SystemCORP ENERGY

Generic Payload

Functional Specification

Documentation Control

Author:	Nicholas Rixson
Version:	1.02
Version History:	1.00 – Initial Specification 1.01 – Change DF1.2 to apply the data object size in bytes, instead of number of objects. Add suitability tables on each data format. Add DF1.3. 1.02 – Add more examples and minor adjustments
Creation Date:	21 January 2020
Last Version Date:	4 March 2020
Product Reference:	**190-0013**
Document Status:	Released

Table of Contents

1 Introduction

The purpose of this document is to describe a generic data format payload which can be used across different media and protocols.

This payload is directed at serving (possibly independent) data objects, consisting of the data value, its quality indicating the validity and information about the data, and the timestamp typically of last change.

The initial application of this payload is directed for use in a generic LoRa payload.

1.1 Typical Use

The payload is typically carried by some media, protocol or internal serialised pipe, socket or API. The header and footer may be pre-defined by the protocol, or other use cases may affect what is required as to when different headers or footers are required. As such, this document described the internal payload, with the header and footer left to be decided as appropriate for the medium in which the payload is carried.

Carrier Header	**Payload**	**Carrier Footer**

Only the payload is defined within this document.

1.2 Definitions

The payload defines the application-level bytes carried by some media between applications.

Term	**Meaning**
Header	Information at the start of a section of payload, typically contains information about the data immediately following the header.
Footer	Data after the primary data payload, typically containing a checksum of data to detect corruption.
Value	the current or historical data value, such as a measurement or system information.
Quality	information about the value, such as its validity.
Timestamp	The time at which the value or quality is defined or changed. Typically, in live flow this refers to the timestamp of last change.
Object Address	A byte sequence to identify an object.

1.3 Abbreviations

Term	**Meaning**
DF	Data format; a method of encoding data objects into a payload
DO	Data object; an information reference typically containing value, quality and timestamp.
e.g.	Example
IP	Internet Protocol
LoRa	Long range low-power wide area network
TBD	To be determined
UDP	User Datagram Protocol
TCP	Transmission Control Protocol

2 Definitions

Defines the meaning and values of items which may be encoded.

2.1 Value definition

A value holds the primary measurement or information being expressed.

If a data type is encoded into:

- 4 bits, then there are 16 values available
- 5 bits, then there are 32 values available

Each value has a *data type*, defined as follows.

Data Type	Name	Description
0	Extended	Inspect packet further for details.
1	Boolean	Has two possible values. 0 (false, off) and 1 (true, on)
2	Dbpos	Has four possible values representing two contacts in a switch. Made of 2 bits, the 'off' bit and the 'on' bit. Binary 00 = INTERMEDIATE or in transition (touching neither contact) Binary 01 = OFF (touching only the 'off' contact) Binary 10 = ON (touching only the 'on' contact) Binary 11 = INVALID (touching both 'off' and 'on' contacts).
3	Int8	Signed 8-bit integer, -128 to 127
4	Int8u	Unsigned 8-bit integer, 0 to 255
5	Int16	Signed 16-bit integer
6	Int16u	Unsigned 16-bit integer
7	Int32	Signed 32-bit integer
8	Int32u	Unsigned 32-bit integer
9	Int64	Signed 64-bit integer
10	Int64u	Unsigned 64-bit integer
11	Float32	32-bit floating point, IEEE 754 single precision.
12	Float64	64-bit floating point, IEEE 754 double precision.
13	Octet-String	Sequence of bytes
14	Unicode-String	Sequence of UTF-8 bytes representing a string.
15	Bit-string	Sequence of bits
16..31	TBD	

Further data types are reserved may be assigned in the future.

2.2 Quality definition

Quality flag definitions are defined below. The overall quality consists of a combination of the individual quality flags, as defined below. The GOOD quality is just a name that indicates that all quality flags are 0, that is, no quality flags are set.

Flag	Value (Hex)	Short	Definition
GOOD	00 00 00	OK	Validity: no abnormal condition of the data acquisition is detected
INVALID	00 00 01	IV	Validity: value is invalid or unknown
QUESTIONABLE	00 00 02	Q	Validity: value is in doubt and should not be relied on

OVERFLOW	00 00 04	OV	Detail: value cannot be represented in the given format (should have INVALID validity)
OUTOFRANGE	00 00 08	OR	Detail: value is outside a predefined range
BAD_REFERENCE	00 00 10	BREF	Detail: value may not be correct due to a reference being out of calibration (e.g. in an Analog to Digital Converter a known input gives wrong output)
OSCILLATORY	00 00 20	OSC	Detail: value is currently oscillating (e.g. digital input continually changing 0/1). The data object value is not updated during this time
FAILURE	00 00 40	FAIL	Detail: value is INVALID due to a failure
OUT_DATED	00 00 80	OLD	Detail: value is out of date, it may have changed since the last notification (should have QUESTIONABLE validity)
INACCURATE	00 01 00	INAC	Detail: value does not meet the stated accuracy of the source (should have QUESTIONABLE validity)
INCONSISTENT	00 02 00	INC	Detail: an evaluation function has detected an inconsistency (should have QUESTIONABLE validity)
TRANSIENT	00 04 00	TR	Detail: Equipment is in a transient state. Used in step-position information
CARRY	00 08 00	CY	Detail: Carry indicates (e.g. counter) overflow occurs when the value goes over-bound and resets to zero
COUNTER_ADJUSTED	00 10 00	CA	Detail: Counter was adjusted since the last reading (e.g. initialised to a new value)
DERIVED	00 20 00	DER	Detail: The value is calculated, not a measurement
PROTOCOL_COMMUNICATION_LOST	00 40 00	PNC	Detail: Protocol updating the ADH data point has lost communication to the source
ADH_COMMUNICATION_LOST	00 80 00	ANC	Detail: Not connected to the ADH application source of the data point
…	01 00 00 to 10 00 00		Bits are reserved for future use
SUBSTITUTED	20 00 00	SB	Source: value, quality or time has been substituted
TEST	40 00 00	TEST	Test: test values should not be used for operational purposes
OPERATOR_BLOCKED	80 00 00	BL	Further update of the value has been blocked by an operator. The value shall be the same as it was before blocking

2.3 Timestamp definition

The timestamp is an absolute timestamp in seconds since Epoch, with an added sub-second component. It uses UTC (Universal co-ordinated time), not local time.

It is made up of

- Unix timestamp; seconds since Epoch (1970-01-01 00:00:00), 32-bit unsigned integer.
- Microseconds into this second, 20-bit ranged [0, 999 999].
    - 4 most significant bits reserved
    - 20 least significant bits used for microseconds

2.4 Object address definition

An object address is a byte-sequence which uniquely identifies an object at the communication endpoint. The use, size and specific meaning of this value is implementation-defined. The number of bytes in an object address must be known at configuration time of both communication endpoints, for correct encoding and decoding of packets.

Some Examples:

a) Some implementations might choose this to be a “1 byte logical address” followed by a “3 byte object number” that addresses objects within that logical address. This means the total object addressing scheme uses 4 bytes.
    a. [logical address (1 byte), object number (3 bytes)]
b) A 6 bytes of device address (or Ethernet MAC address) then 2 bytes of further object addressing.
    a. [Device hardware address (6 bytes), object number (2 bytes)]
c) It could encapsulate e.g. IEC 60870-5-104 APDUs by using the common address and information object addresses
    a. [common address/CASDU (2 bytes), Information object address (3 bytes)]
d) It could encapsulate or represent a message mirroring a DLMS-COSEM message:
    a. 6 bytes OBIS Code (A,B,C,D,E,F addressing), and 1 byte attribute/method index.

3 Value Encoding Formats

Values are encoded according to the data type as follows. The data types are defined in 2.1 Value definition.

3.1 Extended

Extended data types are to be defined by some other mechanism within the following bytes of the message. This is not yet used.

3.2 Boolean

A Boolean value is encoded in one byte. All bits at 0 mean false, and any other value is true.

3.3 Dbpos

Double positions are encoded in one byte, with the least significant two bits representing the state. Remaining bits are set to 0.

Binary 00 = 0 = INTERMEDIATE or in transition (touching neither contact)
Binary 01 = 1 = OFF (touching only the 'off' contact)
Binary 10 = 2 = ON (touching only the 'on' contact)
Binary 11 = 3 = INVALID (touching both 'off' and 'on' contacts).

Bit	7	6	5	4	3	2	1	0
Use							'on'	'off'

3.4 Integer types

Signed and unsigned integer types are encoded in twos-complement Big-Endian format.

- Int*[n]* - indicates a signed integer of *[n]* bits in size.
- Int*[n]*u - indicates an unsigned integer of *[n]* bits in size.

3.5 Float types

Float32 and Float64 floating point types are in the packet in IEEE 754 floating point formats, with the bit-string sent in the order as the data type described by IEEE. That is:

- Sign bit
- Exponent
- Fraction

Implementation note: Internally, many processors treat the bytes like an integer, with the bytes swapped according to the endian-ness of the processor. In this case, `htonl()` is performed on these bytes just as it is on integer types.

3.6 Octet string and Unicode string

Octet strings and Unicode strings are treated the same way.

1. The length in bytes is encoded as per 3.11 VAU: Variable length unsigned integer
2. The bytes are appended.

3.7 Bit string

Bit strings are encoded as:

1. The length in **bits** as per 3.11 VAU: Variable length unsigned integer
2. The bytes containing the bit string. Any remaining bits are padded with 0 in the least significant bits of the final byte.

Example:

Bit-string to encode	Length {byte encoding}	**Encoded bytes**		
		Byte 1	Byte 2	Byte 3
101	3 {03}	1010 0000		
1100 1110	8 {08}	1100 1110		
1111 1001 001	11 {0B}	1111 1001	0010 0000	

3.8 Quality

Quality is encoded as 3 bytes directly.

For example, to encode INVALID and FAILURE:

Quality
00 00 41

3.9 Timestamp

The timestamp is encoded as:

- 32-bit unsigned, Unix timestamp in seconds since Epoch UTC.
- 4-bits reserved (set to 0).
- 20-bit microseconds into the second.

Bytes	1	2	3	4	5	6	7
Encoded	Seconds since Epoch, Int32u				Res	Microseconds, Int20u	

3.10 Message Type

The message type indicates the kind of operation the message represents. This is included before data objects definitions to determine how the information should be processed.

The *message type value* is defined as follows.

Value	**Name**	**Description**
0	Info	Information about the current or historical values of objects
1	Select	Check an operation and reserve it ready to be operated
2	Operate	Control the output
3	Cancel	Cancel the reservation made by *Select*.

The message type also contains:

- RESP – A flag to indicate this is a **response**.
- ACK – an acknowledgement flag to indicate a **successful** response.

The message type is encoded in one byte, with the least significant bits representing the *message type value*, and remaining bits representing flags. Unused values are reserved and are set to 0 when encoding.

Bit	7	6	5	4	3	2	1	0
Use	RESP	ACK				*value*		

3.11 VAU: Variable length unsigned integer

This can be used to encode unsigned integers, without knowing its length ahead of time. Each byte uses the lowest 7 bits as part of the length value, and the highest bit as an indicator whether there are more bytes defining the length to follow (1 if more bytes, and 0 if this is the final byte). The final byte contains the least significant bits.

Examples when encoding values of different sizes:

0-7 bits value	**0**xxx xxxx (LSB)			
8-14 bits	**1**xxx xxxx	**0**xxx xxxx (LSB)		
15-21 bits	**1**xxx xxxx	**1**xxx xxxx	**0**xxx xxxx (LSB)	
22-29 bits	**1**xxx xxxx	**1**xxx xxxx	**1**xxx xxxx	**0**xxx xxxx (LSB)

4 Payload Encoding Formats

Multiple payload formats are defined. One is chosen according to the requirements of the task. The data format must be known at configuration time at both endpoints to correctly encode and decode packets.

4.1 Data object format

Data object “DO”

- Data object header, made up of:
  - Object address
  - Presence flags (3 bits)
    - Value present 0x80
    - Quality present 0x40
    - Time present 0x20
  - Data Type (5 bits) [0x00, 0x1F]
- Value (if present)
- Quality (if present)
- Timestamp (if present)

Data Object

Object address	DO Header type byte		Value (if present)	Quality (if present)	Timestamp (if present)
--- size according to system ---	Pres.	Data type	Size according to data type	3 bytes	7 bytes

The data type can be ignored if the “value present” flag is 0.

4.2 DF1.1

The payload consists of a message type followed by a sequence of independent data objects.

Payload

Message Type	DO1	DO2	… DO [n]

Suitability

Protocol	Suitable?
Message oriented (e.g. UDP/IP)	Yes
Stream oriented (e.g. TCP/IP)	No

4.3 DF1.2

The payload consists of one or more “sections”, where each section contains a message type, followed by the ‘data size’ in bytes of the remaining data objects, and then the data objects.

Section header	DO1	… DO [n]	Section header	DO1	… DO[n]	… Section [m]

Section header:

- Message Type (1 byte)
- Number of bytes in the following data objects of the section encoded as per 3.11 VAU: Variable length unsigned integer

These sections can be ‘chained’ one after the other, if required. In stream-based protocols, the message chain is indistinguishable from independent messages.

Suitability

Protocol	Suitable?
Message oriented (e.g. UDP/IP)	Yes
Stream oriented (e.g. TCP/IP)	Yes

4.4 DF1.3

The payload consists of independent sections containing a single data object each, with the data object length always encoded. This has the advantage that data objects whose type or size that are not understood by the parser do not affect the parser’s ability to decode data objects which come after the unknown data object.

Payload

Section 1			… Section [N]
Section Header			
Message Type	Size in bytes	DO1	

Section header:

- Message Type (1 byte)
- Number of bytes in the following data object encoded as per 3.11 VAU: Variable length unsigned integer

Suitability

Protocol	Suitable?
Message oriented (e.g. UDP/IP)	Yes
Stream oriented (e.g. TCP/IP)	Yes

5 Interoperability

System settings	Value chosen	Notes
System object address size		0 or greater
System data format		e.g. DF1.1

Object address encoding rules, if any:

Other notes:

Pre-defined object address meanings:

6 Appendix

6.1 Examples

6.1.1 Example 1, DF1.1

System settings	Value chosen
System object address size	2
System data format	DF1.1

Object address encoding rules, if any:
Object address is made up of:

- Object number (2 bytes, int16)

Pre-defined object address meanings:

Object number	Data Type	Meaning
100	Boolean	System health
101	Int8u	Battery percentage [0 - 100 %]
102	Int8	Temperature [°C]
200	Boolean	Moisture detected (1=yes, 0=no)

The payload will send an *information* message, telling the remote side its

- Battery percentage is 75%
- Temperature is 24°C
- Moisture is not detected in its vicinity.

Then the payload is encoded as follows:
DF1.1 Payload

Message Type	DO1	DO2	… DO [n]

Message type “information (0)”, it is not a response (RESP=0), nor an acknowledgement (ACK=0).

- Message Type = 00 (hex).

DO1: The battery percentage is 75%. Include value, quality and time.

- Object address = Object number = 101, encoded into the object address size of 2 bytes
  - 0x0065
- Presence flags = (0x80 + 0x40 + 0x20) = 0xE0
- Data type = Int8u = 4
  - First byte is 0xE4
- Value is present, so must be encoded.
  - 75% = 75dec = 0x4B
- Quality is included, so must be encoded
  - Quality = GOOD = 00 00 00.
- Timestamp is included
  - Let’s say the date is 2020-01-01 at 10:00 and 42ms.
  - Epoch seconds = 1577872800 = 5E0C6DA0
  - Microseconds into second = 42ms = 42000us = 0x00A410

DO1:

Object address	DO Header type byte		Value (if present)	Quality (if present)	Timestamp (if present)
--- size according to system ---	Pres.	Data type	Size according to data type	3 bytes	7 bytes
Dec(101)	0xE0	0x04	Dec(75)	3 bytes	7 bytes (4 + 3)
00 65	E4		4B	00 00 00	5E 0C 6D A0 00 A4 10

DO2: The temperature is 24°C. Include value, no quality or time.

- Object address = Object number = 102, encoded into the object address size of 2 bytes
  - 0x0065
- Presence flags = (0x80) = 0x80
- Data type = Int8 = 3
  - First byte is 0x83
- Value is present, so must be encoded.
  - 24°C = 24dec = 0x18
- Quality and time are not included in the message.

DO2:

Object address	DO Header type byte		Value (if present)	Quality (if present)	Timestamp (if present)
Dec(102)	0x80	0x03	Dec(24)	Not present	Not present
00 66	83		0x18		

DO3: Moisture is not detected in the sensor's vicinity. Include value, no quality or time.

Object address	DO Header type byte		Value (if present)	Quality (if present)	Timestamp (if present)
Dec(200)	0x80	0x01	Dec(0)	Not present	Not present
00 C8	81		0x00		

The assembled payload is

Message Type	DO1	DO2	DO3
00	00 65 E4 4B 00 00 00 5E 0C 6D A0 00 A4 10	00 66 83 18	00 C8 81 00

Payload: 00 <u>00 65 E4 4B 00 00 00 5E 0C 6D A0 00 A4 10</u> 00 66 83 18 <u>00 C8 81 00</u>
The underlines are showing the separated data objects in the packet.

6.1.2 Example 2, DF1.2

System settings	Value chosen
System object address size	3
System data format	DF1.2

Object address encoding rules, if any:
Object address is made up of:

- Logical address (1 byte)
- Object number (2 bytes, int16)

Pre-defined object address meanings:

Object number	Data Type	Meaning
100	Dbpos	Circuit breaker position
101	Dbpos	Earth switch position
200	Int32	Real power flow [Watts]
201	Int32	Reactive power flow [VAr]
202	Int32	Apparent power flow [VA]

The payload will send an *information* message, telling the remote side its

- Logical address 1 circuit breaker position is Closed
- Logical address 1 earth switch is Open
- Logical address 2 real power flow of 42kW.

Then the payload is encoded as follows:

DF1.2 Payload

Section header	DO1	… DO [n]	Section header	DO1	… DO[n]	… Section [m]

Section header:

- Message Type (1 byte)
- Number of bytes in the following data objects of the section encoded as per 3.11 VAU: Variable length unsigned integer

➢ The message type is information (0), with no response or ACK flags (0).
➢ The number of bytes in the following objects will be determined after the DOs are encoded.

DO1:
Logical address 1, circuit breaker closed.

Object address	DO Header type byte		Value (if present)	Quality (if present)	Timestamp (if present)
--- size according to system ---	Pres.	Data type	Size according to data type	3 bytes	7 bytes
Logical address, object number	Value	Dbpos	Closed (2)	Not included	Not included
L=1, obj=100	0x80	2			
01, 00 64	82		02		

DO2:
Logical address 1, earth switch open
DO2 has the same message type, “information” with no extra flags, so it can be encoded into the same section.

Object address	DO Header type byte		Value (if present)	Quality (if present)	Timestamp (if present)
--- size according to system ---	Pres.	Data type	Size according to data type	3 bytes	7 bytes
Logical address, object number	Value	Dbpos	Open (1)	Not included	Not included

L=1, obj=101	0x80	2			
01, 00 65	82		01		

DO3:
Logical address 2, real power flow of 42kW.
DO3 also has the same message type, “information” with no extra flags, so it can be encoded into the same section. 42kW is translated into Watts, for a value of 42000.

Object address	**DO Header type byte**		**Value (if present)**	**Quality (if present)**	**Timestamp (if present)**
--- size according to system ---	Pres.	Data type	Size according to data type	3 bytes	7 bytes
Logical address, object number	Value Time	Int32	42000	Not included	2020-01-01 at 10:00 and 42ms. Epoch seconds = 1577872800 = 5E0C6DA0 Microseconds into second = 42ms = 42000us = 0x00A410
L=2, obj=200	0x80 0x20	7	0x 00 00 A4 10		
02, 00 C8	A7		00 00 A4 10		5E 0C 6D A0 00 A4 10

Then we build the complete payload:

Section header	DO1	DO2	DO3
00, length of DOs	01 00 64, 82, 02	01 00 65, 82, 01	02 00 C8, A7, 00 00 A4 10, 5E 0C 6D A0 00 A4 10
	Length = 5	Length = 5	Length = 15
	Length = 25dec = 0x19		
00 19	01 00 64 82 02 01 00 65 82 01 02 00 C8 A7 00 00 A4 10 5E 0C 6D A0 00 A4 10		

The total payload is:
00 19 01 00 64 82 02 01 00 65 82 01 02 00 C8 A7 00 00 A4 10 5E 0C 6D A0 00 A4 10

6.1.3 Example 3, DF1.3

System settings	Value chosen
System object address size	1
System data format	DF1.3

Object address encoding rules, if any:
Object address is made up of:

- Object number (1 byte, Int8u)

Pre-defined object address meanings:

Object number	Data Type	Meaning
0	Unicode-String	Device's name, description or location
1	Bit-string	Digital input states, where each bit in the string represents the value of one input
2	Bit-string	Digital output states, where each bit in the string represents the value of one digital output. Can be used with information and control.

The payload will send an *information* message, telling the remote side its

- Device name – "Valve00192"
- State of its digital inputs (there are 12)
- State of its digital outputs (there are 6)

Then the payload is encoded as follows:

DF1.3 Payload

Section 1			… Section [N]
Section Header			
Message Type	Size in bytes	DO1	

Section header:

- Message Type (1 byte)
- Number of bytes in the following data object of the section encoded as per 3.11 VAU: Variable length unsigned integer

In DF1.3, each data object is separately encoded into different sections in the message.

Section 1, containing DO1:

- The message type is information (0), with no response or ACK flags (0).
- The number of bytes in the following objects will be determined after the DO is encoded.

DO1:
Device name "Valve00192". Value only.

Object address	DO Header type byte		Value (if present)	Quality (if present)	Timestamp (if present)
--- size according to system ---	Pres.	Data type	Size according to data type	3 bytes	7 bytes
Object number	Value	Unicode-string	Valve00192	Not included	Not included
0	0x80	14dec = 0x0E	Length (10dec = 0x0A hex), followed by Decimal 'V' 86 'a' 97 'l' 108 'v' 118 'e' 101 '0' 48 '0' 48 '1' 49 '9' 57 '2' 50		
00	8E		0A 56 61 6C 76 65 30 30 31 39 32		

The total DO1 size is 2 + 11 = 13 bytes. So the section 1 header must then encode size-in-bytes=13.

Section 1		
Section Header		
Message Type	Size in bytes	DO1
00	13dec = 0x0D hex	00 8E 0A 56 61 6C 76 65 30 30 31 39 32

Total section 1 payload:
00 0D 00 8E 0A 56 61 6C 76 65 30 30 31 39 32

Section 2, containing DO2:

- The message type is information (0), with no response or ACK flags (0).
- The number of bytes in the following objects will be determined after the DO is encoded.

DO2:
State of digital inputs (12 bits). Value only.
Let's say the state of the inputs is

Input	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Value	0	0	0	1	0	0	0	0	1	1	1	0

Bit-string to encode	Length {byte encoding}	**Encoded bytes**	
		Byte 1	Byte 2
0001 0000 1110	12 {0C}	**0001 0000**	**1110** 0000
	0x0C	**0x10**	**0xE0**

Object address	**DO Header type byte**		**Value (if present)**	**Quality (if present)**	**Timestamp (if present)**
--- size according to system ---	Pres.	Data type	Size according to data type	3 bytes	7 bytes
Object number	Value	Bit-string	10 E0	Not included	Not included
1	0x80	15dec = 0x0F			
01	8F		0C 10 E0		

The total DO2 size is 5 bytes. So the section is then

Section 2		
Section Header		
Message Type	Size in bytes	DO1
00	5dec = 0x05 hex	01 8F 0C 10 E0

Total section 2 payload:
00 05 01 8F 0C 10 E0

Section 3, containing DO3:

- The message type is information (0), with no response or ACK flags (0).
- The number of bytes in the following objects will be determined after the DO is encoded.

DO2:
State of digital outputs (6 bits). Value only.
Let's say the state of the inputs is

Input	**1**	**2**	**3**	**4**	**5**	**6**
Value	0	0	1	1	0	1

Bit-string to encode	Length {byte encoding}	Encoded bytes
		Byte 1
0011 01	6 {06}	**0011 01**00
	0x06	**0x34**

Object address	DO Header type byte		Value (if present)	Quality (if present)	Timestamp (if present)
--- size according to system ---	Pres.	Data type	Size according to data type	3 bytes	7 bytes
Object number	Value	Bit-string	Valve00192	Not included	Not included
2	0x80	15dec = 0x0F			
02	8F		06 34		

The total DO3 size is 4 bytes. So the section is then

Section 3		
Section Header		
Message Type	Size in bytes	DO1
00	4dec = 0x04 hex	02 8F 06 34

Total section 3 payload:
00 04 02 8F 06 34

Then the complete payload is the combination of each section.

00 0D 00 8E 0A 56 61 6C 76 65 30 30 31 39 32 00 05 01 8F 0C 10 E0 00 04 02 8F 06 34