LoRaWAN payload decompression - Uplink and Downlink

ASYSTOM

Abstract

The purpose of this document is to provide detailed information about the LoRaWAN payload content received from/sent to Asystom Beacons, so that data decompression can be done at LoRa level if required.

This document was written starting with beacon versions 4.23. For more recent versions, any divergences with previous versions will be clearly mentioned in their respective chapters.

It is recommended to review this document together with AsystomView User Guide in order to better understand the purpose of certain values or names.

Contents

Changelog

Version number	Written by	Revision date	Note
1.0	Piotr Diop	16/07/2021	Initial draft
1.1	Piotr Diop	20/07/2021	Added more details to the abstract
1.2	Piotr Diop	16/02/2022	Corrected conversion typo in synchronization settings
1.3	Piotr Diop	08/09/2022	Updated Downlink FPorts for settings
1.4	S Lhuisset	28/09/2022	Updated with FFT Zoom payload
1.5	Piotr Diop	15/12/2022	Added chapter on uplink packet split mechanism

1 LORAWAN FRAME DESCRIPTION

A data frame is a composition of at least one of these two types of elements:

- One-dimensional scalar elements (e.g Battery level)
- Multidimensional vectored elements (e.g Signature)

There can be only one vector type at most inside each frame. It will always be the last element of the frame.

In order to evaluate the amount of information present in an uplink packet, we use the frame port (FPort). The way it is used will be detailed in relevant chapters.

For downlink messages, we use the FPort to encode different request types

The following table summarizes the port usage for different message types

For beacon version up to 4.40:

FPort	Uplink	Downlink
1-63	Reserved for uplink	N/A
64	N/A	Request the device to Reboot
67	System status report	N/A
70	N/A	Update scheduling settings
71	N/A	Update advanced settings
74	N/A	Update signature settings

For beacon version strictly greater than 4.40:

FPort	Uplink	Downlink
1-63	Reserved for uplink	N/A
64	N/A	Request the device to Reboot
70	N/A	Update scheduling settings
71	N/A	Update advanced settings
74	N/A	Update signature settings

In those versions, the system status report is processed like any other uplink frame.

2 UPLINK PACKET FORMAT

2.1 Standard uplink packet (FPort 1-63)

The general structure of an uplink packet is as follows:

```
<elt_1 id><elt_1 val><elt_2 id><elt_2 val>.....<elt_n id><elt_n val_0><elt_n val_1>...<elt_n val_p>
```

> ***NOTE:*** In the first chapter we specified that there can be at most one vector in a packet. This means that only the final element of a packet can be an array.

The following table enumerates the Id and the expected size for each publicly available element:

Element	Id (in hexadecimal format)	Type	Size of the value
Battery Level	00	Scalar	2 Bytes (4 chars)
Humidity	02	Scalar	2 Bytes (4 chars)
Signature (standard)	0b	Vector	98 Bytes (196 chars)
Signature (FFT Zoom)	0d	Vector	Settings-dependent
Temperature	0e	Scalar	2 Bytes (4 chars)

All `<elt_X id>` have a fixed 1 byte size

For any scalar element, the value `<elt_X val>` is fixed to 2 bytes. Values are always encoded in 16-bit Little Endian.

The FPort encodes the number of elements present in an uplink packet.

Unless specified otherwise, all values are 16-bit Little Endian.

2.1.1 Uplink packet with only scalar data

When an uplink packet is composed exclusively of scalar elements, the packet size is exactly 3 times (1-Byte id + 2-Byte value) the value of FPort.

For a packet containing only scalar elements, Packet size = FPort * 3

2.1.2 Uplink packet terminated by a vector

On the other hand, when the above check fails, there is a vector for which the Id is located at the offset equal to 3x(FPort-1).

For a packet terminated by a vector:

Packet size != FPort * 3

The number of scalar elements is FPort-1

The last element is the vector

The vector data starts at position: PacketSize - 3*(FPort-1)

This is also true when there is no scalar data inside the uplink packet.

2.1.3 Summary

A purely scalar packet containing Battery, Humidity and Temperature will look as follows (values set to 0xFFFF for the example)

```
[
    {
    "FPort": 03,
    "payload": "00FFFF02FFFF0EFFFF"
    }
]
```

A packet containing only a Signature (vector) will present itself as a byte array of 99 Bytes (1-Byte Id + 98 Bytes array) with the following format:

```
[
    {
    "FPort": 01,
    "payload": "0bFFFFFFFFFFFF......FF"
    }
]
```

A packet containing Battery level, Temperature, Humidity and a Signature at the end will have a size of 108 Bytes and have the following format:

```
[
    {
    "FPort": 04,
    "payload": "00FFFF02FFFF0EFFFF0bFFFFFFFFFFFF......FF"
    }
]
```

3 UPLINK PACKET CONTENT

3.1 Scalar content

Due to the restrictions regarding the value size, a simple rescaling method is used to properly reflect the actual value. The rescaling equation is

```
RealValue = (Value_read_as_uint16LE * (max_range - min_range) / scaling) + min_range
```

The values are 16-bit Little Endian, the scaling is always 65535. The following table defines the range values.

Element	Id (in hexadecimal format)	min value	max value
Battery Level	00	0	100
Humidity	02	0	100
Temperature	0e	-273.15	2000

3.1.1 Example

We receive a packet on FPort 3 with payload 0023080e2a2102cc01.

- Battery level is 0x0823 * 100 / 65535 = 3.17V
- Temperature is (0x212a * (2000 + 273.15) / 65535) - 273.15 = 21.33°C
- Humidity is 0x01cc * 100 / 65535 = 0.7%

3.2 Vector content (Byte Arrays)

Vectors are arrays of 8-bit or 16-bit values encoded in Little Endian.

3.2.1 Signature (standard)

This is the default signature encompassing various measurements and statistics about the vibration and audio signal.

"Signature (standard)" contain 49 subfields, totaling 98 Bytes (196 chars). All values are 16-bit values encoded in Little Endian. The ordering is as follows:

```
s_00, s_01, s_02, s_03, s_04, s_05, s_06, s_07, s_08, s_09, s_10,
s_11, s_12, s_13, s_14, s_15, s_16, s_17, s_18, s_19, vib_x_acc,
vib_x_vel, vib_x_peak, vib_x_kurt, vib_x_root, vib_x_f1, vib_x_f2, vib_x_f3,
vib_y_acc, vib_y_vel, vib_y_peak, vib_y_kurt, vib_y_root, vib_y_f1, vib_y_f2,
vib_y_f3, vib_z_acc, vib_z_vel, vib_z_peak, vib_z_kurt, vib_z_root, vib_z_f1,
vib_z_f2, vib_z_f3, temp, rfu, rfu, sonic_rmslog, rfu
```

Vector elements are also subject to the same rescaling equation. `<scaling>` is currently 65535

```
RealValue = (Value_read_as_uint16LE * (max_range - min_range) / scaling) + min_range
```

The following table defines the unit, ranges and physical signifiance for each subfield

Subfield	Unit	min value	max value	Detail
s_00..s_09*	dB	-150	0	10 vibration frequency bands s_00 to s_09, linearly spaced between 0 and 2kHz, 0dB represents the sensor full scale equivalent to 16g. Bands are defined as follows: {[0-200], [200-400], [400-600], [600-800], [800-1k], [1k-1.2k], [1.2k-1.4k], [1.4k-1.6k], [1.6k-1.8k], [1.8k-2k]} (Hz)
s_10..s_19*	dB	-150	0	10 sound frequency bands s_10 to s_19, linearly spaced between 0 and 80kHz. 0dB represents the sensor full scale equivalent to 120dB SPL. Bands are defined as follows: {[0-8k], [8k-16k], [16k-24k], [24k-32k], [32k-40k], [40k-48k], [48k-56k], [56k-64k], [64k-72k], [72k-80k]} (Hz)
vib_x_acc	g	0	16	X-Axis: RMS acceleration value
vib_x_vel	mm/s	0	100	X-Axis: RMS level of vibration velocity
vib_x_peak	g	0	16	X-Axis: absolute peak acceleration level
vib_x_kurt	-	0	100	X-Axis: kurtosis (reflects the shape of the acceleration amplitude probability distribution)
vib_x_root	RPM	0	30000	X-Axis: Root frequency(expressed in RPM, typically the speed of a rotating machine)
vib_x_f1	mm/s	0	100	X-Axis: RMS velocity (around the root frequency)
vib_x_f2	mm/s	0	100	X-Axis: RMS velocity (around the 1st harmonic)
vib_x_f3	mm/s	0	100	X-Axis: RMS velocity (around the 2nd harmonic)
vib_y_[..]	-	see x axis	see x axis	Y-Axis follows the same description as X-axis for the above parameters
vib_z_[..]	-	see x axis	see x axis	Z-Axis follows the same description as X-axis for the above parameters
temp	°C	-273.15	2000	Surface temperature as measured at the accelerometer level
rfu	-	-	-	Reserved For Future Use
rfu	-	-	-	Reserved For Future Use
sonic_rmslog	dB	-150	0	RMS value for ultrasonic data in the ultrasonic band [25kHz-80kHz]
rfu	-	0	65535	Reserved For Future Use

3.2.2 Signature (FFT Zoom)

This is an alternate signature as it comes in lieu of the standard signature when measurements get triggered. See settings below for more information about its activation schemes.

When a FFT zoom gets produced the payload is as follows:

"Signature (FFT Zoom)" contains a variable number of bands defined by the compression parameter used (see settings section below). Values are either 8-bit or 16-bit values encoded in Little Endian as defined by the compression parameter used. There is an additional 1-Byte `<handle>` appended at the end of the frame (see settings section below to understand the management of the handle parameter). The ordering is as follows,

```
<band_1 val><band_2 val><band_3 val>.....<band_n val><handle>
```

Values are dBFS values and are also subject to the rescaling equation.

```
RealValue = (Value_read_as_uint16LE * (max_range - min_range) / scaling) + min_range
```

where:

`<scaling>` is either 255 (8-bit compression) or 65535 (16-bit compression) depending on the compression parameter used (see settings section below). `<max_range>` is 0 (dBFS) `<min_range>` is -150 (dBFS)

Note that for the accelerometer 0 dBFS corresponds to 16 g for acceleration and envelope spectrums and 100 mm/s for velocity spectrums.

3.3 Special (non-periodic) content

3.3.1 System status report

System status report is sent each time a device joins a network.

For legacy device versions up to 4.40, it uses FPort 67, has a size of 84 Bytes, and uses the id 0xFF. For device version strictly greater than 4.40, it behaves as other uplink messsages, FPort contining the value 1, has a size of 84 Bytes, and uses the id 0xFF.

The report is an array of bytes composed of six elements: the status id, one RFU field, device version, scheduling settings, advanced settings and signature settings. The format is as follows:

```
<sys_status id><RFU val><Version val><sched_settings val><adv_settings val><sig_settings val>
     0xFF       4 bytes   5 bytes         10 bytes            38 bytes          26 bytes
```

Decoding the content of each field will be done in the 'Examples' chapter

4 UPLINK EXAMPLES

4.1 System status report

Lets take the following of a payload received on port 67 (we split the payload into chunks for easier visualization in the next parts:

```
FF 00000082 76342E3336 054114001E001E006801
0F000000401F0000D00700001C0016000500320000000000B3009101000000002B000700100E
000001000300D00700000000000002000000000000000000000000
```

This frame totals 168 characters i.e 84 Bytes.

4.1.1 System status id (1 byte)

Is and will aways be 1 byte: 0xFF

4.1.2 RFU1 (4 bytes)

Its typical value is “00000082”. The content is irrelevant to the purpose of this document.

4.1.3 Beacon version (5 bytes)

Can be directly translated into ascii. For example 76342e3336 -> v4.36

4.1.4 Scheduling Settings (10 bytes)

For devices using version 4.36, the default value is 054114001e001e006801

This settings array encodes two accessible public parameters. Each mush be read as 16 bit little endian.

- General stack periodicity. This setting tells how often the system will evaluate features related to the general stack. Currently the general stack covers Battery level, Temperature and Humidity sensors. The value is expressed in 10*seconds. This means a value of 30 must be interpreted as 300s i.e 5 minutes.
- Predictive stack periodicity. This setting tells how often the system will schedule vibration & ultrasonic sensors to do predictive maintenance computations, i.e. to generate either a standard signature or a FFT zoom based on settings. The value is also expressed in 10*seconds.

Message	Size	Default value (hex LE)	Decimal value	Real value
RFU	4 Bytes	05411400		
General stack periodicity	2 Bytes	1e00	30	300s
Predictive stack periodicity	2 Bytes	1e00	30	300s
RFU	2 Bytes	6801		

4.1.5 Advanced Settings (38 Bytes)

In the example, the value is

0f000000401f0000d00700001c00160005003200000000000b3009101000000002b000700100e

Message	Size	Default value (hex LE)	Decimal value	Scaling	Real value
RFU	12 Bytes	0f000000401f0000d0070000	-	-	-
RPM Max	2 Bytes	1c00	28	x60	1680 RPM
RPM Min	2 Bytes	1600	22	x60	1320 RPM
RFU	8 Bytes	0500320000000000	-	-	-
Synchronization settings	4 Bytes	b3009101	see next chapter	-	-
Synchronization delay	2 Bytes	0000	see next chapter	-	-
Synchronization timeout	2 Bytes	0000	see next chapter	-	-

4.1.6 Synchronization settings & modes

There are 5 modes a Beacon can be configured into. Each one of them uses a specific synchronization method.

The modes are Inactive, Wake-on-Motion, Wake-on-Scheduler, Wake-on-Analog and Wake-on-Contact.

The synchronization settings block is a 32-bit Little Endian field composed of three publicly available elements.

- Wakeup threshold (14 bits). Defines the vibration threshold in mg above which the machine is assumed to be running. The value is expressed in mg and comprised between 0 and 1 g. Default value is 100 mg.
- RFU: 2 bits
- Special parameters (11 bits). This field's meaning depends on the used mode. currently it only applies to Wake on Analog mode.
- Flag (1 bit). Defines a boolean value switching between different modes (see table below)
- Mode (4 bits). Sets the Beacon into one of the above mentioned modes.

In practice, the interpretation of a payload presented in the above table as "b3009101" would go as follows:

Hex (LE)	Binary Form	wakeup threshold	RFU	Special	flag	mode
b3009101	00000001100100010000000010110011	00000001100100	01	00000000101	1	0011

For more information in the physical interpretation of these parameters, please refer to the AsystomView User Guide documentation.

The next table specifies how flags are interpreted for each selected mode.

Mode	Sync mode	flag interpretation
Inactive	0000	-
Wake on Motion	0001	0: No visual cue if motion is detected
		1: Red led on the side of the device will blink if motion is detected
RFU	0002	RFU
Wake on Scheduler	0003	RFU
Wake on Analog	0004	0: Outside selected region (see AsystomView User Guide for more details)
		1: Inside selected region (see AsystomView User Guide for more details)
Wake on Contact	0005	0: Trigger on signal falling edge
		1: Trigger on signal rising edge

The following table will specify the signification of different fields depending on the chosen mode

Mode	Sync mode	wakeup threshold	special parameter	sync delay	sync timeout
Inactive	0000	-	-	-	-
Wake on Motion	0001	Value in mg	-	In seconds	In seconds
RESERVED	0002	-	-	-	-
Wake on Scheduler	0003		-	-	-
Wake on Analog	0004	-	Value in %*	In seconds	-
Wake on Contact	0005	-	-	In seconds	-

- Wake on Analog is the only case where the special parameter is currently relevant. It represents the allowed variation around the values delimited by the wakeup threshold. Please refer to AsystomView User Guide documentation for more details.

4.1.7 Signature settings (26 bytes)

These settings allow to configure the device to either generate a standard signature or a FFT zoom signature when measurements are triggered.

In the example, the value is 000001000300d007000000000000020000000000000000000000

The format is as follows:

Message	Size	Default value (hex LE)	Decimal value
handle	1 Byte	00	0
activation	1 Byte	00	0
steps	1 Byte	01	1
RFU	1 Byte	00	0
sensor	1 Byte	03	3
orientation	1 Byte	00	0
param_1	4 Bytes	d0070000	2000
param_2	4 Bytes	00000000	0
param_3	4 Bytes	02000000	2
param_4	4 Bytes	00000000	0
param_5	4 Bytes	00000000	0

The handle shall be an incremental server-side generated value ranging from 1 to 255. It shall be incremented each time new FFT zoom settings get sent to the device. When a FFT zoom is computed, this handle is appended to the payload. Server-side, it allows matching the payload data with their corresponding settings.

The activation field can take the following values:

- 0: no FFT zoom, only the standard signature gets computed when measurements get triggered.
- 1: periodic scheduling of FFT zoom versus standard signature (with a 1/steps duty cycle)
- 2: burst scheduling of FFT zoom versus standard signature for a number of consecutive measurements defined by the steps parameter.

The steps parameter is used by the activation mode as defined above.

The sensor parameter specifies either the accelerometer (0x3) or the microphone (0xc).

The orientation parameter only applies to the acelerometer and can take the following values:

- 0: averaging of the three axis of the accelerometer
- 1: X axis
- 2: Y axis
- 4: Z axis

Additional parameters are given below:

param_1: upper bound for the FFT zoom window (in Hz)

param_2: lower bound for the FFT zoom window (in Hz)

param_3: compression as defined below:

- 0: 50 frequency bands encoded as 8 bits
- 1: 50 frequency bands encoded as 16 bits
- 2: 100 frequency bands encoded as 8 bits
- 3: 100 frequency bands encoded as 16 bits
- 4: 200 frequency bands encoded as 8 bits

param_4 and param_5 are only meaningful for device versions strictly greater than 4.40

param_4: spectrum type as defined below:

- 1: RMS spectrum
- 2: Peak spectrum
- 3: RMS velocity spectrum (vibration sensor only)
- 4: Peak velocity spectrum (vibration sensor only)
- 5: RMS Envelope spectrum (vibration sensor only)
- 6: Peak Envelope spectrum (vibration sensor only)

param_5: high pass filer cut-off frequency (envelope spectrum only).

5 UPLINK PACKET SPLIT

In situations where radio quality is insufficient to transfer an uplink in one go, the device will split the uplink in multiple packets while respecting the regional duty cycle allowance.

Uplink FPorts 100 to 104 are reserved for this purpose. For each new message, the first packet will always be on FPort #100.

On order to properly rebuild the integral message, the full message length (1 byte) followed by the number of elements (1 byte) are added at the beginning and a CRC (2 bytes) is piggybacked at the end.

You may recall that for non split messages, we used the FPort to encode the number of elements. Now since the Fport is repurposed for the split function, the value holding the number of elements is pushed back into the payload.

5.1 Example

Lets take a typical 108-byte long payload that would be received on FPort 4 (meaning there are 4 elements to expect inside the payload)

```
006E060E642102724A0BE87DBF7D277DC47C917B397A21794B767F741A72F1887065C66FD96A8C628
C744F6EB972EF743B701B0035006C009807000000000000000190034005F00A20700000000000000
0019003900560096070000000000000000A6210000D907CD7E0000
```

In EU868 Region, the DR1 (SF11) has a maximum payload size of 51 bytes and thus, wouldn't be able to hold the entirety of the message.

So for this specific case, the message will be split into 3 parts.

Reminder :

- The message is always prefixed by its length and the number of elements, and suffixed by a CRC16. The length is 2 + 108 + 2 = 112 bytes (or 0x70 in hex). There are 4 (0x04) elements.

With the new prefix, the payload so far becomes

```
70 04 006E060E642102724A0BE87DBF7D277DC47C917B397A21794B767F741A72F1887065C66FD96
A8C628C744F6EB972EF743B701B0035006C009807000000000000000190034005F00A20700000000
0000000019003900560096070000000000000000A6210000D907CD7E0000
```

- Message integrity is verified by a piggybacked CRC (crc_ccitt_false) at the end of the last transmission.

The CRC16 (crc_ccitt_false) of the above message is E9B0

The final 112 bytes long payload is:

```
70 04 006E060E642102724A0BE87DBF7D277DC47C917B397A21794B767F741A72F1887065C66FD96
A8C628C744F6EB972EF743B701B0035006C009807000000000000000190034005F00A20700000000
0000000019003900560096070000000000000000A6210000D907CD7E0000 E9B0
```

In EU868 DR1, the mesage will be split into three parts, of varying sizes depending on LoRaWAN additional options:

On FPort 100

```
7004006E060E642102724A0BE87DBF7D277DC47C917B397A21794B767F741A72F1887065C66FD96A8
C628C744F6EB972EF74
```

On FPort 101

```
3B701B0035006C009807000000000000000190034005F00A207000000000000000019003900560096007000000000000000
```

On FPort 102

```
A6210000D907CD7E0000E9B0
```

6 DOWNLINK PACKET FORMAT

There are 4 types of messages that can be sent to the Beacon:

FPort	Payload (string)	Request
64	00	Request the device to Reboot
70	<array>	Update scheduling settings
71	<array>	Update advanced settings
74	<array>	Update signature settings

The request to initiate a reboot is simply defined by the FPort used. The payload is “00”.

To update scheduling, advanced or signature settings, please refer to Uplink Examples section above.

IMPORTANT: Before sending any updates to a device, your first step shall be to read the last settings updated by the said device, then to only update the fields related to the changes you want to make. Modifying RFU fields may generate undefined behaviors.

6.1 Example 1: Updating General or Predictive Stack periodicity

Get the current settings value by reading the last data uploaded on FPort 67 from the right offset (see Uplink chapter to identify the right offset)

Let’s assume the payload we have is 054114001e001e006801

Predictive stack periodicity is 1e00 -> 300s -> 5 minutes.

If you want to update it to e.g 2 hours = 7200s. You need to write 720 in the right offset.

720 = 0x2D0. In Little endian, we’ll have to write ‘D002’ two bytes (4 characters) from the right.

Thus, the final payload will be 054114001e00d0026801

6.2 Example 2: Updating Advanced settings

The steps are the same as in the 1st example.

You need to read the existing value first, update the values you want to set and send them back using the relevant port.

Make sure to keep the fields you have not updated intact.