VSEW_mk4_MQTT

MQTT Protocol

Feb 20 2023

Bruno Paillard

1 Revision History

1. July 15 2023 Initial version of this document.

2 Introduction

The *VSEW_mk4_MQTT* is a new variant of the *VSEW_mk4* series that introduces an MQTT communication protocol. That means that the instrument can communicate with MQTT brokers and report to MQTT-based platforms.

3 Endianness

Unless otherwise noted, the endianness is Little-Endian.

4 Basic Types

The following basic types may be used in this protocol:

Type Name	Description	Endianness
U8	Single byte unsigned	N/A
U16	16-bit word unsigned	Little-Endian
U32	32-bit word unsigned	Little-Endian
U64	64-bit word unsigned	Little-Endian
I8	Single byte signed	N/A
I16	16-bit word signed	Little-Endian
I32	32-bit word signed	Little-Endian
I64	64-bit word signed	Little-Endian
Sgl	32-bit word in IEEE 754 floating point format	Little-Endian
Dbl	64-bit word in IEEE 754 floating point format	Little-Endian
String	Strings are concatenations of 8-bit ASCII characters, terminated by an end-of-text (0x00) byte.	N/A

Table 1

5 Connection

The instrument always connects to the broker with a clean session. It keeps track of what data has already been published, so it will only publish whatever data has not yet been acknowledged by the broker.

The instrument always publishes and subscribes with a QoS of 1. In case the broker has received some of the recorded data, but has failed to acknowledge it, the same data will be sent again (duplicated) at

the next connection opportunity, but that should not matter because it will be the same data already published. Each message includes all the information needed to localize the data in time.

If the instrument loses power or is manually reset, it will lose track of what it has already published. In that case it will proceed to re-publish all the data in its memory. Again, that should not matter because the data is published together with information needed to localize it in time.

Since every session is clean, the instrument will re-subscribe to all subscribed messages. That includes the settings. This is done in the first few seconds of the connection. At that time the broker should send the current settings, which must be published with a *retain* flag at 1.

The instrument will only reprogram the settings and restart a recording if the new settings are different than the settings already in effect. If they are identical to the settings in effect no action will be taken.

6 Message formats

6.1 Topics

The *VSEW_mk4_MQTT* has two MQTT modes of operation:

1. **Forced Topics**: Pub and Sub topics can be defined by the user. In this mode there is only one Pub topic and only one Sub topic for all types of messages. Both the Pub and Sub topic strings can be defined freely and written into Flash by the user using *Instrument Manager*. This is to comply with some MQTT broker platforms that impose the topic strings to the client. In this case, the user must rely on the first 8 bytes of the message content to determine the precise type of message being transmitted/received.

2. **Standard Topics**: Pub and Sub topics are defined by the instrument. In this (more conventional) mode, the Pub and Sub topic strings are defined for each type of message, and indicate explicitly that type. There are as many topic strings as there are types of messages. The topic string for each type of message is indicated in the document below. This mode is selected when the *Forced* Pub and/or Sub topic strings are left empty. In this case the first 8 bytes of the message content still indicate the precise type of the message being transmitted/received, but these 8 bytes can be ignored/zero-padded, since they are redundant with the information included in the topic.

Note that the mode of operation can be mixed (*Forced* on the Pub topics and *Standard* on the Sub topics, or vice-versa).

6.2 Message types

In both modes of operation, the beginning of every message is comprised of two U32 numbers that indicate the message type and structure:

- ***Model/Format*:** This field indicates:
    - **Model**: The model of the instrument is indicated in the 3 lower bytes. For the VSEW_mk4_MQTT instrument, those three bytes are 0x34, 0x53, 0x56
    - **FW Version**: The firmware version is indicated by the upper byte. The major revision is indicated by the upper 4 bits, and the minor version is indicated by the lower 4 bits. At the time of this writing, the firmware version is 1.2. So the upper byte is 0x12

Implicitly those 4 bytes indicate the nature of the instrument that is connecting to the broker, and how the rest of the message should be interpreted. For the VSEW_mk4 FW 1.2, the value of that field is equal to 0x12345356.

- ***Type*:** This indicates the type of message that is published.

The value of *Type* completely describes how the rest of the message can be interpreted.

For the *VSEW_mk4_MQTT*, the *Type* field can have the following values:

Type (Hex)	Function
0x0000000A	Vital Signs
0x0000000F	Settings
0x00000020	Data

Table 2

Note: On messages that the instrument publishes, when the mode of operation is *Standard Topics*, those two U32 numbers can be ignored by the recipient since the same information is provided explicitly in the topic.

Note: On messages that the instrument subscribes to, when the mode of operation is *Standard Topics*, those two U32 numbers can be set to any value. The instrument ignores them and relies on the received topic to determine the nature of the message that is being received.

6.3 Published messages

6.3.1 Vital Signs

Topic (*Standard* mode): Inst_Class/*Model*/FW*xx*/*Client_ID*/Vitals

- **Inst_Class**: "VS" for Vibration Sentry series
- ***Model***: "VSEW_mk4_MQTT" for this instrument.
- ***FWxx***: Two-digit number that indicates the firmware revision (for instance "FW12" indicates firmware version 1.2)
- ***Client_ID***: Client ID string as programmed by the user. This defaults to the serial number of the instrument if the string is empty.

This is a message that is published every time the instrument connects. It contains all the vital signs of the instrument, as well as the instrument UTC and clock error.

This message is published with a retain flag at 1.

Field	Type	Size (bytes)	Value	Function	Scale
Model/Format	U32	4	*NSRTW_mk4->* 0xMm34534E *VSEW_mk4->* 0xMm345356	4 bytes that identify the model and firmware revision. "M" symbolizes the Major rev number. "m" symbolizes the minor rev number. Presently 2 models are recognized: NS4 NSRTW_*mk4* VS4 *VSEW_mk4*	
Type	U32	4	0x0000000A	Indicates a message of type "Vital Signs"	
UTC[1]	U64	8		$UTC_{instrument}$. This represents the instrument's internal clock	seconds
UTC_err	I32	4		This represents the difference between a reference UTC from an SNTP server, and the instrument UTC: $UTC_{SNTP} - UTC_{instrument}$. It is only updated when the SNTP time is accessible. Otherwise, the last updated value is transmitted.	seconds
Batt	Sgl	4		Battery voltage	Volt
Temp	Sgl	4		Temperature	degC
RSSI	Sgl	4		WiFi signal strength	dBm

Table 3

6.3.1.1 Use of the UTC_Err value

The instrument will attempt to connect to an SNTP server and get a precise UTC every time it connects to the internet. That exact UTC is used to reset the internal instrument clock when a recording is started. But the instrument will not set its internal clock while it is recording to avoid any discontinuity. Since a recording can last for several weeks, the drift of its internal clock can grow to large values. Instead, the instrument will transmit the difference between the SNTP's reference UTC and its internal clock's UTC at the time of the connection. This way the graphs can be precisely time-aligned if required. Ultimately it is the responsibility of the consumer of the information to do something with that clock error, or not.

6.3.2 Recorded Data

Topic (*Standard* mode): Inst_Class/*Model*/*FWxx*/*Client_ID*/*Data*

- **Inst_Class**: "VS" for Vibration Sentry series
- ***Model***: VSEW_mk4_MQTT for this instrument.

[1] UTCs (Universal Time Codes) are referenced to January 1st 1904. To get a Unix UTC, subtract 2082844800 seconds.

- ***FWxx***: Two-digit number that indicates the firmware revision (for instance "FW12" indicates firmware version 1.2)
- ***Client_ID***: Client ID string as programmed by the user. This defaults to the serial number of the instrument if the string is empty.

6.3.2.1 Records, Frames, Messages and Values

The instrument can be set to record specific *Values*. For instance, these might be RMS values *Xmax*, *Xavg*, *Xmin*, or signal peak values *Xmax*, *Xmin*, *Ymax*, *Ymin*, *Zmax* and *Zmin*, or raw signals X, Y and Z… etc.

A *Record* is defined as a succession of values between the time a recording is started and stopped.

The instrument's memory can contain more than one record. This is the case when raw signals are being recorded in *AutoRec* mode.

A *Frame* is defined as a group of values pertaining to the same recorded time. For instance, if the instrument is set to record the following RMS values: Xmax, Xmin, YMax, YAvg, YMin and Zmax, a frame at frame No "i" consists of the following values in sequence:

[Xmax(i) – Xmin(i) – Ymax(i) – Yavg(i) – Ymin(i) – Zmax(i)].

Successive *Frames* are sent in *Data* messages.

A *Data* message typically contains several successive frames:

[Xmax(i) – Xmin(i) – Ymax(i) – Yavg(i) – Ymin(i) – Zmax(i)]

[Xmax(i+1) – Xmin(i+1) – Ymax(i+1) – Yavg(i+1) – Ymin(i+1) – Zmax(i+1)]

[Xmax(i+2) – Xmin(i+2) – Ymax(i+2) – Yavg(i+2) – Ymin(i+2) – Zmax(i+2)]… etc.

The size of each message is limited to 1kB. So several messages are sent in succession until there is no more data to send in the instrument's memory.

Data messages are published every time the instrument connects if some of the values in memory have not yet been sent to (and acknowledged by) the broker. Each message contains an integer number of full *Frames*, as well as the fractional UTC of the first frame in the current record, the frame number of the first frame in the current message and the settings in effect for the current record.

When a new record is transmitted there is usually a time discontinuity between the last message (containing the last frames of the previous record), and the next message (containing the first frames of the new record). A value of fractional UTC different from the previous message indicates that a new record is being transmitted.

The display time t_i for each frame transmitted in a *Data* message can be calculated from:

- The fractional UTC of the first frame in the current record: ***f_UTC*** (in multiples of 1/8s). That represents the time of the first frame in the current record.

- The frame number: ***N_Frame***. That is the frame number of the first frame in the current message, counted from the beginning (frame 0) of the current record.

- The interval between frames: ***Interval*** (in seconds). That is the time between two successive frames.

$$t_i = \frac{f\_UTC}{8} + \left((N\_Frame + i)\ x\ Interval\right)$$

Where:

i is the frame number of the given frame in the message, counted from the first frame in the current message.

N_Frame is the frame number of the first frame in the current message, counted from the beginning (frame 0) of the current record. *N_Frame* of the current message is greater than *N_Frame* of the previous message by exactly the number of frames contained in the previous message.

f_UTC is the fractional UTC of the first frame in the current record. *f_UTC* stays the same over successive messages pertaining to the same record. When a new record is transmitted, *f_UTC* changes to reflect the time of the first frame in the new record.

Interval is the time between two successive frames in the record. *Interval* stays the same over successive messages pertaining to the same record.

t_i is the time (in seconds) of frame No *i* in the current message.

There is a maximum of 256 values (1024 bytes) in each message. Messages are sent in succession, as long as there is still data to be transmitted in the instrument's memory.

Data messages are published with a retain flag at 1. So any subscriber always receives the last data transmitted by the instrument as soon as it subscribes.

Field	Type	Size (Bytes)	Value	Function	Scale
Model/Format	U32	4	*NSRTW_mk4->* 0xMm34534E *VSEW_mk4->* 0xMm345356	4 bytes that identify the model and firmware revision. "M" symbolizes the major rev number. "m" symbolizes the minor rev number. Presently 2 models are recognized: NS4 NSRTW_*mk4* VS4 *VSEW_mk4*	
Type	U32	4	0x00000020	Indicates a message of type *Data*	
f_UTC	U64	8		This represents the fractional UTC of the first data point in the current record	1/8 s (125 ms)
N_Frame	U32	4		This represents the frame number of the first frame in the current message	

Interval	Sgl	4		This represents the time (in seconds) between two successive frames in the data stream that follows	s
Fs	U16	2		Sampling Frequency	Hz
Manifest	U16	2		This value indicates the type of signal (acceleration or velocity), the type of values (raw signals, signal peaks and average, or RMS peaks and average), as well as the contents of the message (see below)	
High-Pass Filter	Sgl	4		High-Pass filter frequency (a value of zero indicates the high-pass filter is not in effect).	Hz
Low-Pass Filter	Sgl	4		Low-Pass filter frequency (a value of zero indicates the low-pass filter is not in effect).	Hz
KBF Filter	Sgl	4		KBF filter frequency (a value of zero indicates the KBF filter is not in effect).	Hz
Tau	Sgl	4		Time constant	seconds
N_Values	U32	4		Nb of following values	
$Value_1$	Sgl	4			m/s^2 for acceleration m/s for velocity Except for RMS values (see below)
$Value_2$	Sgl	4			m/s^2 for acceleration m/s for velocity Except for RMS values (see below)
…	Sgl	4			m/s^2 for acceleration

					m/s for velocity Except for RMS values (see below)
Value$_{N_Values}$	Sgl	4			m/s^2 for acceleration m/s for velocity Except for RMS values (see below)

Table 4

Note: The scale of values is m/s for velocity and m/s^2 for acceleration, except for RMS values, where it is in dB-m/s for velocity, with a 0 dB reference of 1 m/s, and dB-m/s^2 for acceleration, with a 0 dB reference of 1 m/s^2.

Take the following to get the RMS value in m/s or m/s^2: $X_{RMS} = 10^{Value/20}$

6.3.2.2 How to interpret Manifest and the contents of the message

Manifest is a 16-bit (U16) word indicating what type of data was recorded:

Bits 14 and 15 define the type of data that is transmitted.

F1 (Bit 15)	F0 (Bit 14)	Data type
0	0	RMS Levels
0	1	Signal Pks & Avg
1	0	Raw Signals
1	1	Reserved

Table 5 Data type

Bit 13 indicates the type of signal (acceleration or velocity) being recorded.

Bit 13	Signal type
0	Acceleration
1	Velocity

Table 6 Signal type

The data, and the lower bytes of *Manifest* are interpreted differently depending on the values of *F0* and *F1*.

6.3.2.2.1 RMS Min, Max and Average Levels

Bit_15	Bit_14	Bit_13	Bit_12	Bit_11	Bit_10	Bit_9	Bit_8
F_1 = 0	F_0 = 0	SigType	0	0	0	0	Z-min

Bit_7	Bit_6	Bit_5	Bit_4	Bit_3	Bit_2	Bit_1	Bit_0
Z-av	Z-max	Y-min	Y-av	Y-max	X-min	X-av	X-max

Table 7 RMS Levels Record Manifest

The record manifest contains 9 bits that indicate the data contents of the record.

- **X-max:** Indicates if the x-axis RMS-max value is present (1) or absent (0).
- **X-av:** Indicates if the x-axis RMS-average value is present (1) or absent (0).
- **X-min:** Indicates if the x-axis RMS-min value is present (1) or absent (0).
- **Y-max:** Indicates if the y-axis RMS-max value is present (1) or absent (0).
- **Y-av:** Indicates if the y-axis RMS-average value is present (1) or absent (0).
- **Y-min:** Indicates if the y-axis RMS-min value is present (1) or absent (0).
- **Z-max:** Indicates if the z-axis RMS-max value is present (1) or absent (0).
- **Z-av:** Indicates if the z-axis RMS-average value is present (1) or absent (0).
- **Z-min:** Indicates if the z-axis RMS-min value is present (1) or absent (0).

6.3.2.2.2 Signal Peaks and Averages

Bit_15	Bit_14	Bit_13	Bit_12	Bit_11	Bit_10	Bit_9	Bit_8
F_1 = 0	F_0 = 1	SigType	0	0	0	0	Z-min

Bit_7	Bit_6	Bit_5	Bit_4	Bit_3	Bit_2	Bit_1	Bit_0
Z-av	Z-max	Y-min	Y-av	Y-max	X-min	X-av	X-max

Table 8 Signal Statistics Record Manifest

The record manifest contains 9 bits that indicate the data contents of the record.

- **X-max:** Indicates if the x-axis max value is present (1) or absent (0).

- **X-av:** Indicates if the x-axis average value is present (1) or absent (0).
- **X-min:** Indicates if the x-axis min value is present (1) or absent (0).
- **Y-max:** Indicates if the y-axis max value is present (1) or absent (0).
- **Y-av:** Indicates if the y-axis average value is present (1) or absent (0).
- **Y-min:** Indicates if the y-axis min value is present (1) or absent (0).
- **Z-max:** Indicates if the z-axis max value is present (1) or absent (0).
- **Z-av:** Indicates if the z-axis average value is present (1) or absent (0).
- **Z-min:** Indicates if the z-axis min value is present (1) or absent (0).

6.3.2.2.3 Raw Signals

Bit_15	Bit_14	Bit_13	Bit_12	Bit_11	Bit_10	Bit_9	Bit_8
F_1 = 1	F_0 = 0	SigType	0	0	0	0	0

Bit_7	Bit_6	Bit_5	Bit_4	Bit_3	Bit_2	Bit_1	Bit_0
0	0	0	0	0	Z	Y	X

Table 9 Raw Signal Record Manifest

The record manifest contains 3 bits that indicate the data contents of the record.

- **X** Indicates if the x-axis signal is present (1) or absent (0) in the record.
- **Y** Indicates if the y-axis signal is present (1) or absent (0) in the record.
- **Z** Indicates if the z-axis signal is present (1) or absent (0) in the record.

6.3.2.2.4 Ordering of the values in the *Data* message

Values in each frame of the message are ordered from lower to higher bit position in the *Manifest*. For instance, if *Manifest = 1010000000000111* (indicating raw velocity signals X, Y and Z), then the values are ordered as follows in the message:

[X-Y-Z] [X-Y-Z] [X-Y-Z] [X-Y-Z]… etc

If *Manifest = 0100000101101101* (indicating Signals peaks and averages, Xmax, Xmin, Ymax, Ymin, Zmax, Zmin), then the values are ordered as follows in the message:

[Xmax-Xmin-Ymax-Ymin-Zmax-Zmin] [Xmax-Xmin-Ymax-Ymin-Zmax-Zmin] [Xmax-Xmin-Ymax-Ymin-Zmax-Zmin]… etc.

6.4 Subscribed Messages

Since the instrument opens a clean session every time it connects, it will re-subscribe to all subscribed messages whenever it connects.

6.4.1 Settings

Topic (*Standard* mode): Inst_Class/*Model*/*FWxx*/*Client_ID*/Settings

- **Inst_Class**: "VS" for Noise Sentry series
- ***Model***: VSEW_mk4_MQTT for this instrument.

- ***FWxx***: Two-digit number that indicates the firmware revision for which the message is published (for instance “FW12” indicates version 1.2).
- ***Client_ID***: Client ID string as programmed by the user. This defaults to the serial number of the instrument if the string is empty.

The following settings cannot be changed with that command. They must be set with the Instrument Manager application.

- Signal mode (acceleration/velocity)
- Filters settings
- Sampling frequency

Note: When *Raw Signals* are selected in Manifest, the recording mode is forced to *AutoRec*. In *RMS* and *Signals Peaks and Averages* mode, the recording mode is forced to *Continuous Recording*.

Note: In *Standard Topics* mode the instrument will not accept any *Settings* message that is published with a FW value that is different from its actual firmware revision. For instance, an instrument that has a firmware version of 1.1 will not accept a *Settings* message published with *FW10* FW value. This behavior is slightly different from the behavior in *Forced Topics* mode, where the instrument will accept any *Settings* message published for a firmware revision number *Mm* lower or equal to the instrument’s actual firmware revision (see below).

Field	Type	Size (bytes)	Value	Function	Scale
Model/Format	U32	4	*NSRTW_mk4->* 0xMm34534E *VSEW_mk4->* 0xMm345356	4 bytes that identify the model and firmware revision. “M” symbolizes the major rev number. “m” symbolizes the minor rev number. The revision number indicates the minimum revision number that should be accepted by the instrument (i.e. an instrument which firmware revision is lower than this number should refuse the settings because there is a chance that some of the settings or format are unknown to that firmware). Presently 2 models are recognized: NS4 NSRTW_*mk4* VS4 *VSEW_mk4*	

				In the Standard Topic mode of operation, these 4 bytes are "don't care". The information is taken from the Sub topic.	
Type	U32	4	0x0000000F	Indicates a message of type "Settings"	
Timezone	I32	4		This represents the time zone the instrument is in. For instance, GMT-4 is represented as -14400	Seconds
Tau	Sgl	4		Time constant. For instance, 0.125 for Fast, 1.0 for Slow.	Seconds
Manifest	U16	2		Bits of this word represents which values should be recorded: Bit 15-14: 00 -> RMS Levels 01 -> Signal Pk&Avg 10 -> Raw signals Bit 13: (Not Modified) 0 -> Acceleration 1 -> Velocity Bit 0: -> XMax or XRaw Bit 1: -> XAvg or YRaw Bit 2: -> XMin or ZRaw Bit 3: -> YMax Bit 4: -> YAvg Bit 5: -> YMin Bit 6: -> ZMax Bit 7: -> ZAvg Bit 8: -> ZMin	
Interval	U16	2		This represents the time between two successive frames in the data stream that follows. This is not applicable when recording raw signals	1/8 s (125 ms)
Trigger_Val	Sgl	4		This represents the trigger threshold. This is only used when recording raw signals. It is not applicable when recording RMS levels or signal pk&avg	m/s^2 for acceleration m/s for velocity

Trigger Timeout	U32	4		This represents the trigger timeout. The recording will stop whenever the level stays below the threshold for at least that amount of time. This is only used when recording raw signals. It is not applicable when recording RMS levels or signal pk&avg	Seconds

Table 10