LoRaWAN® UM3110 Payload Structures

This document contains only payload structure definitions for firmware versions 4.0.x, the rest of the information can be found in UM3110 Datasheet .

The number of packets may seem intimidating at first, thus focus on status_packet and usage packet because all necessary information for billing purposes can be found in them. Furthermore all the parameters can be conveniently configured using NAS Connect app.

Since L-Bus uses M-Bus protocol, the parameters in packets are always called *mbus*.

Table of Contents

1 LoRaWAN Payloads

1.1 *usage_packet* and *status_packet*

usage_packet and *status_packet* are based on the same structure, *usage_packet* just contains less data, thus they should have same parser code.

Example A *usage_packet* from fPort 25:

```
02 82                     43 03 24 B29AD03C          01 17060000
```

Example B *status_packet* from fPort 24:

```
82 82 6B D1 16 4A 33 7C 43 03 26 B29AD03C 01382715 01 17060000
```

M-Bus/L-Bus example A *usage_packet* from fPort 25:

```
02 82                     43 23 24 B29AD03C          01 17060000 00                        0374301C00
```

M-Bus/L-Bus example B *status_packet* from fPort 24:

```
82 82 6B D1 16 4A 33 7C 43 23 26 B29AD03C 01382715 01 17060000 60 90 24126270 33380207 0374301C00
```

Header block contains *_packet_type* and *alert_state.*

<table>
<tr><th>Byte</th><th>Type</th><th>Example A</th><th>Example B</th><th>Parameter</th><th>Details</th><th>Example Value</th></tr>
<tr><td>0</td><td>uint8</td><td>0x02</td><td>0x82</td><td><i>packet_type</i></td><td>0x02 - <i>usage_packet</i> (fPort 25)
0x82 - <i>status_packet</i> (fPort 24)</td><td>A: <i>usage_packet</i>
B: <i>status_packet</i></td></tr>
<tr><td>1</td><td>bit8</td><td>0x82</td><td>0x82</td><td><i>alert_state</i></td><td colspan="2">
<table>
<tr><th>Bit</th><th>Parameter</th><th>Example</th></tr>
<tr><td>0</td><td><i>pulse_1_trigger_alert</i></td><td>0</td></tr>
<tr><td>1</td><td><i>pulse_2_trigger_alert</i></td><td>1</td></tr>
<tr><td>2-5</td><td>RFU</td><td>0b0000</td></tr>
<tr><td>6</td><td><i>low_battery</i></td><td>0</td></tr>
<tr><td>7</td><td><i>app_connected_within_a_day</i> (not part of <i>alert_state</i>)</td><td>1</td></tr>
</table>
</td></tr>
</table>

Status info block, contains radio, battery and temperature info, only included when *_packet_type* is *status_packet*.

<table>
<tr><th>Byte</th><th>Type</th><th>Example A</th><th>Example B</th><th>Parameter</th><th>Details</th><th>Example Value</th></tr>
<tr><td>r0</td><td>uint8</td><td></td><td>0x6B</td><td><i>battery_remaining__years</i></td><td>when: <i>packet_type. status_packet</i>
convert: /12.0</td><td>0x6B = 107
107 / 12 ≈ 8,9 (years)</td></tr>
<tr><td>r1</td><td>uint8</td><td></td><td>0xD1</td><td><i>battery_voltage__V</i></td><td>when: <i>packet_type.status_packet</i>
convert: (/100) + 1.5</td><td>0xD1 = 209
209 / 100 + 1,5 = 3,59 (V)</td></tr>
<tr><td>r2</td><td>int8</td><td></td><td>0x16</td><td><i>internal_temperature__C</i></td><td>when: <i>packet_type.status_packet</i></td><td>0x16 = 22(°C)</td></tr>
<tr><td>r3</td><td>bits8</td><td></td><td>0xA4</td><td><i>internal_temperature_</i></td><td colspan="2">when: <i>packet_type.status_packet</i>
<table>
<tr><th>Bit</th><th>Parameter</th><th>Example Value</th></tr>
<tr><td>0-3</td><td><i>internal_temperature_min__C</i>
convert: *-2 + <i>internal_temperature__C</i></td><td>0xA = 10
10 * -2 + <i>internal_ temperature__C</i> = <i>10 * -2 + 22 = -20 + 22 = 2</i></td></tr>
<tr><td>4-7</td><td><i>internal_temperature_max__C</i>
convert: *2 + <i>internal_ temperature__C</i></td><td>0x4 = 4
4 * 2 + <i>internal_ temperature__C</i> = <i>4 * 2 + 22 = 8 + 22 = 30</i></td></tr>
</table>
note: extremes are reset with each <i>status_packet.</i></td></tr>
<tr><td>r4</td><td>uint8</td><td></td><td>0x33</td><td><i>radio_downlink_rssi__dBm</i></td><td>when: <i>packet_type.status_packet</i>
convert: *-1 (multiply with -1 for dBm)</td><td>0x33 = 51
51 * -1 = -51 (dBm)</td></tr>
<tr><td>r5</td><td>bits8</td><td></td><td>0x7C</td><td><i>radio_</i></td><td colspan="2">when: <i>packet_type.status_packet</i>
<table>
<tr><th>Bit</th><th>Parameter</th><th>Example Value</th></tr>
<tr><td>0-3</td><td><i>radio_downlink_snr__dB</i>
convert: *2 - 20</td><td>0xC = 12
12 * 2 -20 = 24 - 20 = 4 (dB)</td></tr>
<tr><td>4-7</td><td><i>radio_uplink_power__dBm</i>
convert: *2</td><td>0x7 = 7
7 * 2 = 14 (dBm)</td></tr>
</table>
example: 0x27 → <i>radio_ downlink_snr__dB</i> = -6 dB and <i>radio_uplink_pwr__dBm</i> = 4 dBm</td></tr>
</table>

Common block containing *actuality_duration* and *reported_interfaces* which determines which interface blocks will follow.

<table>
<tr><th>Byte</th><th>Type</th><th>Example A</th><th>Example B</th><th>Parameter</th><th>Details</th><th>Example Value</th></tr>
<tr><td>r0</td><td>uint8</td><td>0x43</td><td>0x43</td><td>actuality_duration__minutes
actuality_duration_formatted
for all data of reported interfaces</td><td>0-59 - minutes_ago (if <60)
64-155 - quarter_hours_ago (if <156)
157-200 - days_ago (if <201)
208- 253 - weeks_ago (if <254)
254 - over_a_year_ago
255 - n/a
examples: 53 → 53 (minutes), 67 → 67-60 = 7 *15 (min) = 1:45, 161 → 161 - 156 = 5 (days), 210 → 210 - 201 = 9 (weeks)</td><td>0x43 = 67
67 - 60 * 15 (min) = 7 * 15 (min) = 105 (min) = 1: 45</td></tr>
<tr><td>r1</td><td>bits8</td><td>0x23</td><td>0x23</td><td>reported_interfaces</td><td colspan="2">
<table>
<tr><th>Bit</th><th>Parameter</th><th>Value</th><th>Example</th></tr>
<tr><td>0</td><td>pulse_1_sent</td><td></td><td>1</td></tr>
<tr><td>1</td><td>pulse_2_sent</td><td></td><td>1</td></tr>
<tr><td>2-5</td><td>main_interface_type*</td><td>0x00 - none_or_disabled
0x04 - ssi
0x08 - mbus (also L-Bus)</td><td>b0100 → ssi
b1000 → mbus</td></tr>
<tr><td>6-7</td><td>RFU</td><td></td><td>b00</td></tr>
</table>
*UM3110 device can have one main_interface_type determined by ordered hardware.</td></tr>
</table>

Pulse interface 1 block. Only when *reported_interfaces.pulse_1_sent.*

<table>
<tr><th>Byte</th><th>Type</th><th>Example A</th><th>Example B</th><th>Parameter</th><th>Details</th><th>Example Value</th></tr>
<tr><td>r0</td><td>bits8</td><td>0x24</td><td>0x26</td><td>*pulse_1.*</td><td colspan="2">when: *reported_interfaces.pulse_1_sent*
<table>
<tr><th>Bit</th><th>Parameter</th><th>Value</th><th>Example</th></tr>
<tr><td>0</td><td>*input_state*</td><td>*0 - open, 1 - closed*</td><td>*open*</td></tr>
<tr><td>1</td><td>*serial_sent*</td><td>note: sent in *status_packet* if serial has been configured</td><td>A: 0
B: 1</td></tr>
<tr><td>2-3</td><td>*multiplier*
(exponent)</td><td>convert: 10^
note: not relevant in *_trigger* mode, *multiplier* always 1
0 - 1
1 - 10
2 - 100
3 - 1000</td><td>b01 → 10</td></tr>
<tr><td>4-7</td><td>*medium_type*</td><td>00 - *triggers*
01 - *pulses*
02 - *L_water*
03 - *Wh_electricity*
04 - *L_gas*</td><td>*L_water*</td></tr>
</table></td></tr>
<tr><td>r1</td><td rowspan="4">uint32</td><td>0xB2</td><td>0xB2</td><td rowspan="4">*pulse_1.accumulated__triggers*
pulse_1.accumulated__pulses
pulse_1.accumulated__L_water
pulse_1.accumulated__Wh_electricity
pulse_1.accumulated__L_gas</td><td rowspan="4">when: *reported_interfaces.pulse_1_sent*
unit: *pulse_1.medium_type*
convert: **pulse_1.multiplier*</td><td rowspan="4">0x3CD09AB2 → 1020304050
1020304050 * *pulse_1_multiplier* = 10203040500 (units)</td></tr>
<tr><td>r2</td><td>0x9A</td><td>0x9A</td></tr>
<tr><td>r3</td><td>0xD0</td><td>0xD0</td></tr>
<tr><td>r4</td><td>0x3C</td><td>0x3C</td></tr>
<tr><td>r0</td><td rowspan="4">uint32</td><td></td><td>0x01</td><td rowspan="4">*pulse_1.serial*
(sn of external device)</td><td rowspan="4">when: *pulse_1.serial_sent*
note: not available in *triggers* mode</td><td rowspan="4">0x15273801 → 15273801</td></tr>
<tr><td>r1</td><td></td><td>0x38</td></tr>
<tr><td>r2</td><td></td><td>0x27</td></tr>
<tr><td>r3</td><td></td><td>0x15</td></tr>
</table>

Pulse interface 2 block. Only sent when *reported_interfaces.pulse_2_sent.*

<table>
<tr><th>Byte</th><th>Type</th><th>Example A</th><th>Example B</th><th>Parameter</th><th>Details</th><th>Example Value</th></tr>
<tr><td>r0</td><td>bits8</td><td>0x01</td><td>0x01</td><td>*pulse_2.*</td><td colspan="2">when: *reported_interfaces.pulse_2_sent*
<table>
<tr><th>Bit</th><th>Parameter</th><th>Value</th><th>Example</th></tr>
<tr><td>0</td><td>*input_state_immediate*</td><td rowspan="4">see *pulse_1_state*</td><td>*closed*</td></tr>
<tr><td>1</td><td>*serial_sent*</td><td>*not_sent*</td></tr>
<tr><td>2-3</td><td>*multiplier*</td><td>1</td></tr>
<tr><td>4-7</td><td>*medium_type*</td><td>*triggers*</td></tr>
</table></td></tr>
<tr><td>r1</td><td rowspan="4">uint32</td><td>0x17</td><td>0x17</td><td rowspan="4">*pulse_2.accumulated__triggers*
pulse_2.accumulated__pulses
pulse_2.accumulated__L_water
pulse_2.accumulated__Wh_electricity
pulse_2.accumulated__L_gas</td><td rowspan="4">when: *reported_interfaces.pulse_2_sent*
see: *pulse_1.accumulated*</td><td rowspan="4">0x00000617 = 1559 (*triggers*)</td></tr>
<tr><td>r2</td><td>0x06</td><td>0x06</td></tr>
<tr><td>r3</td><td>0x00</td><td>0x00</td></tr>
<tr><td>r4</td><td>0x00</td><td>0x00</td></tr>
<tr><td>r0</td><td rowspan="4">uint32</td><td>-</td><td>-</td><td rowspan="4">*pulse_2.serial*</td><td rowspan="4">when: *pulse_2.serial_sent*
see: *pulse_1.serial*</td><td rowspan="4"></td></tr>
<tr><td>r1</td><td>-</td><td>-</td></tr>
<tr><td>r2</td><td>-</td><td>-</td></tr>
<tr><td>r3</td><td>-</td><td>-</td></tr>
</table>

M-Bus / L-Bus interface block. Only sent when *reported_interfaces.main_interface_type* is *m-bus* . Please note the DIF VIF parsing of *mbus_data_records* content depends on what your M-Bus device is reporting, therefore check your M-Bus Device's datasheet on what data records are available and how to decode them.

<table>
<tr><th>Byte</th><th>Type</th><th>Example A</th><th>Example B</th><th>Parameter</th><th>Details</th><th>Example Value</th></tr>
<tr><td>r0</td><td>bits8</td><td>0x00</td><td>0x60</td><td>*mbus.*</td><td colspan="2">when: *reported_interfaces.main_interface_type* is *m-bus*
<table>
<tr><th>Bit</th><th>Parameter</th><th>Value</th><th>Example</th></tr>
<tr><td>0-3</td><td>*last_bus_status*</td><td>0 - *connected*
1 - *nothing_requested*
3 - *no_response*
5 - *crc_or_len_error*
6 - *parse_error*
7 - *bus_shorted*</td><td>*connected*</td></tr>
<tr><td>4</td><td>*data_records_truncated*</td><td></td><td>0</td></tr>
<tr><td>5</td><td>*state_and_serial_sent*</td><td></td><td>A: 0
B: 1</td></tr>
<tr><td>6</td><td>*serial_extension_sent*</td><td>note: available since 4.0.30</td><td>A: 0
B: 1</td></tr>
<tr><td>7</td><td>RFU</td><td></td><td></td></tr>
</table></td></tr>
<tr><td>r0</td><td>uint8</td><td></td><td>0x90</td><td>*mbus.status*
(status from M-Bus frame)</td><td>when: *mbus.state_and_serial_sent*</td><td></td></tr>
<tr><td>r1</td><td rowspan="4">uint32</td><td></td><td>0x24</td><td rowspan="4">*mbus.serial*
(sn from M-Bus frame)</td><td rowspan="4">when: *mbus.state_and_serial_sent*</td><td rowspan="4">0x70621224 → 70621224</td></tr>
<tr><td>r2</td><td></td><td>0x12</td></tr>
<tr><td>r3</td><td></td><td>0x62</td></tr>
<tr><td>r4</td><td></td><td>0x70</td></tr>
<tr><td>r5</td><td rowspan="2">uint16</td><td></td><td>0x33</td><td rowspan="2">*mbus.manufacturer*</td><td rowspan="4">when: *serial_extension_sent* note: available since 4.0.30 note: M-Bus specification</td><td rowspan="2">0x3833 → "NAS"</td></tr>
<tr><td>r6</td><td></td><td>0x38</td></tr>
<tr><td>r7</td><td>uint8</td><td></td><td>0x02</td><td>*mbus.version* (generation)</td><td>0x02 - 2</td></tr>
<tr><td>r8</td><td>uint8</td><td></td><td>0x07</td><td>*mbus.medium* (type)</td><td>0x07 - *water*</td></tr>
<tr><td>r1</td><td></td><td>0x03</td><td>0x03</td><td rowspan="6">*mbus.data_records [0].*</td><td rowspan="7">when: *reported_interfaces.main_interface_type* note: standard M-Bus data records follow till the end of frame. note: if entire data record does not fit in, it is not included.</td><td>0x03 DIF (int24)</td></tr>
<tr><td>r2</td><td></td><td>0x74</td><td>0x74</td><td>0x74 VIF (actuality, sec)</td></tr>
<tr><td>r3</td><td></td><td>0x20</td><td>0x20</td><td rowspan="3">0x001C20 = 7200 (s)</td></tr>
<tr><td>...</td><td></td><td>0x1C</td><td>0x1C</td></tr>
<tr><td>...</td><td></td><td>0x00</td><td>0x00</td></tr>
<tr><td>...</td><td></td><td>...</td><td>....</td><td>next DIF</td></tr>
<tr><td>...</td><td></td><td>...</td><td>...</td><td>*mbus.data_records [n].*</td><td>...</td></tr>
</table>

Parsed example A:

```
{
  "data": {
    "packet_type": "usage_packet",
    "app_connected_within_a_day": true,
    "active_alerts": {
      "pulse_1_trigger_alert": false,
      "pulse_2_trigger_alert": true,
      "low_battery": false
    },
    "meter_actuality_duration__minutes": 105,
    "meter_actuality_duration_formatted": "1.75 hours",
    "pulse_1": {
      "input_state": "open",
      "muliplier": 10,
      "medium_type": "L_water",
      "accumulated__L_water": 10203040500
    },
    "pulse_2": {
      "input_state": "closed",
      "muliplier": 1,
      "medium_type": "triggers",
      "accumulated__triggers": 1559
    }
  }
}
```

Parsed example B:

```
{

  "data": {
    "packet_type": "status_packet",
    "app_connected_within_a_day": true,
    "active_alerts": {
      "pulse_1_trigger_alert": false,
      "pulse_2_trigger_alert": true,
      "low_battery": false
    },
    "battery_remaining__years": 8.9,
    "battery_voltage__V": 3.59,
    "internal_temperature__C": 22,
    "internal_temperature_min__C": 2,
    "internal_temperature_max__C": 30,
    "radio_downlink_rssi__dBm": -51,
    "radio_downlink_snr__dB": 4,
    "radio_uplink_power__dBm": 14,
    "meter_actuality_duration__minutes": 105,
    "meter_actuality_duration_formatted": "1.75 hours",
    "pulse_1": {
      "input_state": "open",
      "muliplier": 10,
      "medium_type": "L_water",
      "accumulated__L_water": 10203040500,
      "serial": "15273801"
    },
    "pulse_2": {
      "input_state": "closed",
      "muliplier": 1,
      "medium_type": "triggers",
      "accumulated__triggers": 1559
    }
  }
}
```

Parsed example A:

```
{
  "data": {
    "packet_type": "usage_packet",
    "app_connected_within_a_day": true,
    "active_alert": {
      "pulse_1_trigger_alert": false,
      "pulse_2_trigger_alert": true,
      "low_battery": false
    },
    "meter_actuality_duration__minutes": 0,
    "meter_actuality_duration_formatted": "0 minutes",
    "ssi": {
      "sensor": "pressure_30bar_temperature",
      "channel_1": 1.057,
      "channel_2": 23.2
    }
  }
}
```

Parsed example B:

```
{

  "data": {
    "packet_type": "status_packet",
    "app_connected_within_a_day": true,
    "active_alert": {
      "pulse_1_trigger_alert": false,
      "pulse_2_trigger_alert": true,
      "low_battery": false
    },
    "battery_remaining__years": 8.9,
    "battery_voltage__V": 3.59,
    "internal_temperature__C": 22,
    "internal_temperature_min__C": 2,
    "internal_temperature_max__C": 30,
    "radio_downlink_rssi__dBm": -51,
    "radio_downlink_snr__dB": 4,
    "radio_uplink_power__dBm": 14,
    "meter_actuality_duration__minutes": 105,
    "meter_actuality_duration_formatted": "1.75 hours",
    "pulse_1": {
      "input_state": "open",
      "muliplier": 10,
      "medium_type": "L_water",
      "accumulated__L_water": 10203040500,
      "serial": "15273801"
    },
    "pulse_2": {
      "input_state": "closed",
      "muliplier": 1,
      "medium_type": "triggers",
      "accumulated__triggers": 1559
    },
    "mbus": {
      "last_bus_status": "connected",
      "records_truncated": false,
      "status": "0x90",
      "serial": "70621224",
      "data_records_raw": "0374301C00",
      "data_records": {
        "actuality_duration__seconds": 7216
      }
    }
  }
}
```

1.2 *general_configuration_packet*

Example payload from/to fPort 50:

```
12 31 17 02 4F 0A00 01 404B4C00 CEFFFFFF 78563412 0A 00
```

Common half of the packet contains radio parameters and determines, if pulse interfaces are configured.

<table>
<tr><th>Byte</th><th>Type</th><th>Example</th><th>Parameter</th><th>Details</th><th>Example Value</th></tr>
<tr><td>0</td><td>uint8</td><td>0x12</td><td><i>packet_type</i></td><td>0x12 - <i>general_configuration_packet</i></td><td><i>general_configuration_packet</i></td></tr>
<tr><td>1</td><td>bits8</td><td>0x31</td><td><i>configured_parameters</i></td><td colspan="2"><table>
<tr><th>Bit</th><th>Parameter</th><th>Example</th></tr>
<tr><td>0</td><td><i>radio_lorawan_profile_sent</i></td><td>1</td></tr>
<tr><td>1</td><td><i>radio_wmbus_profile_sent</i></td><td>0</td></tr>
<tr><td>2-3</td><td>RFU</td><td>b00</td></tr>
<tr><td>4</td><td><i>pulse_1_sent</i></td><td>1</td></tr>
<tr><td>5</td><td><i>pulse_2_sent</i></td><td>1</td></tr>
<tr><td>6-7</td><td>RFU</td><td>b00</td></tr>
</table></td></tr>
<tr><td>r0</td><td>uint8</td><td>0x17</td><td><i>radio_lorawan_profile</i></td><td>when: <i>configured_parameters.radio_lorawan_profile_sent</i>

0x00 - <i>lorawan_disabled</i>
0x01 - <i>lorawan_24_h_privacy</i>
0x02 - <i>lorawan_24_h</i>
0x03 - <i>lorawan_12_h</i>
0x07 - <i>lorawan_1_h_static</i>
0x08 - <i>lorawan_15_min_static</i>
0x17 - <i>lorawan_1_h_dynamic</i>
0x18 - <i>lorawan_15_min_dynamic</i></td><td><i>lorawan_1_h_dynamic</i></td></tr>
<tr><td>r0</td><td>uint8</td><td>-</td><td><i>radio_wmbus_profile</i></td><td>when: <i>configured_parameters.radio_wmbus_profile_sent</i>

0x00 - <i>wmbus_disabled</i>
0x01 - <i>wmbus_privacy</i>
0x02 - <i>wmbus_driveby</i>
0x03 - <i>wmbus_fixnet</i></td><td></td></tr>
</table>

Pulse 1 interface configuration. Sent when *configured_parameters.pulse_1.sent.*

Example payload to/from fPort 50: 123117024F0A0001404B4C00CEFFFFFF785634120A00

<table>
<tr><th>Byte</th><th>Type</th><th>Example</th><th>Parameter</th><th>Details</th><th>Example Value</th></tr>
<tr><td>r0</td><td>bits8</td><td>0x02</td><td><i>pulse_1.</i></td><td colspan="2">when: <i>configured_parameters.pulse_1.sent</i>
<table>
<tr><th>Bit</th><th>Parameter</th><th>Value</th><th>Example</th></tr>
<tr><td>0-3</td><td><i>input_mode_and_unit</i></td><td>0 - <i>disabled</i> (all other <i>_configured_parameters</i> must be 0)
1 - <i>pulses</i>
2 - <i>L_water</i>
3 - <i>Wh_electricity</i>
4 - <i>L_gas</i>
9 - <i>triggers_1_sec</i>
10 - <i>triggers_10_sec</i>
11 - <i>triggers_1_min</i>
12 - <i>triggers_1_h</i></td><td><i>counter_L_water</i></td></tr>
<tr><td>4-7</td><td>RFU</td><td></td><td>b0000</td></tr>
</table></td></tr>
<tr><td>r1</td><td>bits8</td><td>0x4F</td><td><i>pulse_1.
configured_parameters</i></td><td colspan="2">when: <i>configured_parameters. pulse_1_sent</i>
<table>
<tr><th>Bit</th><th>Parameter</th><th>Availability</th><th>Value</th><th>Example</th></tr>
<tr><td>0</td><td><i>multiplier_sent</i></td><td><i>counter_mode</i> except <i>counter_pulses</i></td><td></td><td>1</td></tr>
<tr><td>1</td><td><i>accumulated_absolute_sent</i></td><td><i>counter_mode</i></td><td></td><td>1</td></tr>
<tr><td>2</td><td><i>accumulated_offset_sent</i></td><td><i>counter_mode</i></td><td></td><td>1</td></tr>
<tr><td>3</td><td><i>serial_sent</i></td><td><i>counter_mode</i></td><td></td><td>1</td></tr>
<tr><td>4-5</td><td>RFU</td><td></td><td></td><td>b00</td></tr>
<tr><td>6-7</td><td><i>multiplier</i>

(applied only on <i>accumulated _absolute</i> and <i>accumulated _offset</i>)</td><td><i>counter_mode</i></td><td>0 - 1
1 - 10
2 - 100
3 - 1000</td><td>b01 → 10</td></tr>
</table></td></tr>
<tr><td>r0</td><td rowspan="2">uint16</td><td>0x0A</td><td rowspan="2"><i>pulse_1.
multiplier_numerator</i></td><td rowspan="3">when: <i>pulse_1.configured_parameters.multiplier_sent</i>
note: <i>medium_type</i> units per pulse, combine <i>multiplier_numerator</i> and <i>multiplier_denominator</i> for fractional multiplier
note: <i>multiplier_numerator</i> / <i>multiplier_denominator</i> must be larger than 1</td><td rowspan="3">0x000A → 10
0x01 → 1

multiplier = 10 / 1 = 10</td></tr>
<tr><td>r1</td><td>0x00</td></tr>
<tr><td>r2</td><td>uint8</td><td>0x01</td><td><i>pulse_1.
multiplier_denominator</i></td></tr>
<tr><td>r0</td><td rowspan="4">uint64</td><td>0x40</td><td rowspan="4"><i>pulse_1.
accumulated_absolute</i></td><td rowspan="4">when: <i>pulse_1.configured_parameters.accumulated_absolute_sent</i>
unit: <i>pulse_1.mode_configuration.input_mode_and_unit</i>
<i>convert: *pulse_1.configured_parameters.multiplier</i></td><td rowspan="4">0x004C4B40
= 5000000

5000000 * <i>multiplier</i> =

5000000 * 10
= 50000000 (L)</td></tr>
<tr><td>r1</td><td>0x4B</td></tr>
<tr><td>r2</td><td>0x4C</td></tr>
<tr><td>r3</td><td>0x00</td></tr>
<tr><td>r0</td><td rowspan="4">int32</td><td>0xCE</td><td rowspan="4"><i>pulse_1.
accumulated_offset</i></td><td rowspan="4">when: <i>pulse_1.configured_parameters.accumulated_offset_sent</i>
unit: <i>pulse_1.mode_configuration.input_mode_and_unit</i>
convert: <i>*pulse_1.configured_parameters.multiplier</i></td><td rowspan="4">0xFFFFFFCE = -50

-50 * <i>multiplier</i> =
-50 * 10 = -500 (L)</td></tr>
<tr><td>r1</td><td>0xFF</td></tr>
<tr><td>r2</td><td>0xFF</td></tr>
<tr><td>r3</td><td>0xFF</td></tr>
<tr><td>r0</td><td rowspan="4">uint32</td><td>0x78</td><td rowspan="4"><i>pulse_1.serial</i></td><td rowspan="4">when: <i>pulse_1.configured_parameters.serial_sent</i>
format: hex</td><td rowspan="4">0x12345678 →
12345678</td></tr>
<tr><td>r1</td><td>0x56</td></tr>
<tr><td>r2</td><td>0x34</td></tr>
<tr><td>r3</td><td>0x12</td></tr>
</table>

Pulse 2 interface configuration. Sent when *configured_parameters.pulse_2.sent.*

Byte	Type	Example	Parameter	Details	Example Value
r0	bits8	0x0A	*pulse_2.input_mode_and_unit*	similar: *pulse_1.mode_configuration*	0xA = 10 → *trigger_10_sec*
r1	bits8	0x00	*pulse_2.configured_parameters*	similar: *pulse_1.configured_parameters*	0b00000000
r0	uint16	-	*pulse_2.multiplier_numerator*	similar: *pulse_1.multiplier_numerator*	
r1		-			
r2	uint8	-	*pulse_2.multiplier_denominator*	similar: *pulse_1.multiplier_denominator*	
r0	uint32	-	*pulse_2.accumulated_absolute*	similar: *pulse_1.accumulated_absolute*	
r1		-			
r2		-			

r3		-			
r0	int32	-	*pulse_2.accumulated_offset*	similar: *pulse_1.accumulated_offset*	
r1		-			
r2		-			
r3		-			
r0	uint32	-	*pulse_2.serial*	similar: *pulse_1.serial*	
r1		-			
r2		-			
r3		-			

Parsed example:

```
{
  "data": {
    "packet_type": "general_configuration_packet",
    "radio_lorawan_profile": "lorawan_1_h_dynamic",
    "pulse_1": {
      "input_mode_and_unit": "L_water",
      "multiplier_numerator": 10,
      "multiplier_denominator": 1,
      "accumulated_absolute": 50000000,
      "accumulated_offset": -500,
      "serial": "12345678"
    },
    "pulse_2": {
      "input_mode_and_unit": "triggers_10_sec"
    }
  }
}
```

1.3 *mbus_configuration_packet*

Example payload to/from fPort 50: 14 12 0C13 8C2013 0CFD11

<table>
<tr><th>Byte</th><th>Type</th><th>Example</th><th>Parameter</th><th>Details</th><th>Example Value</th></tr>
<tr><td>0</td><td>uint8</td><td>0x14</td><td><i>packet_type</i></td><td>0x14 - <i>mbus_configuration_packet</i></td><td><i>mbus_configuration_packet</i></td></tr>
<tr><td>1</td><td></td><td>0x12</td><td><i>general_parameters</i></td><td colspan="2"><table>
<tr><th>Bit</th><th>Parameter</th><th>Details</th><th>Example</th></tr>
<tr><td>0-3</td><td><i>mbus_usage_data_record_count</i></td><td rowspan="2">0 ...10 - <i>count</i>
0xF - <i>ignore</i></td><td>2</td></tr>
<tr><td>4-7</td><td><i>mbus_status_data_record_count</i></td><td>1</td></tr>
</table>
0xEE - <i>wmbus_frame_data_records</i></td></tr>
<tr><td>r0</td><td></td><td>0x0C</td><td rowspan="2"><i>mbus_usage_data_record_headers</i> [0]</td><td rowspan="5">raw block of M-Bus data record Headers that are to be included into <i>usage_packet</i>
count: <i>mbus_usage_data_record_count</i></td><td rowspan="2">0x0C 0x13 (volume)</td></tr>
<tr><td>r1</td><td></td><td>0x13</td></tr>
<tr><td>r2</td><td></td><td>0x8C</td><td rowspan="3"><i>mbus_usage_data_record_headers</i> [1]
...</td><td rowspan="3">0x8C 0x20 0x13 (volume tariff 2)</td></tr>
<tr><td>r3</td><td></td><td>0x20</td></tr>
<tr><td>..</td><td></td><td>0x13</td></tr>
<tr><td>..</td><td></td><td>0x0C</td><td rowspan="3"><i>mbus_status_data_record_headers</i> [0]
...</td><td rowspan="3">raw block of M-Bus data record Headers that are to be included into <i>status_packet</i>
count: <i>mbus_status_data_record_count</i></td><td rowspan="3">0x0C 0xFD 0x11 (ownership number)</td></tr>
<tr><td>..</td><td></td><td>0xFD</td></tr>
<tr><td>..</td><td></td><td>0x11</td></tr>
</table>

It must be considered that at LoRaWAN Spreading Factor 12, the total packet lenght is limited to 51 bytes.

If in *usage_packet* the requested *data_record*s take more space that LoRaWAN can transfer, last *data_record*s will not be sent. Same applies to *status _packet.*

Parsed example:

```
{
  "data": {
    "packet_type": "mbus_configuration_packet",
    "mbus_data_record_header_raw": "0C138C20130CFD11"
  }
}
```

1.4 *location_configuration_packet*

In EU868 region DR0 has maximum payload length of 51 bytes which is less than maximum data in *location_configuration*. Therefore the packet may be sent or received in two parts, the longest address field and all the rest of the fields. *address*, *id_customer* and *id_location* all contain utf-8 string where single unicode symbol can take several bytes.

Example payload to/from fPort 50:
21 1F AC5E6D23 0017C10E F4 12 c396c3b662696b75205374722e2032 2d3136 10 32365630303030303030303337313734 05 3132414236

<table>
<tr><th>Byte</th><th>Type</th><th>Example</th><th>Parameter</th><th>Details</th><th>Example Value</th></tr>
<tr><td>0</td><td>uint8</td><td>0x21</td><td><i>packet_type</i></td><td>0x21 - <i>location_configuration_packet</i></td><td><i>location_configuration_packet</i></td></tr>
<tr><td>1</td><td>bits8</td><td>0x1F</td><td><i>configured_parameters</i></td><td colspan="2"><table>
<tr><th>Bit</th><th>Parameter</th><th>Example</th></tr>
<tr><td>0</td><td><i>gps_position_sent</i></td><td>b1</td></tr>
<tr><td>1</td><td><i>time_zone_sent</i></td><td>b1</td></tr>
<tr><td>2</td><td><i>address_sent</i></td><td>b1</td></tr>
<tr><td>3</td><td><i>id_customer_sent</i></td><td>b1</td></tr>
<tr><td>4</td><td><i>id_location_sent</i></td><td>b1</td></tr>
<tr><td>5-7</td><td>RFU</td><td>b000</td></tr>
</table></td></tr>
<tr><td>r0</td><td rowspan="4">int32</td><td>0xAC</td><td rowspan="4"><i>gps_position_latitude__deg</i></td><td rowspan="8">when: <i>configured_parameters.gps_position_sent</i>
converter: /10e7

0x7FFFFFFF - <i>not_configured</i></td><td rowspan="4">0x236D5EAC = 594370220

594370220 / 10000000 = 59.4370220 = 59°26'13.3"N</td></tr>
<tr><td>r1</td><td>0x5E</td></tr>
<tr><td>r2</td><td>0x6D</td></tr>
<tr><td>r3</td><td>0x23</td></tr>
<tr><td>r4</td><td rowspan="4">int32</td><td>0x00</td><td rowspan="4"><i>gps_position_longitude__deg</i></td><td rowspan="4">0x0EC11700 = 247535360

247535360 / 10000000 = 24.7535360 = 24°45'12.7"E</td></tr>
<tr><td>r5</td><td>0x17</td></tr>
<tr><td>r6</td><td>0xC1</td></tr>
<tr><td>r7</td><td>0x0E</td></tr>
<tr><td>r0</td><td>int8</td><td>0xF4</td><td><i>time_zone__h</i>

(setting winter time recommended)</td><td>when: <i>configured_parameters.time_zone_sent</i>
converter: /4
min: -12 h
max: 14 h</td><td>0xF4 = -12

-12 / 4 = -3 h = UTC-3</td></tr>
<tr><td>r0</td><td>uint8</td><td>0x12</td><td><i>address_len</i>

(length in bytes)</td><td>when: <i>configured_parameters.address_sent</i>
max: 38</td><td>0x12 = 18 (bytes)</td></tr>
<tr><td>r1</td><td rowspan="5">string</td><td>0xc3</td><td rowspan="5"><i>address</i></td><td rowspan="5">when: <i>address_len</i> &gt; 0
length: <i>address_len</i>
type: utf-8 string
note: does not end with '\0'</td><td rowspan="5">0xc3 0x96 0xc3 0xb6
0x62 0x69 0x6b 0x75
0x20 0x53 0x74 0x72
0x2e 0x20 0x32 0x2d
0x31 0x36 →
"Ööbiku Str. 2-16"</td></tr>
<tr><td>r2</td><td>0x96</td></tr>
<tr><td>...</td><td rowspan="3">0xc3
0xb6
0x62
0x69
0x6b
0x75
0x20
0x53
0x74
0x72
0x2e
0x20
0x32
0x2d
0x31 0x36</td></tr>
<tr><td>...</td></tr>
<tr><td>r0+
_<i>address_len</i></td></tr>
<tr><td>r0</td><td>uint8</td><td>0x10</td><td><i>id_customer_len</i>

(length in bytes)</td><td>when: <i>configured_parameters.id_customer_sent</i>
max: 16</td><td>0x10 = 16 (bytes)</td></tr>
<tr><td>r1</td><td rowspan="5">string</td><td>0x32</td><td rowspan="5"><i>id_customer</i></td><td rowspan="5">when: <i>id_customer_len</i> &gt; 0
length: <i>id_customer_len</i>
type: utf-8 string
note: string does not end with '\0'</td><td rowspan="5">0x32 0x36 0x56 0x30
0x30 0x30 0x30 0x30
0x30 0x30 0x30 0x33
0x37 0x31 0x37 0x34 →
"26V0000000037174"</td></tr>
<tr><td>r2</td><td>0x36</td></tr>
<tr><td>...</td><td rowspan="3">0x56
0x30
0x30
0x30
0x30
0x30</td></tr>
<tr><td>...</td></tr>
<tr><td></td></tr>
</table>

	r0+ *_id_customer_len*		0x30 0x30 0x30 0x33 0x37 0x31 0x37 0x34			
	r0	uint8	0x05	*id_location_len* (length in bytes)	when: *configured_parameters.id_location_sent* max: 16	0x05 = 5 (bytes)
	r1	string	0x31	*id_location*	when: *id_location_len* > 0 length: *id_location_len* type: utf-8 string note: string does not end with '\0'	0x31 0x32 0x41 0x42 0x36 → "12AB6"
	r2		0x32			
	...		0x41			
	...		0x42			
	r0+ *_id_location_len*		0x36			

Parsed example:

```
{
  "data": {
    "packet_type": "location_configuration_packet",
    "gps_position_latitude__deg": 59.437022,
    "gps_position_longitude__deg": 24.753536,
    "time_zone__h": -3,
    "address": "Ööbiku Str. 2-16",
    "id_customer": "26V0000000037174",
    "id_location": "12AB6"
  }
}
```

1.5 Configuration requests

The response is sent to same port in the same format as configuration message for this package.

Example payload to fPort 49: 12

Byte	Type	Example	Parameter	Details	Example Value
0	uint8	0x12	*packet_type*	0x12 - *general_configuration_request* 0x14 - *mbus_configuration_request* 0x21 - *location_configuration_request*	*general_configuration_request*

Parsed example:

```
{
  "data": {
    "packet_type": "general_configuration_request"
  }
}
```

1.6 Commands

enter_dfu_command response is *shutdown_packet* with reason *enter_dfu*, sent only if duty allows it. local_time_request is responded with local_time_response.

Example payload to fPort 60: FF

Byte	Type	Example	Parameter	Details	Example Value
0	uint8	0xFF	*packet_type*	0x03 - *local_time_request* 0x81 - *mbus_available_data_records_request* 0xFF - *enter_dfu_command*	*enter_dfu_command*

Parsed example:

```
{
  "data": {
    "packet_type": "enter_dfu_command"
  }
}
```

1.7 *local_time_response*

Example payload from fPort 60: 03 34546A5F

Byte	Type	Example	Parameter	Details	Example Value
0	uint8	0x03	*packet_type*	0x03 - *local_time_response*	*local_time_response*
1	uint32_t	0x34	*device_local_time__s* *device_local_time_formatted* (CM30xx local time)	note: treat as unix epoch in local time	0x5F6A5434 = 1600803892 → 2020-09-22T19:44:52Z (local time)
2		0x54			
3		0x6A			
4		0x5F			

Parsed example:

```
{
  "data": {
    "packet_type": "local_time_response",
    "device_local_time__s": 1600803892,
    "device_local_time_formatted": "2020-09-22T19:44:52Z"
  }
}
```

1.8 *mbus_available_data_records*

At this point M-Bus/L-Bus available data records is sent only triggered from app's Extended Function page. Contains data record headers and M-Bus /L-Bus fixed header for RSP_UD “All data” selection.

Example payload from fPort 61:
  81 C1 78563412 A511 70 07 C1 90 0000 0C138C2013 ..

<table>
<tr><th>Byte</th><th>Type</th><th>Example</th><th>Parameter</th><th>Details</th><th>Example Value</th></tr>
<tr><td>0</td><td>uint8</td><td>0x81</td><td><i>packet_type</i></td><td>0x81 - <i>mbus_connected_packet</i></td><td><i>mbus_connected_packet</i></td></tr>
<tr><td>1</td><td>uint8</td><td>0xC1</td><td><i>configured_parameters</i></td><td colspan="2">
<table>
<tr><th>Bit</th><th>Parameter</th><th>Example</th></tr>
<tr><td>0 - 2</td><td><i>_packet_number</i> (in case data_records do not fit available payload length)</td><td>1</td></tr>
<tr><td>3</td><td><i>_more_packets_following</i> (it is not the last packet)</td><td>0</td></tr>
<tr><td>4 - 5</td><td>RFU</td><td>b00</td></tr>
<tr><td>6</td><td><i>mbus_header_sent</i></td><td>1</td></tr>
<tr><td>7</td><td>RFU</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>r0</td><td rowspan="4">uint32</td><td>0x78</td><td rowspan="12"><i>mbus_headers.</i>
when: <i>configured_parameters. mbus_header_sent</i></td><td rowspan="4"><i>serial</i> (id)
format: bcd</td><td rowspan="4">0x12345678 → "12345678"</td></tr>
<tr><td>r1</td><td>0x56</td></tr>
<tr><td>r2</td><td>0x34</td></tr>
<tr><td>r3</td><td>0x12</td></tr>
<tr><td>r4</td><td rowspan="2">uint16</td><td>0xA5</td><td rowspan="2"><i>manufacturer</i></td><td rowspan="2">0xA5 0x11 → "DME"</td></tr>
<tr><td>r5</td><td>0x11</td></tr>
<tr><td>r6</td><td>uint8</td><td>0x70</td><td><i>version</i></td><td>0x70</td></tr>
<tr><td>r7</td><td>uint8</td><td>0x07</td><td><i>medium</i></td><td>0x07 → Water</td></tr>
<tr><td>r8</td><td>uint8</td><td>0xC1</td><td><i>access_number</i></td><td>0xC1 = 193</td></tr>
<tr><td>r9</td><td>uint8</td><td>0x90</td><td><i>status</i></td><td>0x90</td></tr>
<tr><td>r10</td><td rowspan="2">uint16</td><td>0x00</td><td rowspan="2"><i>signature</i></td><td rowspan="2">0000</td></tr>
<tr><td>r11</td><td>0x00</td></tr>
<tr><td>r0</td><td></td><td>0x0C</td><td rowspan="6"><i>mbus_data_record_header[]</i></td><td rowspan="6">note: standard M-Bus data record header follow till the end of frame.
note: if entire data record does not fit in, it is not included.</td><td rowspan="2">DIF (int32)
VIF (volume)</td></tr>
<tr><td>r1</td><td></td><td>0x13</td></tr>
<tr><td>r2</td><td></td><td>0x8C</td><td rowspan="3">DIF (int32)
DIFE (tariff 2)
VIF (volume)</td></tr>
<tr><td>..</td><td></td><td>0x20</td></tr>
<tr><td>..</td><td></td><td>0x13</td></tr>
<tr><td>..</td><td></td><td>..</td><td>..</td></tr>
</table>

*LVAR only if so defined in respective DIF.

** If packet gets longer than available transmit payload (dependent on radio Spreading Factor and pending MAC commands), particular DIB header is not cut in two but sent instead in extra packet with all of the rest of unsent DIB headers.

NB! Manufacturer specific data is not requestable.

Parsed example:

```
{
  "data": {
    "packet_type": "mbus_available_data_records",
    "packet_number": 1,
    "more_packets_following": false,
    "mbus_headers": {
      "serial": "12345678",
      "manufacturer": "DME",
      "version": "0x70",
      "medium": "water",
      "access_number": 193,
      "status": "0x90",
      "signature": "0000"
    },
    "mbus_data_record_header_raw": "0C138C2013"
  }
}
```

1.9 *faulty_downlink_packet*

If any downlink configuration or command packet parsing fails in UM3110, an error code is sent back.

Example payload from fFort 99: 13 32 05

Byte	Type	Example	Parameter	Details	Example Value
0	uint8	0x13	*packet_type*	0x13 - *faulty_downlink_packet*	*faulty_downlink_packet*
1	uint8	0x32	*packet_fport*	original fPort where the invalid configuration packet arrived	0x32 = 50 (fPort)
2	uint8	0x05	*packet_error_code*	0x00 - n/a 0x01 - n/a 0x02 - *unknown_fport* 0x03 - *packet_size_short* 0x04 - *packet_size_long* 0x05 - *value_error* 0x06 - *protocol_parse_error* 0x07 - *reserved_flag_set* 0x08 - *invalid_flag_combination* *unsupported_header* *internal_error*	*value_error*

Parsed example:

```
{
  "data": {
    "packet_type": "faulty_downlink_packet",
    "packet_fport": 50,
    "packet_error_reason": "value_error"
  },
  "warnings": [
    "faulty_downlink_packet: value_error"
  ]
}
```

1.10 *boot_packet*

fPort 99. Sent after boot and every rejoin.

Example payload:
00 27001650 020312 80 90 06 371100

<table>
<tr><th>Byte</th><th>Type</th><th>Example</th><th>Parameter</th><th>Details</th><th>Example Value</th></tr>
<tr><td>0</td><td>uint8</td><td>0x00</td><td><i>packet_type</i></td><td>0x00 - <i>boot_packet</i></td><td><i>boot_packet</i></td></tr>
<tr><td>1</td><td rowspan="4">uint32</td><td>0x27</td><td rowspan="4"><i>device_serial</i>
(sn of UM3110)</td><td rowspan="4">formatting: hex</td><td rowspan="4">0x50160027 → 50160027</td></tr>
<tr><td>2</td><td>0x00</td></tr>
<tr><td>3</td><td>0x16</td></tr>
<tr><td>4</td><td>0x50</td></tr>
<tr><td>5</td><td>uint8</td><td>0x02</td><td rowspan="3"><i>device_firmware_version</i></td><td><i>hardware</i></td><td rowspan="3">0x02 0x03 0x12 → 2.3.18</td></tr>
<tr><td>6</td><td>uint8</td><td>0x03</td><td><i>major</i></td></tr>
<tr><td>7</td><td>uint8</td><td>0x12</td><td><i>minor</i></td></tr>
<tr><td>8</td><td>bits8</td><td>0x80</td><td><i>wakeup_reason_mcu</i></td><td colspan="2">note: all fields are reset if <i>packet_reason.lorawan_rejoin</i>
<table>
<tr><th>Bit</th><th>Parameter</th><th>Example</th></tr>
<tr><td>0</td><td>RFU</td><td>0</td></tr>
<tr><td>1</td><td><i>watchdog_reset</i></td><td>0</td></tr>
<tr><td>2</td><td><i>soft_reset</i> (e.g. from dfu_mode)</td><td>0</td></tr>
<tr><td>3</td><td>RFU</td><td>0</td></tr>
<tr><td>4</td><td><i>magnet_wakeup</i></td><td>0</td></tr>
<tr><td>5-6</td><td>RFU</td><td>b00</td></tr>
<tr><td>7</td><td><i>nfc_wakeup</i></td><td>1</td></tr>
</table>
example: [<i>nfc_wakeup</i>]</td></tr>
<tr><td>9</td><td>bits8</td><td>0x90</td><td></td><td colspan="2">
<table>
<tr><th>Bit</th><th>Parameter</th><th>Value</th><th>Example</th></tr>
<tr><td>0-3</td><td>RFU</td><td></td><td>b0000</td></tr>
<tr><td>4-6</td><td><i>packet_reason</i></td><td>0 - <i>unknown_reset</i>
1 - <i>from_shutdown</i>
2 - <i>from_dfu</i>
3 - <i>forced_reset</i>
4 - <i>lorawan_rejoin</i></td><td>b001 = 1 → <i>from_shutdown</i></td></tr>
<tr><td>7</td><td><i>configuration_restored</i></td><td></td><td>1</td></tr>
</table></td></tr>
<tr><td>10</td><td>uint8</td><td>0x06</td><td><i>hardware_configuration</i></td><td>0x00 - <i>pulse_only</i>
0x04 - <i>pulse_mbus</i>
0x06 - <i>pulse_lbus</i>

<i>0x08 - pulse_only</i></td><td><i>pulse_lbus</i></td></tr>
<tr><td>13</td><td>uint24</td><td>0x37</td><td><i>device_uptime_accumulated__h</i></td><td></td><td>0x001137 = 4407 (h)</td></tr>
</table>

Parsed example:

```
{
  "data": {
    "packet_type": "boot_packet",
    "device_serial": "50160027",
    "device_firmware_version": "2.3.18",
    "wakeup_reason_mcu": [
      "nfc_wakeup"
    ],
    "packet_reason": "from_shutdown",
    "configuration_restored": true,
    "hardware_configuration": "pulse_lbus",
    "device_uptime_accumulated__days": 183.63
  }
}
```

1.11 *shutdown_packet*

fPort 99. If LoRaWAN is not duty-limited, shutdown packet is sent out to indicate the shutdown reason.

Example payload:
  01 33 82826BD1164A337C432326B29AD03C01382715011706000020902412627003743O1C00

Byte	Type	Example	Parameter	Details	Example Value
0	uint8	0x01	*packet_type*	0x01 - *shutdown_packet*	*shutdown_packet*
1	uint8	0x33	*shutdown_reason*	*enter_dfu* 0x33 - *app_shutdown* 0x34 - *switch_to_wmbus*	*app_shutdown*
2	rest of bytes	0x82	*status_packet*	comment: regular *status_packet* follows	see *status_packet* example
3		0x82			
4		0x6B			
...		...			
...		...			

Example parsed:

```
{
  "data": {
    "packet_type": "shutdown_packet",
    "app_connected_within_a_day": true,
    "active_alerts": {
      "pulse_1_trigger_alert": false,
      "pulse_2_trigger_alert": true,
      "low_battery": false
    },
    "battery_remaining__years": 8.9,
    "battery_voltage__V": 3.59,
    "internal_temperature__C": 22,
    "internal_temperature_min__C": 2,
    "internal_temperature_max__C": 30,
    "radio_downlink_rssi__dBm": -51,
    "radio_downlink_snr__dB": 4,
    "radio_uplink_power__dBm": 14,
    "meter_actuality_duration__minutes": 105,
    "meter_actuality_duration_formatted": "1.75 hours",
    "pulse_1": {
      "input_state": "open",
      "muliplier": 10,
      "medium_type": "L_water",
      "accumulated__L_water": 10203040500,
      "serial": "15273801"
    },
    "pulse_2": {
      "input_state": "closed",
      "muliplier": 1,
      "medium_type": "triggers",
      "accumulated__triggers": 1559
    },
    "mbus": {
      "last_bus_status": "connected",
      "records_truncated": false,
      "status": "0x90",
      "serial": "70621224",
      "data_records_raw": "0374301C00",
      "data_records": {
        "actuality_duration__seconds": 7216
      }
    },
    "shutdown_reason": "app_shutdown"
  }
}
```

2 wM-Bus Payload

2.0.1 Variable Data Records Example

Starts with internally accumulated values if enabled (Pulse channels). If L-Bus/M-Bus is connected, data records in the same order will follow.

Maximum length of forwarded frame is 89 if Pulse channels are disabled and 74 if both Pulse channels are enabled.

Note: *pulse_1.accumulated__x* is sent only if channel is enabled. *pulse_1.serial is only sent if the serial has been configured.*

Note: Longer frame means shorter battery life. See battery estimation in NAS Connect app. Remove unnecessary pulse channel or serial or change wM-Bus transmission mode to optimise battery life.

Example payload: 2F2F 0374 C00300 02FD74 2107 0403 87D61200 0C78 01382715 844013 CECA2300 8C4078 44332211 2F2F2F2F2F2F2F2F2F 2F2F2F2F2F2F

Type	Example	Parameter	Details	Example Value
uint8	0x2F	decryption validation	Frame always starts with 0x2F, 0x2F	
uint8	0x2F			
uint8	0x03	data record header	DIF: uint24	
uint8	0x74		VIF: Actuality duration, sec	
uint24	0xC0	data record data *actuality_duration__s*		0x0003C0 = 960 s
	0x03			
	0x00			
uint8	0x02	data record header	DIF: uint16	
uint8	0xFD		VIF: Remaining battery, days	
uint8	0x74			
uint16	0x21	data record data *battery_remaining__days*		0x0721 → 1825 (days)
	0x07			
uint8	0x04	data record header	DIF: uint32, subunit 0	
uint8	0x03		VIF: Wh	
uint32	0x87	data record data *pulse_1_accumulated__Wh*		0x0012D687 = 1234567 Wh
	0xD6			
	0x12			
	0x00			
uint8	0x0C	data record header	DIF: 8 digit BCD, subunit 0	
uint8	0x78		VIF: Fabrication ID	
uint32	0x01	data record data *pulse_1_serial*		0x15273801 → 15273801
	0x38			
	0x27			
	0x15			
uint8	0x84	data record header	DIF: uint32, subunit 1	
uint8	0x40			
uint8	0x13		VIF: m3	
uint32	0xCE	data record data *pulse_2_accumulated__m3*		0x0023CACE = 2345,678 m3 = 2345678 L
	0xCA			
	0x23			
	0x00			
uint8	0x8C	data record header	DIF: 8 digit BCD, subunit 1	
uint8	0x40			
uint8	0x78		VIF: Fabrication ID	
uint32	0x44	data record data *pulse_2_serial*		0x11223344 → 11223344

	0x33			
	0x22			
	0x11			
... ...		The rest of the data records (forwarded from L-Bus).		
uint8 .. uint8	0x2F .. 0x2F	Block padding	Padding that ensures that the frame length is multiple of 16.	

2.0.2 Possible wM-Bus Data Record headers from pulse channels:

Actuality duration DIF VIF:
0x03 0x74 - actuality duration, s

Pulse Counter DIF:
0x04 - *pulse_1*
0x84 0x40 - *pulse_2*

Pulse Counter VIFs for selected input mode:
0xFD 0x61 - counter, pulses
0xFD 0x61 - counter, trigger alerts
b00010nnn - counter, water (m3)
b00010nnn - counter, gas (m3)
b00000nnn - counter, electricity (Wh)

Pulse Channel Serial DIF VIF:
0x0C 0x78 - *pulse_1_serial*
0x8C 0x40 0x78 - *pulse_2_serial*

Variable data records are wrapped inside Data Link Layer and Transport Layer as seen below

2.1 Data Link Layer

Type	Example	Parameter Name	Example Value
uint8	0x44	C-Field	
uint8	0x33	M-Field (Manufacturer ID) 3338 converts to	0x33 0x38 → "NAS"
uint8	0x38		
uint32	0x44	A-Field (Identification No.) wM-Bus address, which is second half of LoRaWAN DevEUI. So NAS DevEUI 70B3D5B020048844 → 20048844	0x44 0x88 0x04 0x20 → 20048844 (wM-Bus address)
	0x88		
	0x04		
	0x20		
uint8	0x01	Version	1
uint8	0x0E	Medium	0x0E → Bus/System

2.2 Transport Layer

When L-Bus / M-Bus meter is connected (with Diehl Hydrus Meter):

Type	Example	Parameter Name	Example Value
uint8	0x72	Control Information	0x7A - Full M-Bus frame, long data header
uint32	0x11	Identification Number	65222711
	0x27		
	0x22		
	0x65		
uint8	0xA5	Manufacturer ID	"DME"
uint8	0x11		
uint8	0x70	Version	
uint8	0x07	Medium	0x07 - Water
uint8	0x00	Access Counter	
uint8	0x90	Status	
uint8	0x00	Configuration Field for Security	
uint8	0x02		

No L-Bus / M-Bus:

Example	Parameter Name	Example Value
0x7A	Control Information	0x7A - Full M-Bus frame, short data header
0x00	Access Counter	
0x00	Status	
0x00	Configuration Field for Security	
0x02		

3 Annex: Understanding Payload Structures

3.1 Example Payload Structure

Every Payload structure description comes in following structure: example payload and then table with similar columns.

Example payload from/to fPort 50: 02 4A 10 0E 32

<table>
<tr><th>G</th><th>Byte</th><th>Type</th><th>Example</th><th>Parameter</th><th>Details</th><th>Example Value</th></tr>
<tr><td></td><td>0</td><td>uint8</td><td>0x02</td><td><i>_packet_type</i></td><td>0x02 - <i>configuration_packet</i></td><td><i>configuration_packet</i></td></tr>
<tr><td></td><td>1</td><td>bits8</td><td>0x4A</td><td><i>configured_parameters</i></td><td colspan="2">Example: 0x4A = b0100 10 1 0
<table>
<tr><th>Bit</th><th>Parameter</th><th>Example</th></tr>
<tr><td>0</td><td><i>secondary_interval_sent</i></td><td>0</td></tr>
<tr><td>1</td><td>RFU</td><td>1</td></tr>
<tr><td>2-3</td><td><i>some_2bit_parameter</i></td><td>b10 = 2</td></tr>
<tr><td>4-7</td><td><i>some_4bit_parameter</i></td><td>b0100 = 4</td></tr>
</table></td></tr>
<tr><td></td><td>2</td><td rowspan="2">int16</td><td>0x10</td><td rowspan="2"><i>main_interval__s</i></td><td rowspan="2">converter: *2</td><td rowspan="2">0x0E 10 = 3600 * 2 = 7200 (s)</td></tr>
<tr><td></td><td>3</td><td>0x0E</td></tr>
<tr><td></td><td>r0</td><td>uint8</td><td>0x32</td><td><i>secondary_interval</i></td><td>when: <i>configured_parameters:secondary_interval_sent</i>
0xFF - <i>not_configured</i></td><td>0x32 = 50</td></tr>
</table>

The payloads are described in top to bottom structure. Bitwise least significant bit comes first (lsb), bytewise least significat byte comes first (LSB).

Human readability of hex arrays introduces confusion into this. E.g. decimal value of 1000000 in hexadecimal is 0x000F4240 (LSB on the right) but in hexadecimal payload string it is usually printed LSB on the left 40 42 0F 00 .

Example payload hex has LSB on the left. Different parameter portions in hex are separated with whitespace. Packet fPort and direction (in reference to NAS module) are also signified with to (downlink) and from (uplink).

3.2 Payload Structure Columns Explanation

Column: G

Optional group column to help to identify which blocks are inseparable / grouped.

Column: Byte

Byte column expresses byte position. Always starting with 0 due to array first member being 0. The sequence is broken with the start of every optional block. From the first optional block onwards the positions of the bytes are relative, therefore e.g. r0, r1.

Column: Type

Type column determines encoding and length of the data.

Type	Length in bytes	Encoding
byte	1	does not matter
uint8 / int8	1	8 bit unsigned / signed integer
uint16 / int16	2	16 bit unsigned / signed integer
uint32 / int32	4	32 bit unsigned / signed integer
uint64 / int64	8	64 bit unsigned / signed integer
bits8 / bits16	1 / 2	8 bit / 16 bit bitfield (flags and/or decimals)
float	4	IEEE 754 floating point number
string	N	N bytes of string (byte array), encoding utf-8

Column: Example

Example bytes to help to understand how bytes from example payload are mapped. Hex values on Example column hex values can be matched top-down with Example payload hex left to right. Grey hex means that this value is constant.

Column: Parameter

parameter_name is descriptive name of parameter, formatted in *italic_snake_case*. They are used consistently throughout the documentation and intended to be used in parsers.
RFU - reserved for future use
grey_parameter - should be hidden in the end parser to declutter the results.
x_parameter_sent - sent keyword indicates that this bit controls following block denoted like when: *x_parameter_sent*.
main_interval__s - the unit is always behind double underscores '__'. Unit L/h would be __L_h. Unit °C would be '__C'.

Column: Details

defined_options or *special_values* are italic and indigo blue.
n/a - not available / not applicable

Under detailed bitfields tables Bit 0 is 0x01 and Bit 7 is 0x80.

Used Keywords:
when: defines the conditions when the block will be active. When shows the relationships between flags and existence of optional blocks.
converter: defines how to convert the value to extract the intended value by e.g. adding offset or multiplying with something.
formatter: defines how to format the value.

Column: Example Value

Shows how to convert the Example column's hex into useful output value.