload Message Fields

- The *Message Length* field indicates the total number of bytes in the message before it is encoded, including 2 bytes for the CRC.
  - > The receiver resynchronizes its state machine whenever a raw 0x7E is received (encoding should prevent any 0x7E's in the remainder of the message or the checksum)
  - > The receiver can evaluate its buffer when the total number of bytes received equals the value of *Message Length* after decoding.
- The *Destination Address* indicates which domain and service within that domain the message should be delivered to.
  - > For some message types, 0xFF indicates a broadcast across all domains or all services within a domain.
- The *Source Address* indicates which domain and services any output messages should be sent to (the source address is often used as the destination address in responses.)

©2016. Company Confidential.

Payload Message Fields

- The *Transaction ID* field is set by the sender of a command. Any resulting output messages (ACK or NAK) should use the same transaction ID as was present in the command message so that the sender can match up responses with commands.
- The *Message Type* field indicates the over-all meaning of the message and determines how the remainder of the message should be interpreted.
  - NAK (0) or "Negative Acknowledge" is sent as a response to a command that is rejected or fails
  - ACK (1) or "Acknowledge" is sent as a response to a command that is accepted or succeeds, and sometimes includes additional resulting data
  - REQUEST_ID (0x10) or "Enumeration Request" is a command that is broadcast across domains on a payload to request a description of the domains present
  - FLOAT_TELEMETRY_DATA (0x15) is sent as a command to carry copies telemetry messages when forwarding is requested
  - REQUEST_PIB_STATUS (0x22) or "Status Request" is a polling command sent periodically from the Host to each enumerated domain
  - SEND_FWD_ICD_CMD (0x24) forwards a copy of a command or a command for execution to the payload (Write Console Script or file transfer messages)
  - POWER_NOTIFICATION (0x30) is sent when the payload is about to be turned off, giving the software a chance to shut down gracefully
  - REQUEST_WAITING_PACKET (0x40) is a command to pick up any waiting messages queued for transmission
  - ACK_REQUESTED_PACKET (0x41) forwards an acknowledgement message to a message sent from a service
- The high order bit of the Message Type field indicates whether a command message expects a response (0) or if no response should be given (1, or add 32768 to the message type)

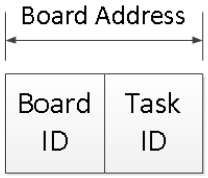
©2016. Company Confidential.

Payload Addressing

A *Board Address* is a 16-bit integer that identifies a particular service that is the target of a command invocation or is a source of data.

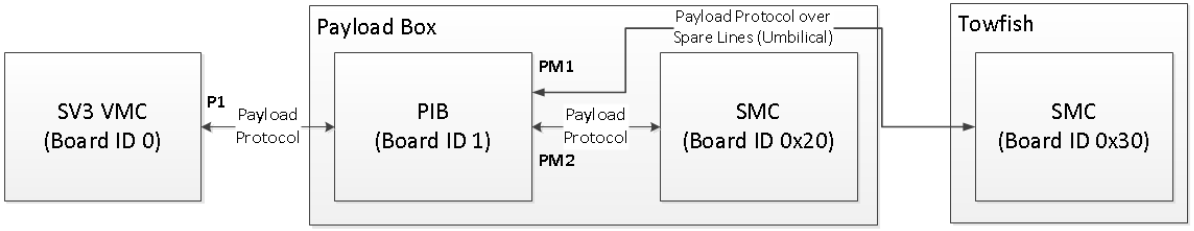
The upper byte of the Board Address is the *Board ID*. It uniquely defines a domain on a vehicle in which one or more services can exist. Typically, a Board ID represents a computer.

The lower byte of the Board Address is the *Task ID*. It defines a particular service within its domain.



For Board Address 8192 (0x2000), the Board ID is 32 (0x20) and the task ID is 0.

Board Address = 256*Board ID + Task ID



0x0000 = Generic Float C&C
0x0001 = Command Interface
0x0002 = Telemetry Interface
0x0003 = Debug Interface (SV2 only)
0x0010 = Message Forwarding (Write Console Acks)

0x0100 = PIB Configuration
0x0101 = PM1 Access
0x0102 = PM2 Access
0x0103 = PM3 Access

0x2000 = PayloadInterface
0x2010 = Teledyne RDI ADCP
0x2025 = GPSWaves
0x2026 = Seabird GPCTD

0x3000 = PayloadInterface
0x3012 = WetLabs EcoPuck
0x3026 = Seabird GPCTD

©2016. Company Confidential.

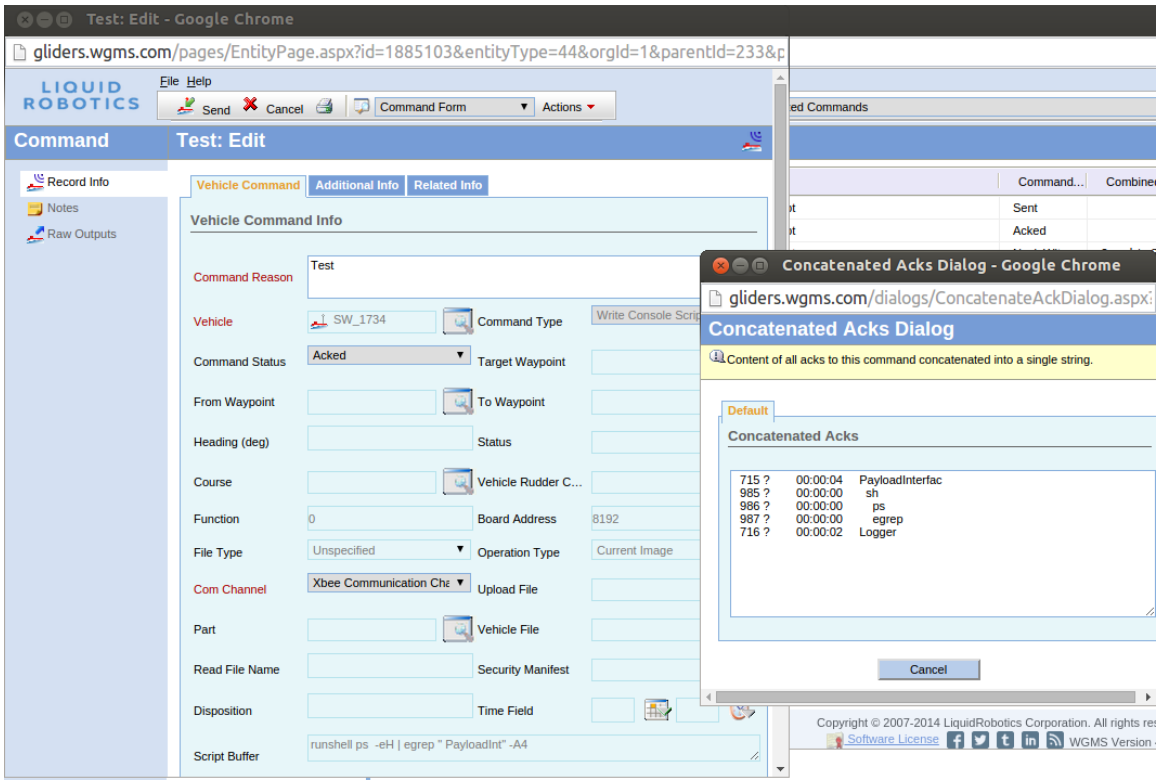
Payload Addressing

- Board ID's are arbitrary on a given vehicle, except that board ID 0 is the VMC and board ID 0xFF is reserved for broadcasting.
- Board ID's must be unique on a vehicle

  Implementers: please allow the board ID to be reconfigured
- The Write Console Script message uses board addresses to determine which service will receive the command.
  - > When using Sensor ports, the board address is an important designator for how to target commands on the payload.
  - > When using Expansion ports, the board address is always zero because the format of the command indicates the target.
- Telemetry messages are tagged with the Board Address to indicate the source of the data, even over Expansion ports

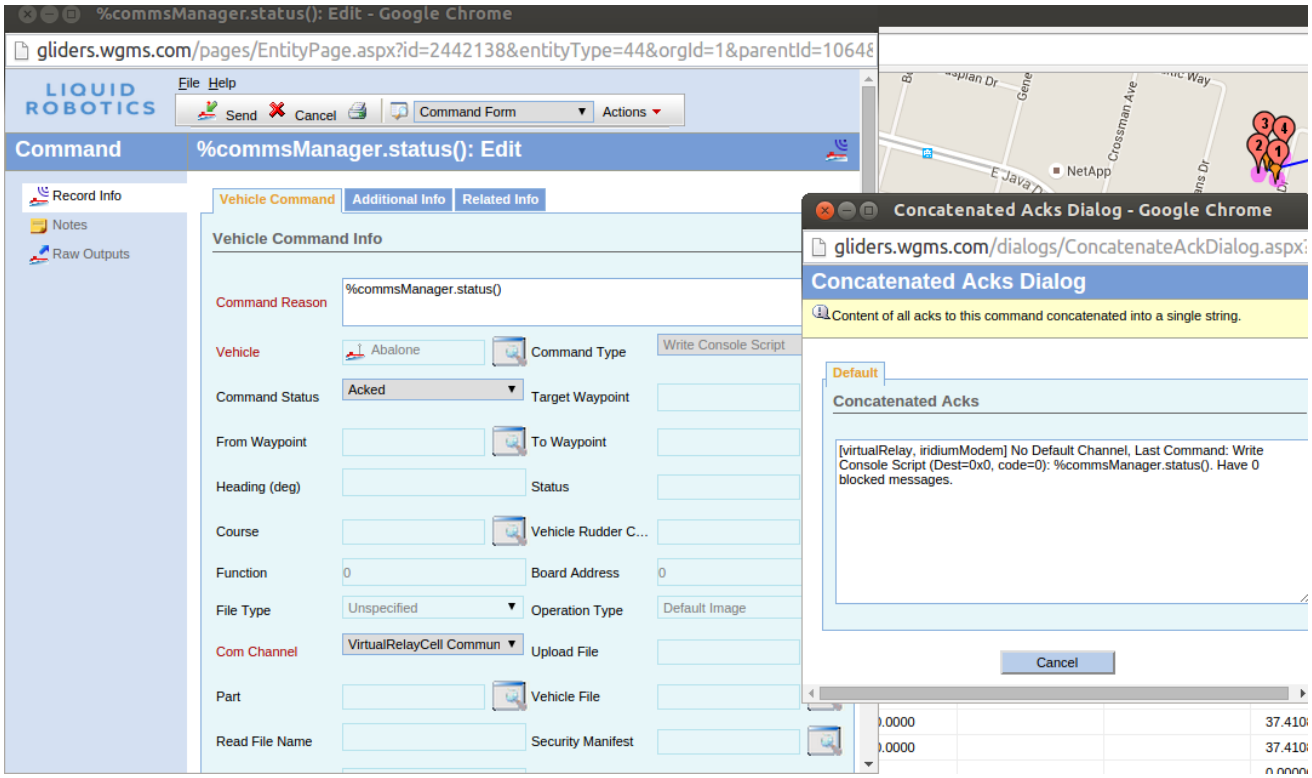
©2016. Company Confidential.

WCS to Sensor Port Payload



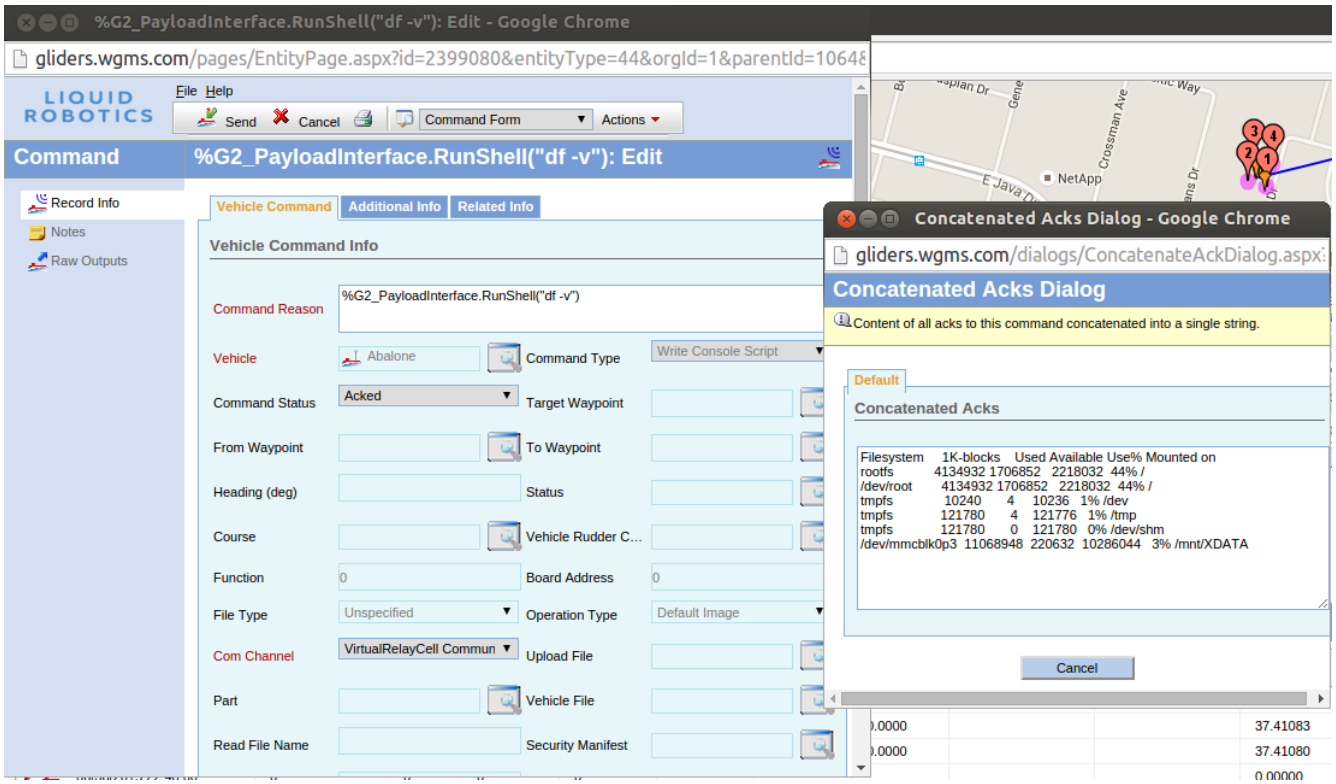
©2016. Company Confidential.

WCS Command to VMC



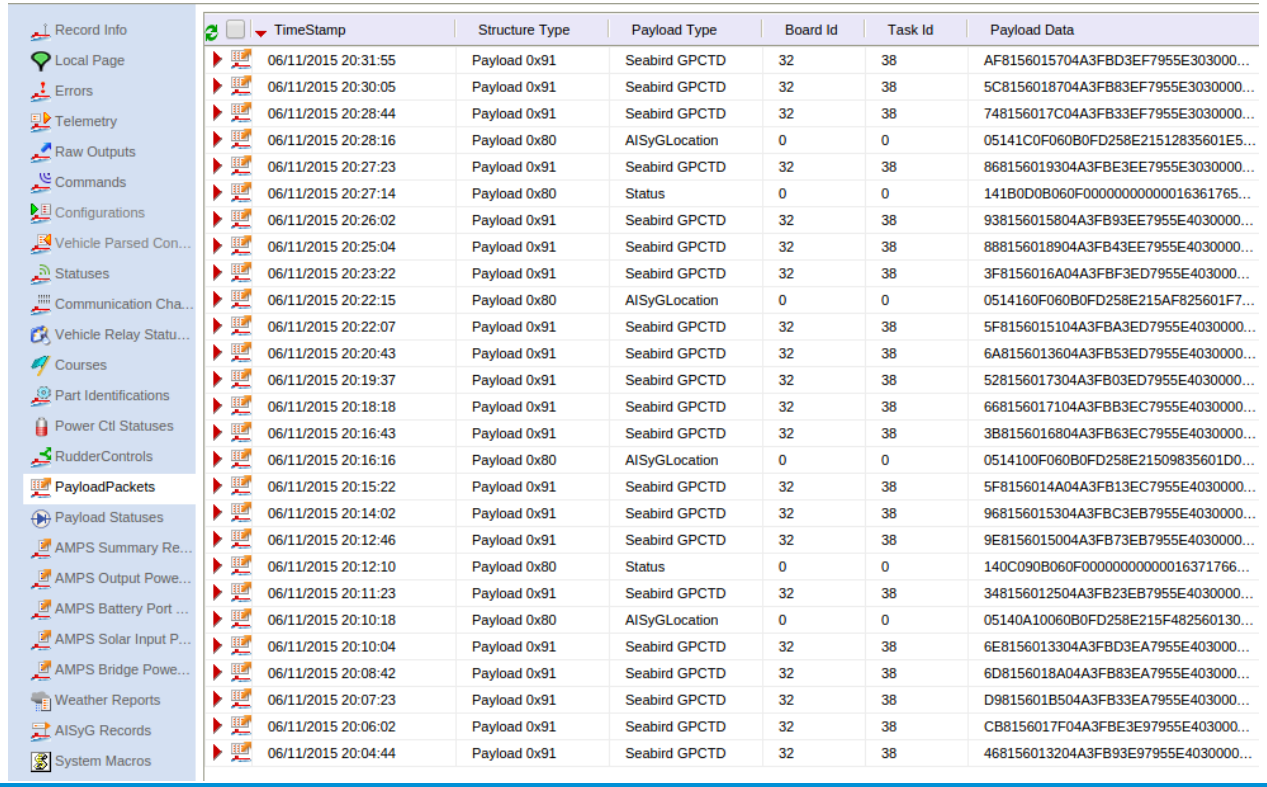
©2016. Company Confidential.

WCS to Expansion Port Payload



©2016. Company Confidential.

Telemetry with Board ID & Task ID



TimeStamp	Structure Type	Payload Type	Board Id	Task Id	Payload Data
06/11/2015 20:31:55	Payload 0x91	Seabird GPCTD	32	38	AF8156015704A3FBD3EF7955E303000...
06/11/2015 20:30:05	Payload 0x91	Seabird GPCTD	32	38	5C8156018704A3FB83EF7955E3030000...
06/11/2015 20:28:44	Payload 0x91	Seabird GPCTD	32	38	748156017C04A3FB33EF7955E3030000...
06/11/2015 20:28:16	Payload 0x80	AISyGLocation	0	0	05141C0F060B0FD258E21512835601E5...
06/11/2015 20:27:23	Payload 0x91	Seabird GPCTD	32	38	868156019304A3FBE3EE7955E3030000...
06/11/2015 20:27:14	Payload 0x80	Status	0	0	141B0D0B060F0000000000016361765...
06/11/2015 20:26:02	Payload 0x91	Seabird GPCTD	32	38	938156015804A3FB93EE7955E4030000...
06/11/2015 20:25:04	Payload 0x91	Seabird GPCTD	32	38	888156018904A3FB43EE7955E4030000...
06/11/2015 20:23:22	Payload 0x91	Seabird GPCTD	32	38	3F8156016A04A3FBF3ED7955E403000...
06/11/2015 20:22:15	Payload 0x80	AISyGLocation	0	0	0514160F060B0FD258E215AF825601F7...
06/11/2015 20:22:07	Payload 0x91	Seabird GPCTD	32	38	5F8156015104A3FBA3ED7955E4030000...
06/11/2015 20:20:43	Payload 0x91	Seabird GPCTD	32	38	6A8156013604A3FB53ED7955E4030000...
06/11/2015 20:19:37	Payload 0x91	Seabird GPCTD	32	38	528156017304A3FB03ED7955E4030000...
06/11/2015 20:18:18	Payload 0x91	Seabird GPCTD	32	38	668156017104A3FBB3EC7955E4030000...
06/11/2015 20:16:43	Payload 0x91	Seabird GPCTD	32	38	3B8156016804A3FB63EC7955E4030000...
06/11/2015 20:16:16	Payload 0x80	AISyGLocation	0	0	0514100F060B0FD258E21509835601D0...
06/11/2015 20:15:22	Payload 0x91	Seabird GPCTD	32	38	5F8156014A04A3FB13EC7955E4030000...
06/11/2015 20:14:02	Payload 0x91	Seabird GPCTD	32	38	968156015304A3FBC3EB7955E4030000...
06/11/2015 20:12:46	Payload 0x91	Seabird GPCTD	32	38	9E8156015004A3FB73EB7955E4030000...
06/11/2015 20:12:10	Payload 0x80	Status	0	0	140C090B060F0000000000016371766...
06/11/2015 20:11:23	Payload 0x91	Seabird GPCTD	32	38	348156012504A3FB23EB7955E4030000...
06/11/2015 20:10:18	Payload 0x80	AISyGLocation	0	0	05140A10060B0FD258E215F482560130...
06/11/2015 20:10:04	Payload 0x91	Seabird GPCTD	32	38	6E8156013304A3FBD3EA7955E403000...
06/11/2015 20:08:42	Payload 0x91	Seabird GPCTD	32	38	6D8156018A04A3FB83EA7955E403000...
06/11/2015 20:07:23	Payload 0x91	Seabird GPCTD	32	38	D9815601B504A3FB33EA7955E403000...
06/11/2015 20:06:02	Payload 0x91	Seabird GPCTD	32	38	CB8156017F04A3FBE3E97955E403000...
06/11/2015 20:04:44	Payload 0x91	Seabird GPCTD	32	38	468156013204A3FB93E97955E4030000...

©2016. Company Confidential.

Messaging Design

- Payload Protocol was designed to work with few CPU resources
  - > No operating system, single-threaded
  - > Speak only when spoken to
  - > Polling latency varies with host CPU load
- The first 16-bits of each Ack is the originating message type
- The high order bit (0x8000) indicates whether there are any messages waiting to be picked up

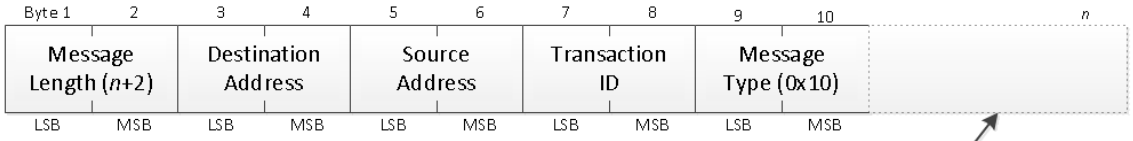
©2016. Company Confidential.

Enumeration

- The enumeration sequence is meant to discover which board ID's are present on the payload port and need to be polled
- The response describes the payload and what its needs are
- The message supports daisy-chained communications (designed for PIB's)
- Enumeration happens in a certain sequence and timing, and each payload must respond. Any that do not respond in the allowed time will never be polled, and they will never be able to receive or send messages.

©2016. Company Confidential.

REQUEST_ID 0x10 (Enumeration Request)




```
typedef struct _LRI_REQUEST_ID_
{
    uint16_t    communicationProtocols;
    uint16_t    commandFormats;
} LRI_REQUEST_ID, *PLRI_REQUEST_ID;
```

ACK (1) with Response to Enumeration

High order bit:
1 = pending message waiting
0 = no messages waiting

```
typedef struct _LRI_ID_PACKET_
{
    uint16_t    version;
    uint16_t    deviceType;
    uint16_t    address;
    char        serialNum[6];
    uint8_t     port;
    uint32_t    pollingRate;
    uint8_t     extraInfo;  //Bit 0 = telemetry, Bit 1 = power, Bit 2 = events, Bit 3 = Echoed ACKS
    uint8_t     firmwareMajor;
    uint8_t     firmwareMinor;
    uint16_t    firmwareRev;
    char        description[20];
}LRI_ID_PACKET, *PLRI_ID_PACKET;
```


©2016. Company Confidential.

Payload Shutdown

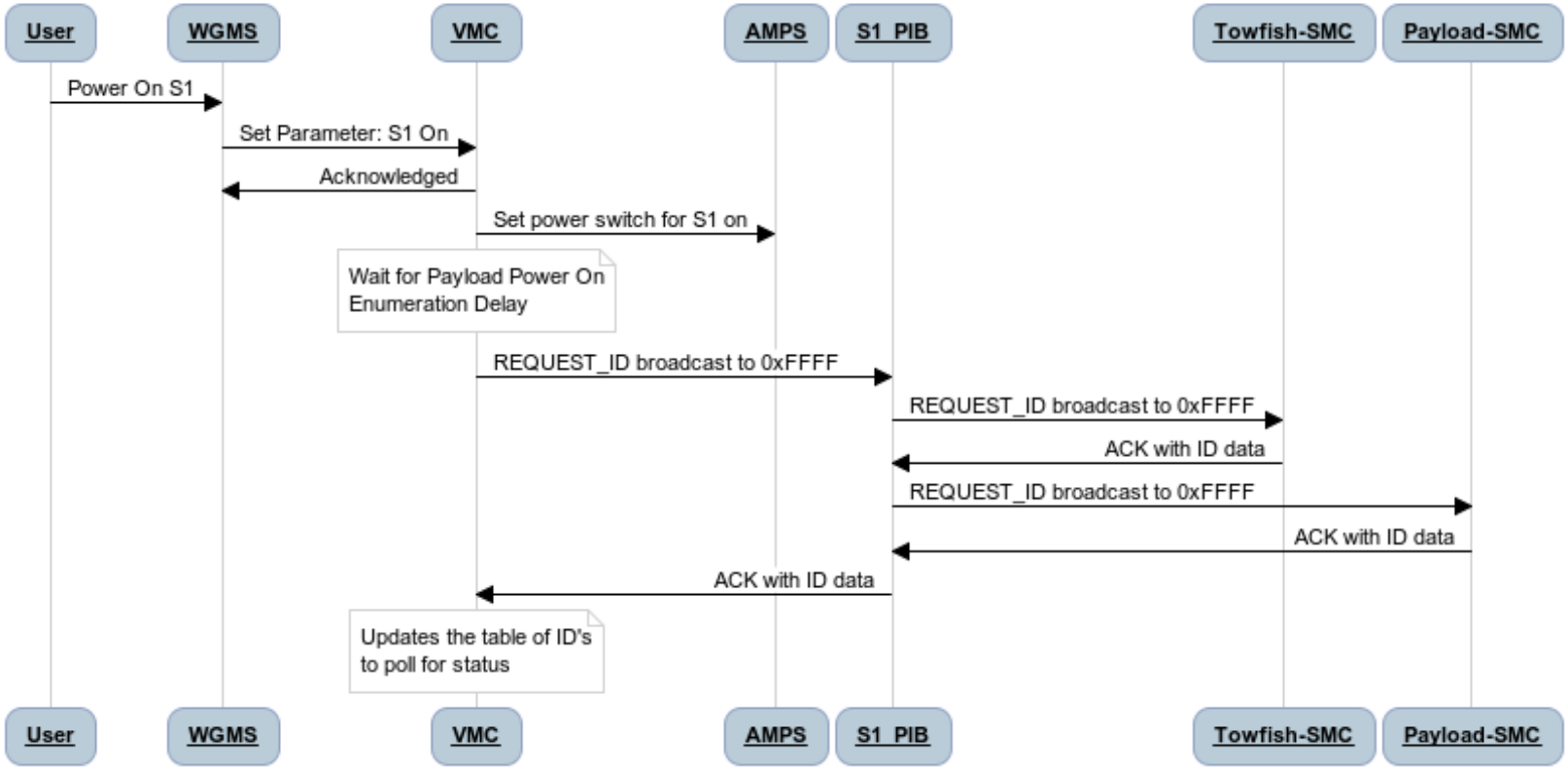
Anything that requests the payload to turn off triggers the shutdown sequence

- Set Parameter message (from WGMS, a WGMS client or another payload)
- Request from the SV3 web page
- Request from the XBee Relay console (SV2 only)

The protocol allows payloads to request a delay before the power is cut (allows for graceful shutdown)

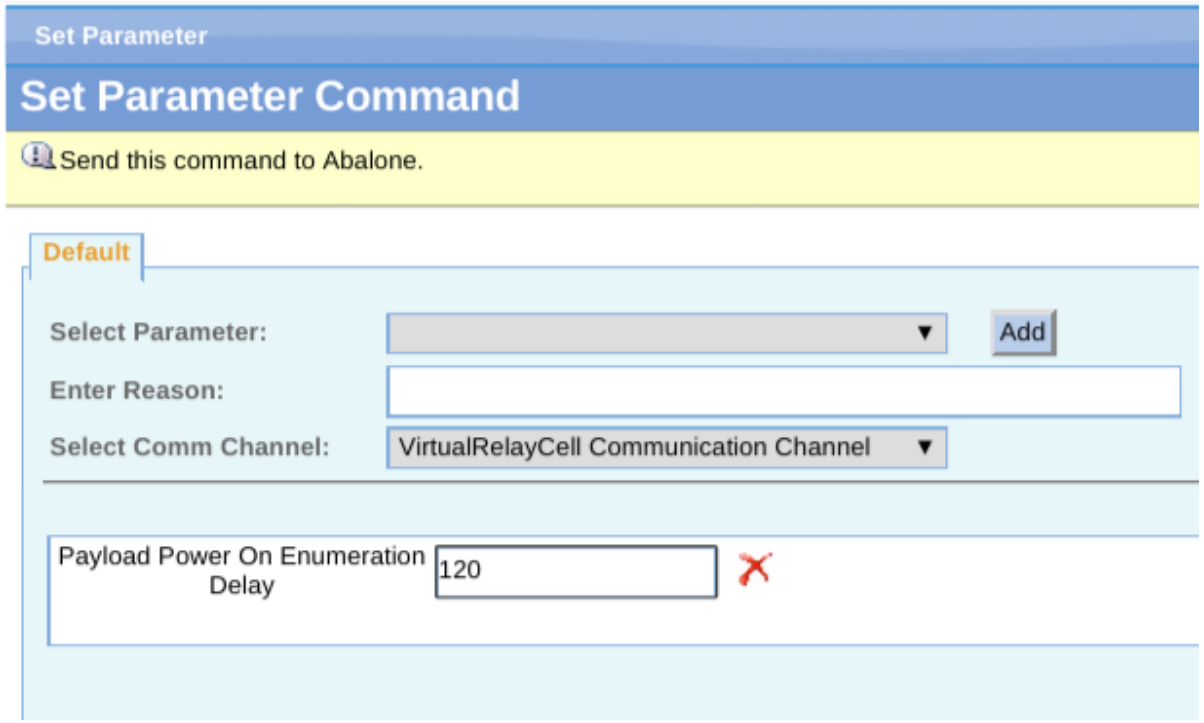
©2016. Company Confidential.

Payload Protocol: Power On and Enumeration



©2016. Company Confidential.

Payload Power On Enumeration Delay



©2016. Company Confidential.

Payload Shutdown

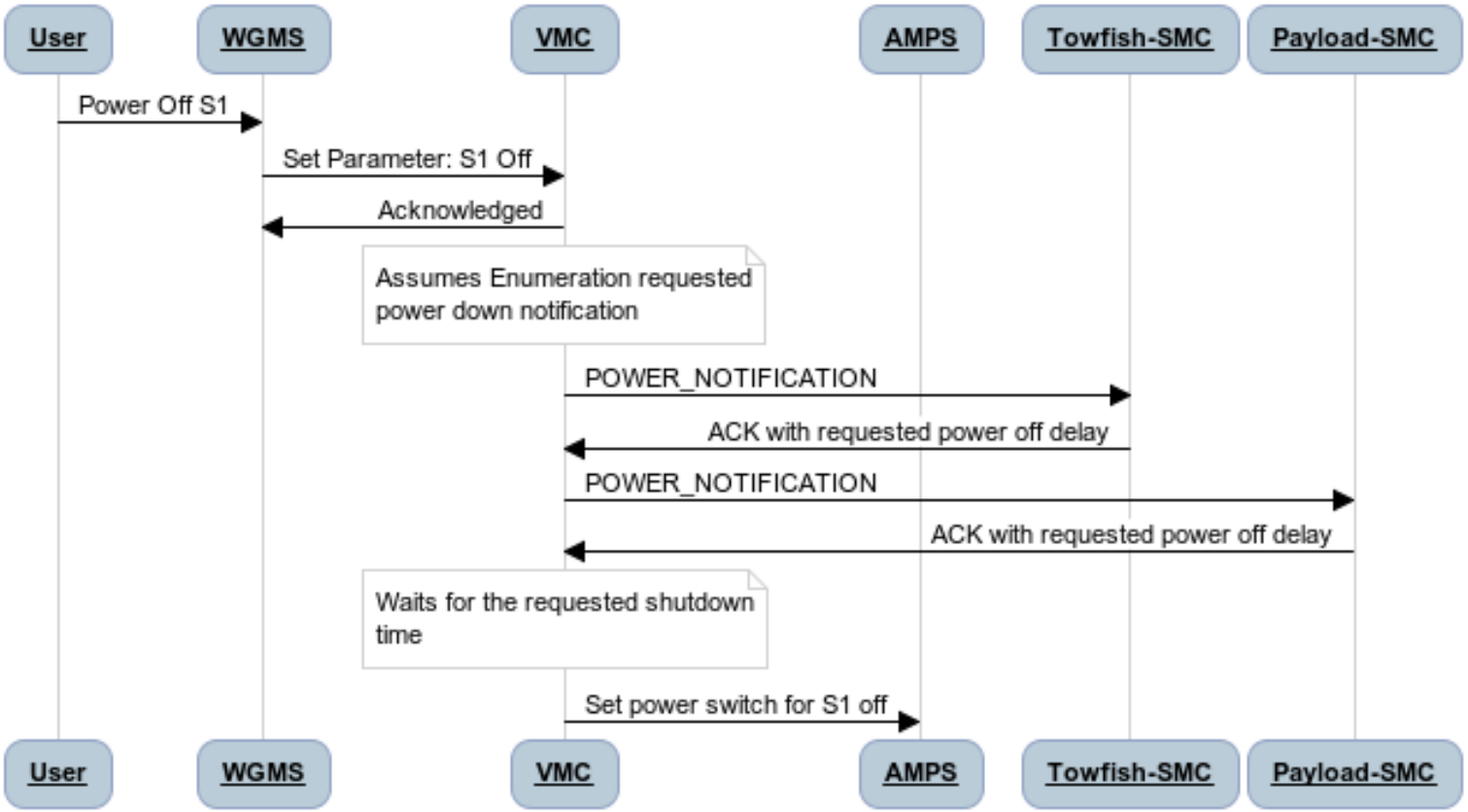
Anything that requests the payload to turn off triggers the shutdown sequence:

- Set Parameter message (from WGMS, a WGMS client or another payload)
- Request from the SV3 web page
- Request from the XBee Relay console (SV2 only)

The protocol allows payloads to request a delay before the power is cut (allows for graceful shutdown)

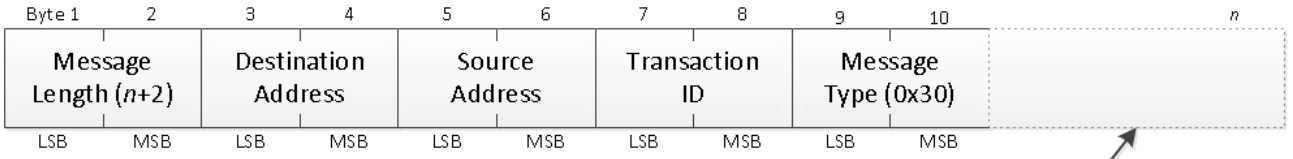
©2016. Company Confidential.

Payload Protocol: Power Down



©2016. Company Confidential.

POWER_NOTIFICATION (0x30)




```
typedef struct _PYLD_POWER_NOTIFICATION_
{
    uint8_t     notificationType;
    uint8_t     boardID;
    uint8_t     newState;
}PYLD_POWER_NOTIFICATION, *PPYLD_POWER_NOTIFICATION;
```

ACK (1) with Response to Power Notification

High order bit:
1 = pending message waiting
0 = no messages waiting

```
typedef struct _ACK_PYLD_POWER_NOTIFICATION_RECORD_
{
    uint16_t    address;
    uint8_t     boardID;
    uint8_t     state;
    uint8_t     delayOff;
    uint8_t     delayOn;
}ACK_PYLD_POWER_NOTIFICATION_RECORD, *PACK_PYLD_POWER_NOTIFICATION_RECORD;
```

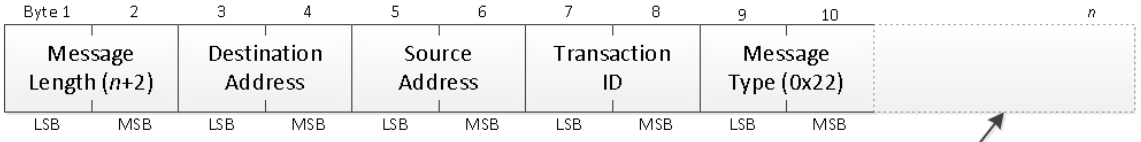

©2016. Company Confidential.

Polling for Status, Handling Waiting Messages

- Payload devices do not asynchronously originate messages, so polling is required
- ACK messages can indicate whether there are any messages waiting in the transmit queue, but only de-queue and send the messages when REQUEST_WAITING_MESSAGES is sent
- Note that the polling message includes the latest time and location, but the time update is not very accurate. If it is possible, use the GPS receiver output routed through the cable to the payload.

©2016. Company Confidential.

REQUEST_PIB_STATUS (0x22)




```
typedef struct _LRI_REQUEST_PIB_STATUS_
{
    TIMESTAMP   timestamp;
    int32_t latitude;
    int32_t longitude;
    char        fixValid;
} LRI_REQUEST_PIB_STATUS, *PLRI_REQUEST_PIB_STATUS;
```

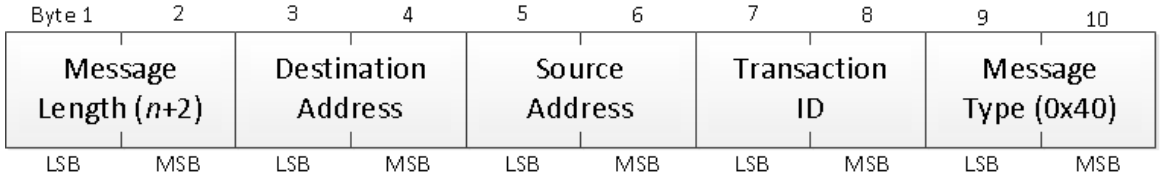
ACK (1) with Response to Request PIB Status

High order bit:
1 = pending message waiting
0 = no messages waiting

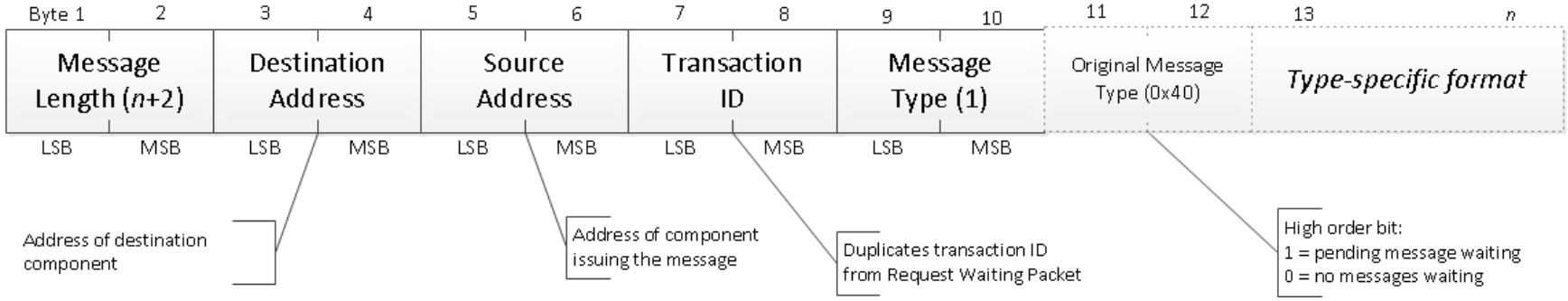
```
typedef struct _LRI_PIB_STATUS_
{
    uint16_t    version;    //Hi bit = Comand/Data Waiting
    uint16_t    address;    //PIB address
    uint16_t    alarms;
    uint8_t     leak1;      //Leak Sensor 1 status
    uint8_t     leak2;      //Leak Sensor 2 status
    int8_t      humidTemp;  //Temperature Form the Humidity Sensor (deg C)
    uint8_t     humidity;   //Relative Humidity (%)
    int8_t      presTemp;   //Temperature From the Pressure Sensor (deg C)
    uint8_t     pressure;   //Pressure (kPa)
}LRI_PIB_STATUS, *PLRI_PIB_STATUS;
```


©2016. Company Confidential.

REQUEST_WAITING_PACKET (0x40)

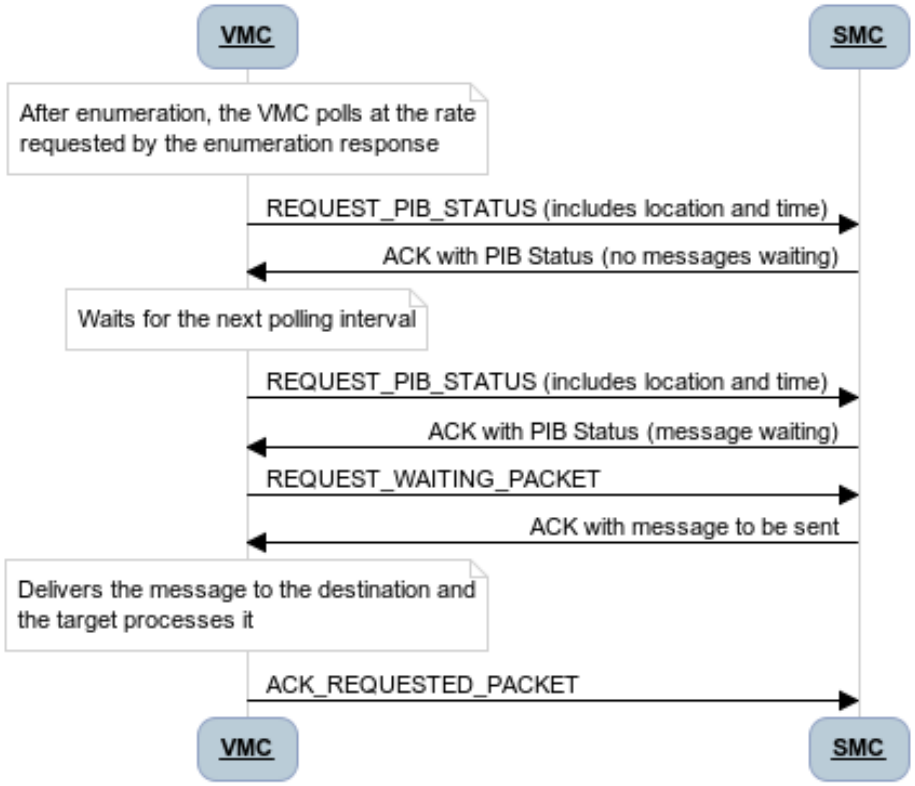


ACK (1) with Response to Request Waiting Packet



©2016. Company Confidential.

Payload Protocol: Polling for Status and Handling Waiting Messages



©2016. Company Confidential.

Float C&C Target Addresses

- Command Execution (0x0001)

  Messages sent to address #1 are expected to be commands in S2S Level 2 format that the VMC will execute. In this way, the payload can issue navigation and configuration commands.

- Telemetry (0x0002)

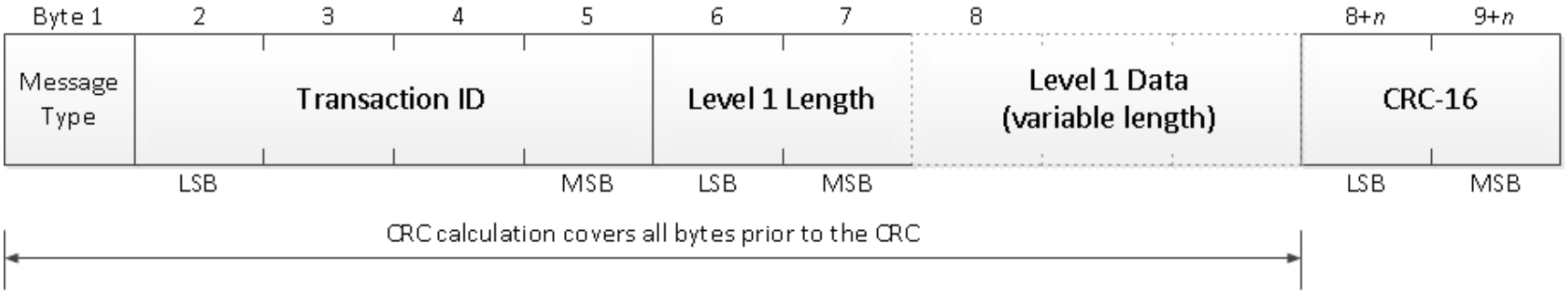
  Messages sent to address #2 are expected to be in S2S Level 2 format, except the transaction ID is not sent (it is expected to be zero)

- Message Forwarding (0x0010)

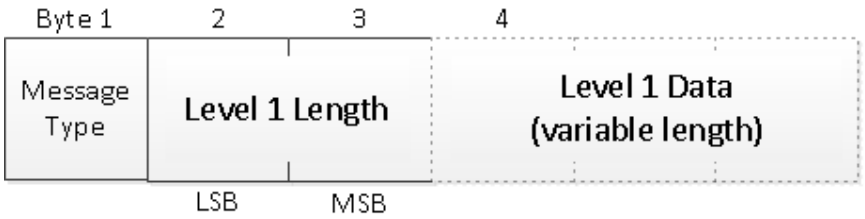
  Messages sent to address 0x0010 are targeted to some other address on the vehicle. Use this form when sending Write Console Ack messages or when targeting components in other payloads.

©2016. Company Confidential.

Level 2 S2S Message

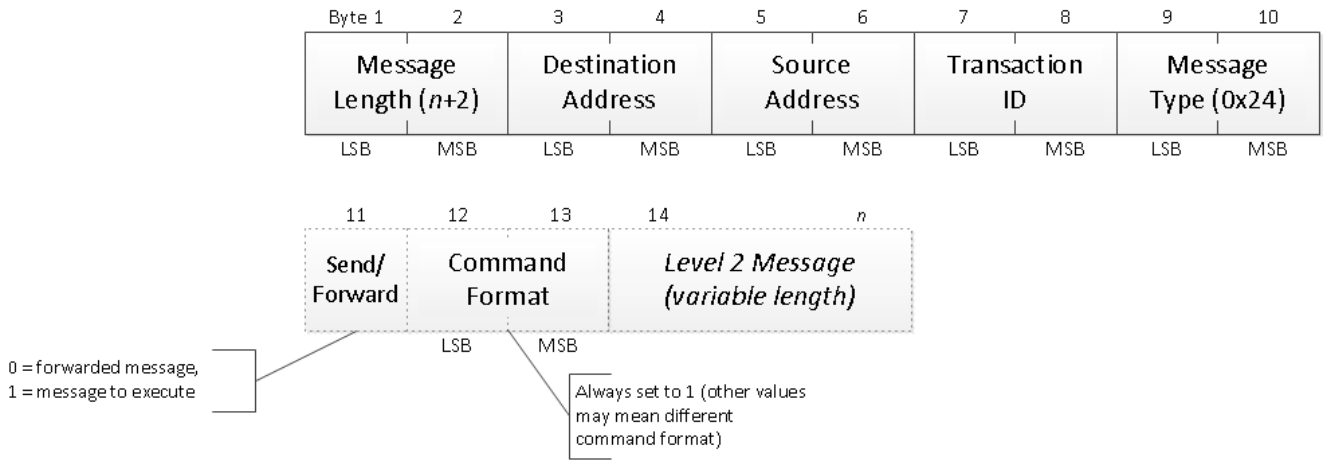


Omit the transaction ID field when sending to address 2, and there's no need to include the CRC since the payload message has a CRC:

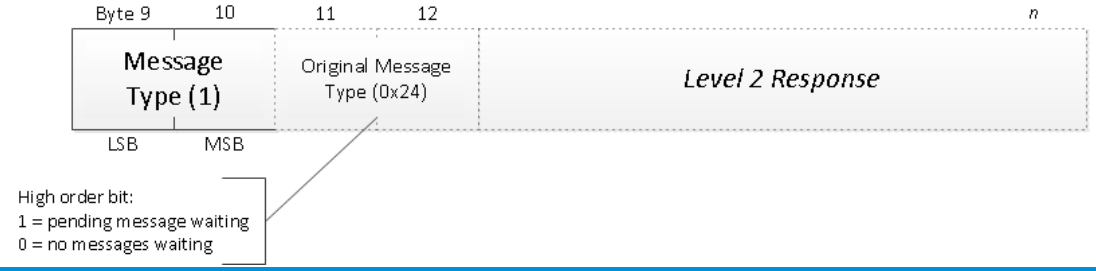


©2016. Company Confidential.

SEND_FWD_ICD_CMD (0x24)



ACK (1) with Response to Send/Forward ICD Command



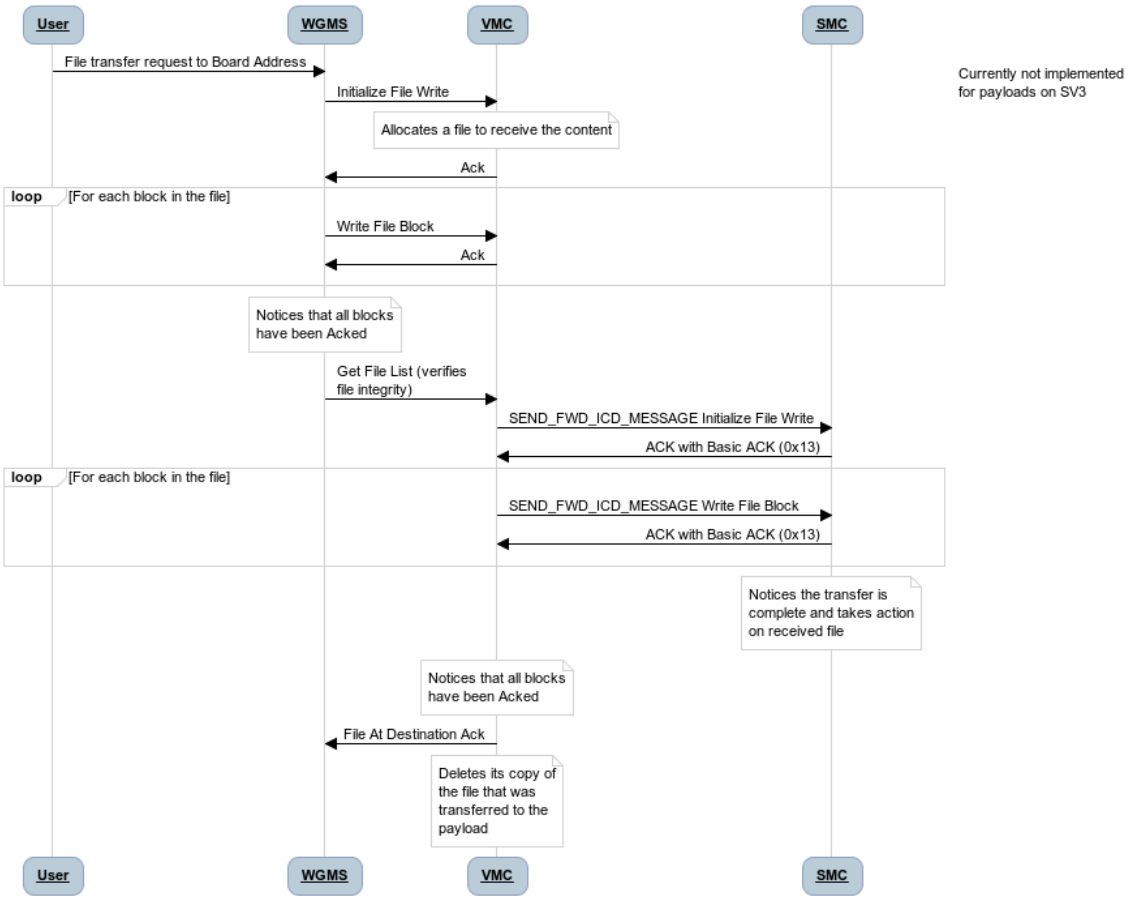
©2016. Company Confidential.

File Transfer To and From Payloads

- The file transfer messages implemented in the Ship to Shore (S2S) message set can be forwarded to a payload by specifying the Storage Controller address in the messages as Board Address
- Files sent to the payload can be installed as new software or data files or can carry data along with a command
- The payload can originate file transfers back to WGMS to return larger command responses or sensor data

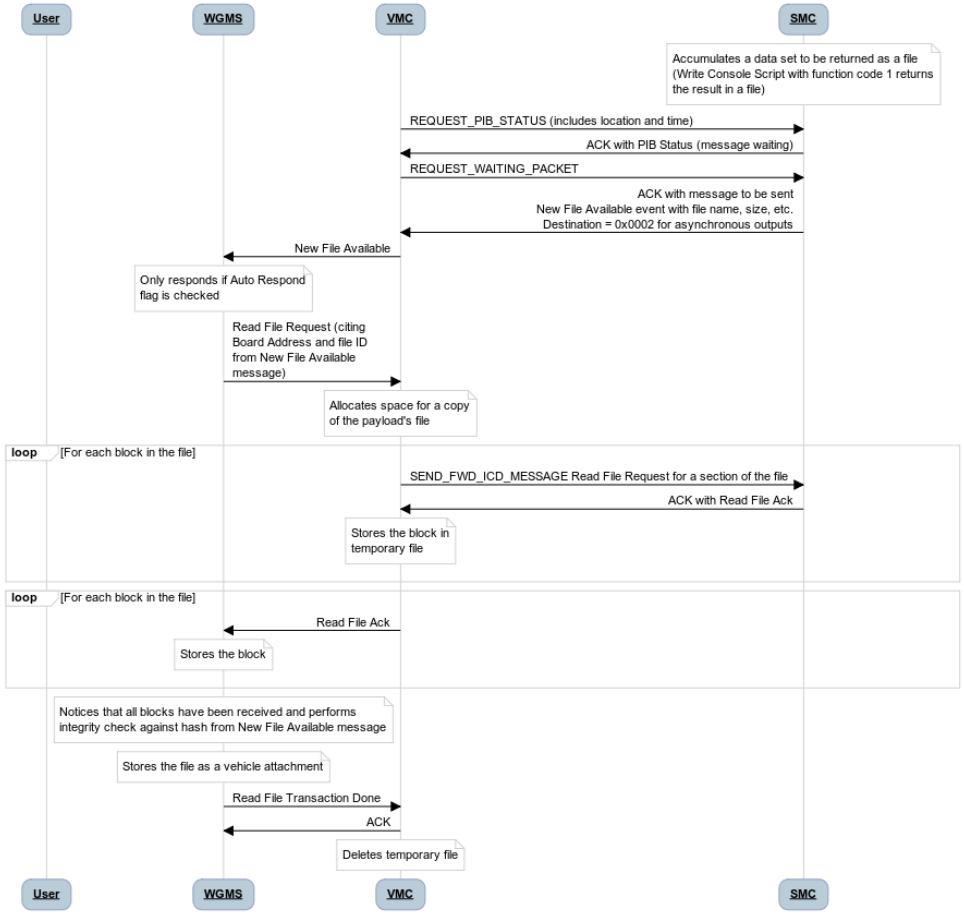
©2016. Company Confidential.

Payload Protocol: File Transfer to Payload



©2016. Company Confidential.

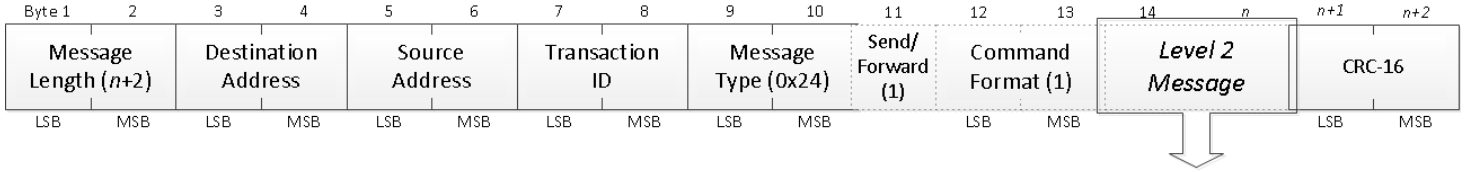
Payload Protocol: File Transfer from Payload



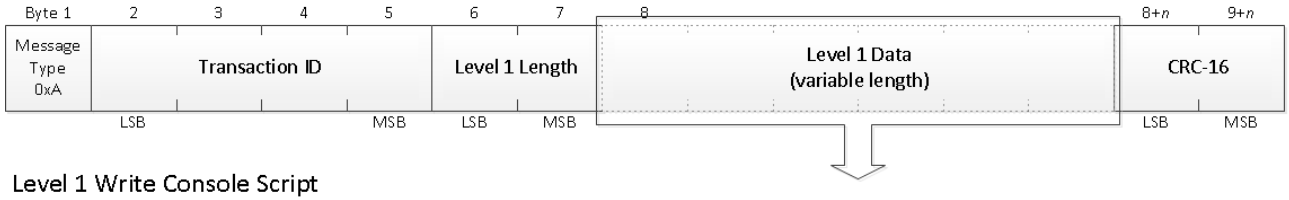
©2016. Company Confidential.

Responding to Write Console Script

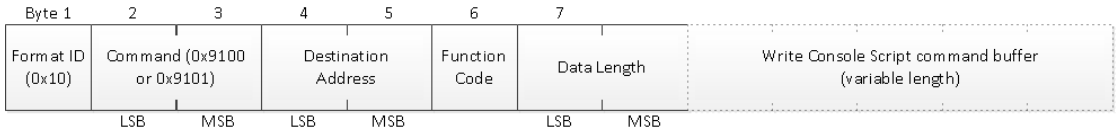
SEND_FWD_ICD_CMD (0x24)



Level 2 Command Message



Level 1 Write Console Script



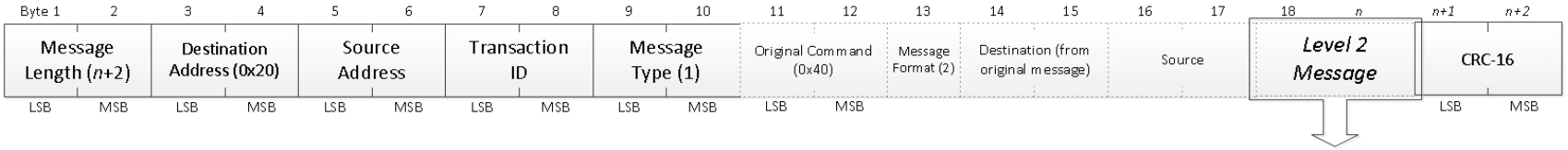

```
2D 00 00 20 0A 00 BF 01 24 00 01 01 00 0A 03 00 AA 7A 15 00 1E 00 91 00 20 00 0D 00 72 75 6E
73 68 65 6C 6C 20 64 61 74 65 AC 95 DD D5
```

From 0xA to 0x2000, Write Console Script to address 0x2000 function code 0 “runshell date”

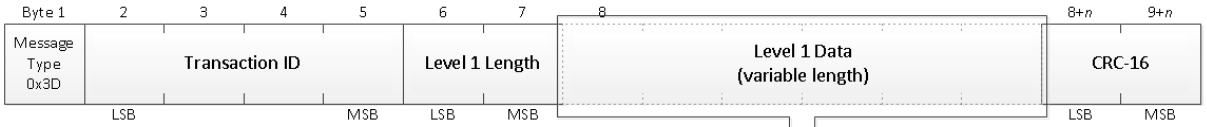
©2016. Company Confidential.

Responding to Write Console Script

ACK to REQUEST_WAITING_PACKET (0x40) following Write Console Script



Level 2 ACK Message



Level 1 Write Console ACK

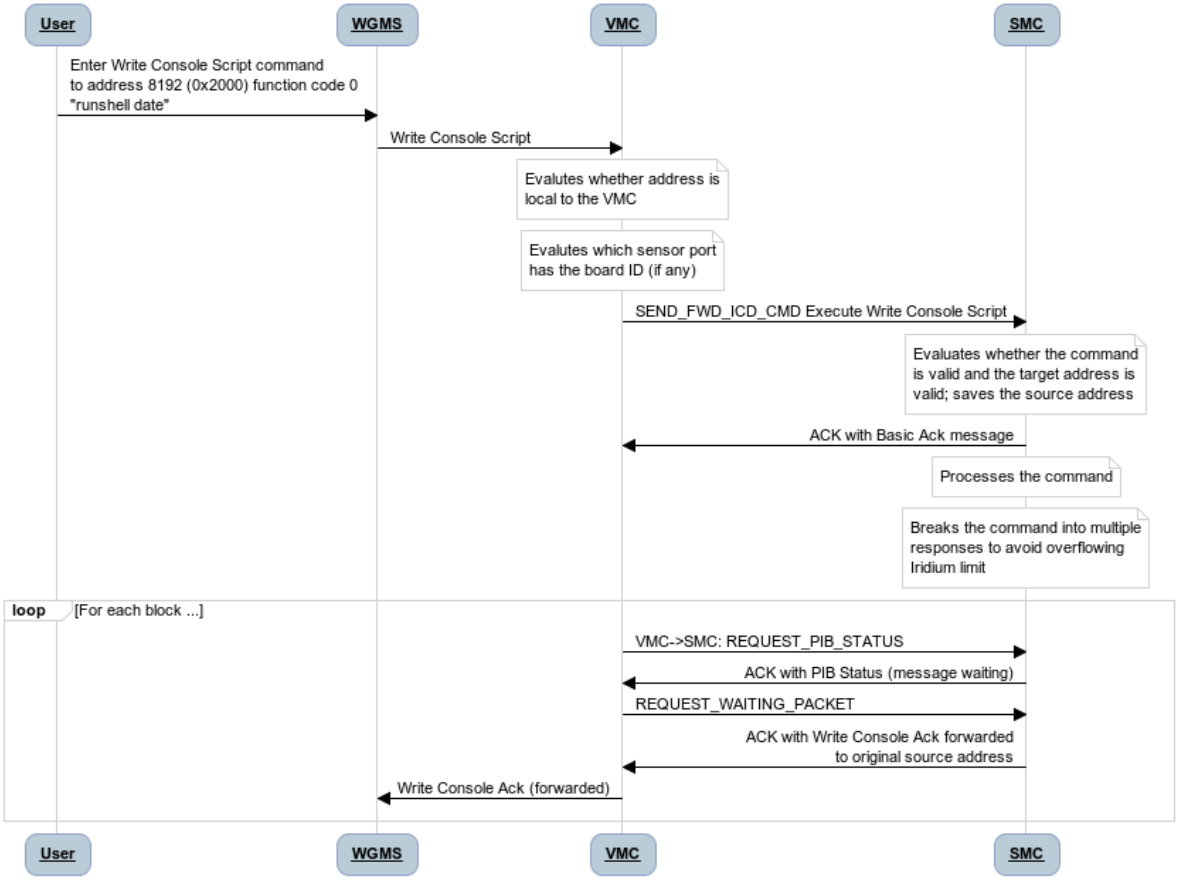
OK=1, 2=Truncated, 10=Unknown Command, 11=Busy, 12=Unsupported Command, 31=Unknown Destination (see ICD)

```
3A 00 20 00 00 20 C1 01 01 00 40 00 02 0A 00 00 20 3D 03 00 AA 7A 1E 00 00 01 1A 00 4D 6F 6E 20
4A 75 6C 20 20 36 20 32 32 3A 35 31 3A 30 39 20 20 32 30 31 35 0A 49 92 2F E0
```

From 0x2000 to 0x20, Write Console Ack from 0x2000 to 0xA (source of the original command) with disposition 1 (OK): "Mon Jul 6 22:51:09 2015"

©2016. Company Confidential.

Payload Protocol: Responding to Write Console Script



©2016. Company Confidential.

Telemetry Reporting

Telemetry is any unsolicited message from the Wave Glider. Two message types are reserved for use by integrators:

- Type 0x91 telemetry is used for telemetry that should be parsed by WGMS. It can contain multiple samples, each of which has a location and time.
- Type 0x94 telemetry is used for free-form reporting to an external client and is not parsed by WGMS.

The *payload type ID* is a 32-bit number that allows WGMS and other software to distinguish between types of reported data. **Integrators coordinate with Liquid Robotics for assignment of payload type ID's.**

The *board ID* and *task ID* in the message distinguish between multiple reports of the same type of data.

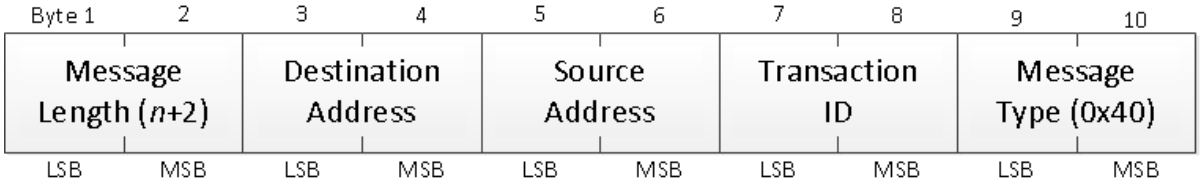
©2016. Company Confidential.

Telemetry Reporting

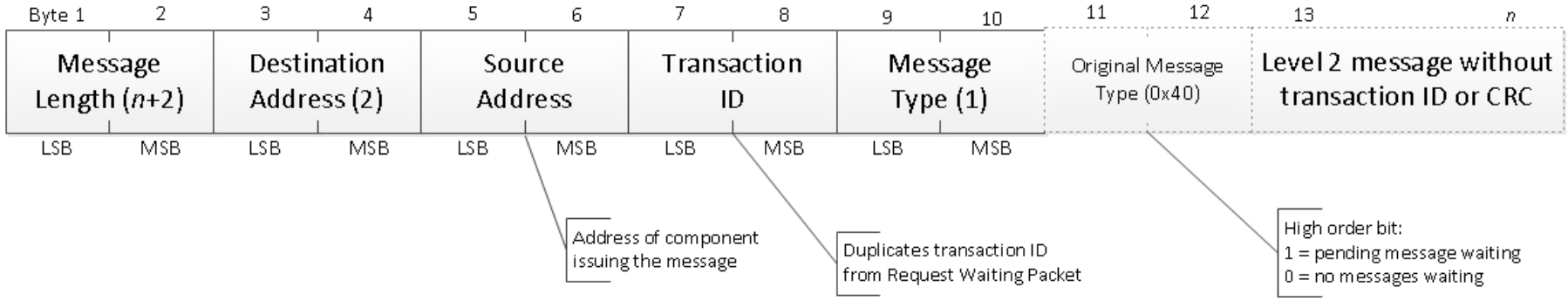
- The *data format* byte allows the same payload type ID to select any of 16 different data formats.
- The *latitude* and *longitude* fields in the type 0x91 telemetry message are sent as signed integers in real degrees multiplied by 600,000. This preserves the resolution of the number in a 4-byte footprint.
- The *timestamp* of each sample is reported as 32-bit Linux epoch time sometimes referred to as a *time_t* (the number of seconds since 1-January 1970, UTC)

©2016. Company Confidential.

REQUEST_WAITING_PACKET (0x40)



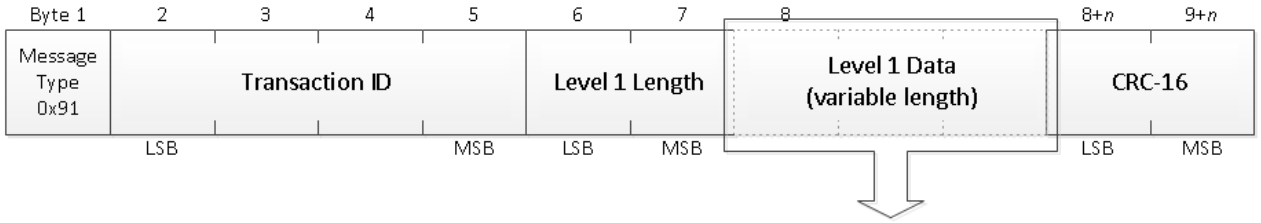
ACK (1) with Response to Report Telemetry



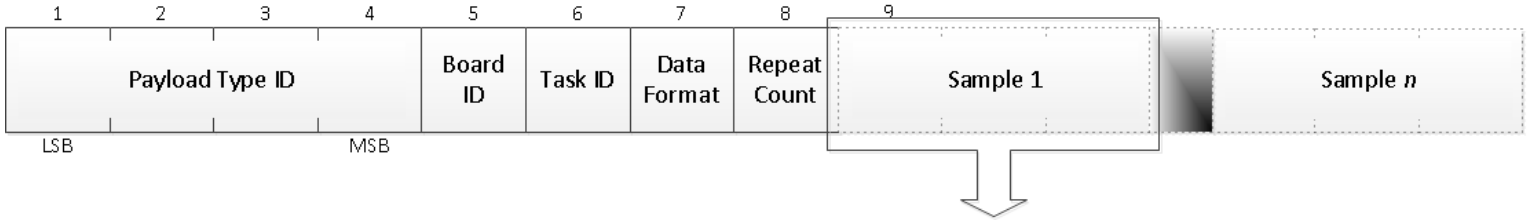
©2016. Company Confidential.

Type 0x91 Message Formatting

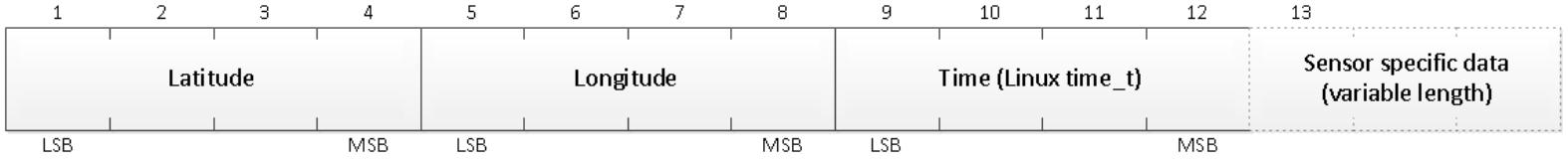
Message Type 0x91 Level 2



Message Type 0x91 Level 1



Message Type 0x91 Sample



©2016. Company Confidential.

Type 0x91 Assembly Walk-Thru

Liquid Robotics is at (37.410922,-122.004492), which is sent as (22446553,−73202695) or (0x15681D9, 0xFBA303F9)

2-July 2015 at 23:18:44 UTC is 1435879125 in Unix epoch time or 0x5595C6D5

Payload type 200 is reserved for reporting ASCII strings as telemetry: let's suppose we want to send "Hello, world!" from board ID 32 (typical for an SMC) and task ID 0, the message would be written as:

91 00 00 00 00 21 00 C8 00 00 00 20 00 00 01 D9 81 56 01 F9 03 A3 FB D5 C6 95 55 48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 21

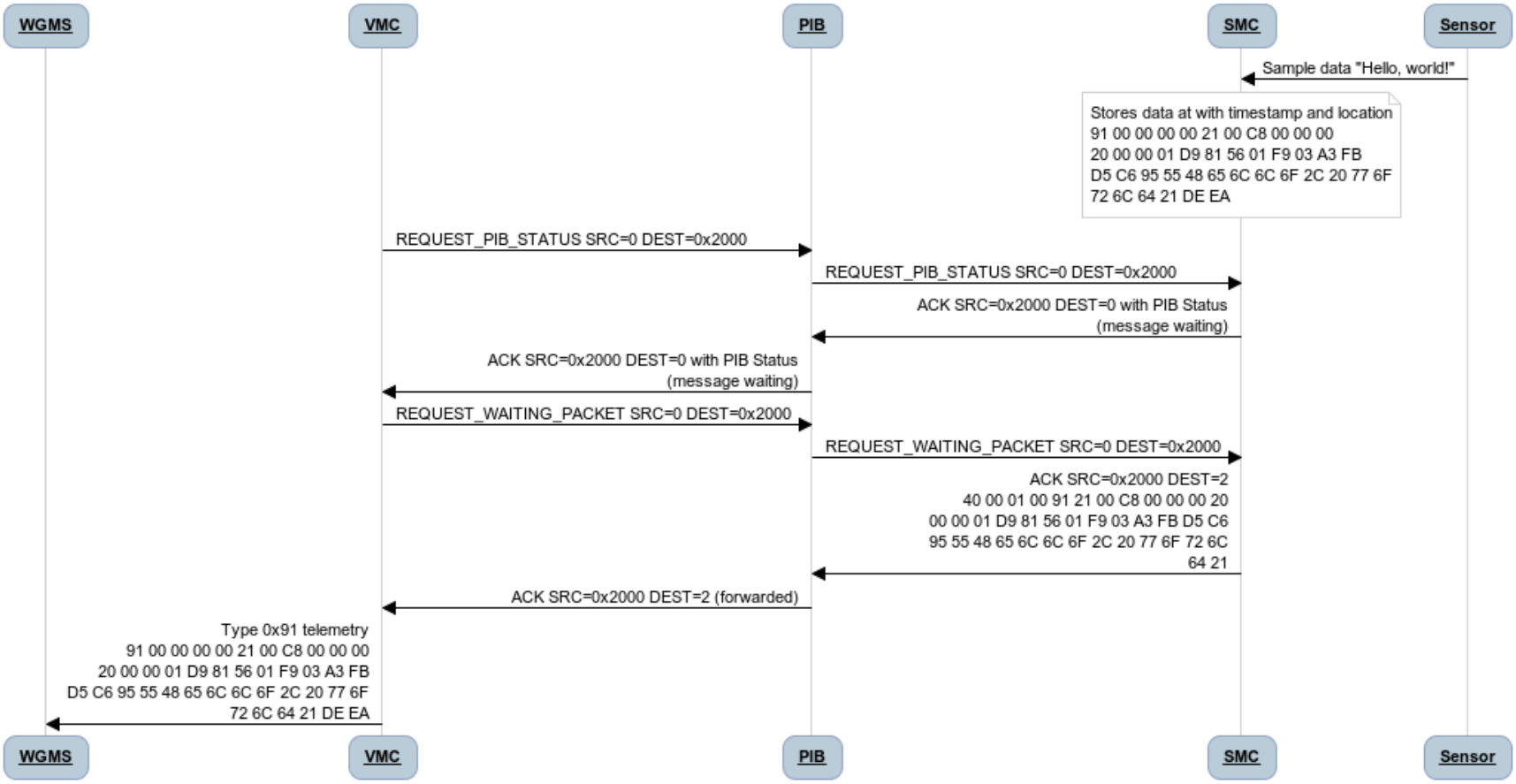
Running the CRC-16 algorithm results in a value of 0xEADE, so the full message is:

91 00 00 00 00 21 00 C8 00 00 00 20 00 00 01 D9 81 56 01 F9 03 A3 FB D5 C6 95 55 48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 21 DE EA

I have colored the Level 2 wrapper in blue, the type 0x91 fixed header in green, the per-sample head in dark orange and the sample data in red.

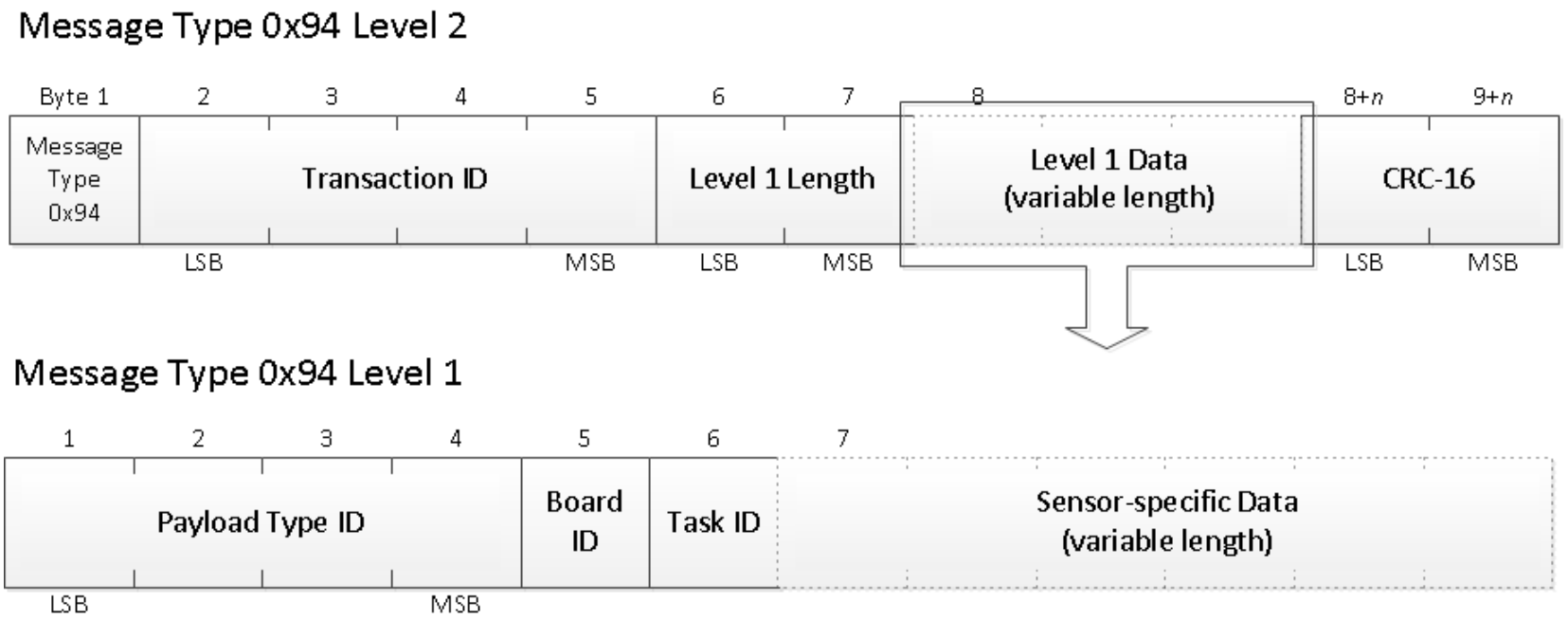
©2016. Company Confidential.

Payload Protocol: Reporting Telemetry Type 0x91



©2016. Company Confidential.

Type 0x94 Message Formatting



©2016. Company Confidential.

Type 0x94 Assembly Walk-Thru

Suppose a customer is assigned payload type 1035 (0x40B) and wants to send only the current time (2-July 2015 at 23:18:44 UTC is 1435879125 in Unix epoch time or 0x5595C6D5) from board ID 32 and task ID 0, the message would be written as:

94 00 00 00 00 0A 00 0B 04 00 00 20 00 D5 C6 95 55

Running the CRC-16 algorithm results in a value of 0xEF05, so the full message is:

94 00 00 00 00 0A 00 0B 04 00 00 20 00 D5 C6 95 55 05 EF

I have colored the Level 2 wrapper in blue, the type 0x94 fixed header in green and the sample data in red.

©2016. Company Confidential.

Reference Implementations

- The **Payload Console Simulator** is a program written in portable C that implements the Payload Protocol as a client (payload.) It is available as source.
- The **SMC software** is available as source and is portable to many Linux platforms. The **Payload Interface** server implements the Payload Protocol and interacts with programs that implement the appropriate CORBA interface. SMC software is available pre-built for the SMC platform or as source.

©2016. Company Confidential.

Key Take-Aways

- Payload Protocol is a full-featured mechanism for:
  - > Registering a payload with the navigation computer
  - > Orchestrating safe payload shutdown
  - > Collecting commands and sending back responses
  - > Transferring files
  - > Reporting telemetry to WGMS
  - > Controlling the navigation computer configuration and execution
  - > Sending messages between modules
- Payload Protocol is custom to Liquid Robotics
- Payload Protocol was designed for SV2
- Payload Protocol was designed for small CPU's and lower power consumption

©2016. Company Confidential.

Recommended Reading

Wave Glider® SV3 Interface Control Description, available from Liquid Robotics.

©2016. Company Confidential.