## Overall idea

MT messages can come down to the device whether the STxx is currently looking for them or sleeping. The SPI flash on the AP3 will be used to store the messages. The server will send API commands to do writes to the SPI flash and the STxx will read the flash when it is awake and looking for MT messages.

The set of flash commands available to both ends are the commands starting with “G” in the AssetPack Command Set for Skybitz/Ametek Integration document. There are commands to read, erase, write, append, and alter the SPI flash. Alter will copy the page(s) to a scratch area, changing the specified bytes, then erase the original page(s), and finally copy back the modified page(s). Write and append will not erase anything, so they must target previously erase flash areas. The append will look for 5 consecutive 0xFF bytes after the specified address to start writing. Commands (except erase) are limited to at most 64 bytes at a time.

The SPI flash can be erased in 4Kbyte pages. It is erased to all 0xFF bytes and any bit can be written to a 0. An erase is the only way to turn a 0 into a 1. Pages 8-23 (addresses 0x70000-0x7FFFF) are available for our use. Since 4Kbytes can store a relatively large number of messages, only pages 8 and 9 will be used for MT messaging, although it could easily be extended to more pages.

The server will append modified messages to the start of page 8. This allows them to cross to the next page when page 8 becomes full. The original message will be converted to binary to save bandwidth. And sequence of 4 0xFF bytes will have a 0x00 byte inserted after it that will be ignored by the STxx to avoid creating a sequence that the append command will see as a place to start writing. A marker byte (0xAA) will be appended to the message to indicate a new message is completely written to flash and the length including the marker byte and any inserted 0x00 bytes will be prepended to allow easy scanning by the STxx. The binary append command over UART requires any byte 0x0F or less to be escaped by preceding it with a 0x0F byte, but **this is not required by the web API**. A text write of hex values is possible without escapes but may require twice as many bytes sent over the satellite link to represent the data.

On a reset it will be necessary for the STxx device to scan through the flash to find the location of the next unread message. After that, a pointer can be kept in memory to avoid needing to scan again. Having the length written first will aid the scanning process by allowing the STxx to skip through flash without reading it all. Having a partial read message at the beginning of a page or any bad data overwritten by 0x00 will also aid skipping past this data with fewer checks. When it is ready, the STxx will read the flash, checking for new messages. When a new message is found and read out of the flash, the marker byte will be written with 0x00 to indicate it has been read. The read message will be scanned for inserted 0x00 bytes and have them removed before processing. When the first message on a page is marked read, the other page will be erased so the server can write there again. Not all messages can be written in one command, so it is possible for the start of a message to be on one page and the end on another. If this happens, the end of the message will be overwritten with all 0 bytes before erasing the page containing the start of the message. When looking for messages, the length bytes (first 2 bytes) are first checked. The first byte must be between the second byte plus 4 and 1.25 times the second byte plus 4 to indicate a possibly valid message. In addition, the marker must match the expected value. When looking for the next message if an unrecognized byte is encountered, it will be written with 0x00 and the following byte checked.

## Converting an MT SMS message to a Satellite message and sending.

This message will be used as an example:

>35000000313B9AC9ED000000000D82800500FFFFFFFFFFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7

1. Strip off the leading ‘>’ and convert to binary.

{0x35, 0x00, 0x00, 0x00, 0x31, 0x3B, 0x9A, 0xC9, 0xED, 0x00, 0x00, 0x00, 0x00, 0x0D, 0x82, 0x80, 0x05, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x02, 0x12, 0x00, 0x05, 0x83, 0x92, 0x7F, 0x3C, 0xB4, 0x00, 0x12, 0x84, 0xBA, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x0A, 0x10, 0x00, 0x00, 0xC8, 0xC8, 0x4D, 0xE7}

2. After every sequence of 4 0xFF bytes add a 0x00 byte.

{0x35, 0x00, 0x00, 0x00, 0x31, 0x3B, 0x9A, 0xC9, 0xED, 0x00, 0x00, 0x00, 0x00, 0x0D, 0x82, 0x80, 0x05, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0xFF, 0xFF, 0xFF, 0x02, 0x12, 0x00, 0x05, 0x83, 0x92, 0x7F, 0x3C, 0xB4, 0x00, 0x12, 0x84, 0xBA, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x0A, 0x10, 0x00, 0x00, 0xC8, 0xC8, 0x4D, 0xE7}

3. Append 0xAA.

{0x35, 0x00, 0x00, 0x00, 0x31, 0x3B, 0x9A, 0xC9, 0xED, 0x00, 0x00, 0x00, 0x00, 0x0D, 0x82, 0x80, 0x05, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0xFF, 0xFF, 0xFF, 0x02, 0x12, 0x00, 0x05, 0x83, 0x92, 0x7F, 0x3C, 0xB4, 0x00, 0x12, 0x84, 0xBA, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x0A, 0x10, 0x00, 0x00, 0xC8, 0xC8, 0x4D, 0xE7, 0xAA}

4. Prepend the total length. **This is what will end up in flash.**

{0x3A, 0x35, 0x00, 0x00, 0x00, 0x31, 0x3B, 0x9A, 0xC9, 0xED, 0x00, 0x00, 0x00, 0x00, 0x0D, 0x82, 0x80, 0x05, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0xFF, 0xFF, 0xFF, 0x02, 0x12, 0x00, 0x05, 0x83, 0x92, 0x7F, 0x3C, 0xB4, 0x00, 0x12, 0x84, 0xBA, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x0A, 0x10, 0x00, 0x00, 0xC8, 0xC8, 0x4D, 0xE7, 0xAA}

5. Split into smaller sequences of at most 64 bytes. If the last byte of any sequence would be 0xFF, move it to the next sequence. There are only 63 bytes in this example, so for illustration purposes it will be split at 27 bytes (27th byte highlighted). This will also demonstrate moving 0xFF bytes to the next sequence.

{0x3A, 0x35, 0x00, 0x00, 0x00, 0x31, 0x3B, 0x9A, 0xC9, 0xED, 0x00, 0x00, 0x00, 0x00, 0x0D, 0x82, 0x80, 0x05, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0x00}

{0xFF, 0xFF, 0xFF, 0x02, 0x12, 0x00, 0x05, 0x83, 0x92, 0x7F, 0x3C, 0xB4, 0x00, 0x12, 0x84, 0xBA, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x0A, 0x10, 0x00, 0x00, 0xC8, 0xC8, 0x4D, 0xE7, 0xAA}

6. Prepare the JSON for the API. The data is “\u0006cn 8/” followed by the sequence of bytes. This example will be 2 commands. Replace the ESN with the ESN of the device you are sending to.

[{"file":"remote","action":"spiflash","filter":"(esn=300234068885830)","data":"\u0006cn 8/:5\u0000\u0000\u00001;\u009A\u00C9\u00ED\u0000\u0000\u0000\u0000\u000D\u0082\u0080\u0005\u0000\u00FF\u00FF\u00FF\u00FF\u0000"},

{"file":"remote","action":"spiflash","filter":"(esn=300234068885830)","data":"\u0006cn 8/\u00FF\u00FF\u00FF\u0002\u0012\u0000\u0005\u0083\u0092\u007F<\u00B4\u0000\u0012\u0084\u00BA\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u0000\u001E\u000A\u0010\u0000\u0000\u00C8\u00C8M\u00E7\u00AA"}]

7. Send it to the API.

## Device test cases

Test setup is done using a USB to RS-485 adapter connected directly to the AP3. The flash is erased, and data programmed into the flash for testing. The device is the reset to start reading flash fresh. Some cases will then have additional writes to flash. Flash is only scanned on reset and then pointers to locations in flash are kept to improve access time.

MO messages are deleted frequently between tests to reduce satellite data usage.

Note: These messages were actual messages to RTU 999999981. They may have undesirable effects on other devices.

### 1 valid unread message at start of page 8

Setup with the following 3 commands:

1. Gec 8
2. Gec 9
3. Gfc 8/3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Reset the device and observe that the message is processed

### 1 valid unread message at start of page 9

Setup with the following 3 commands:

1. Gec 8
2. Gec 9
3. Gfc 9/3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Reset the device and observe that the message is processed. Observe that both pages are erased after the message is deleted.

### Starting with 0 byte on page 8

Setup with the following 3 commands:

1. Gec 8
2. Gec 9
3. Gfc 8/003A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Reset the device and observe that the message is processed.

### Starting with 0 byte on page 9

Setup with the following 3 commands:

1. Gec 8
2. Gec 9
3. Gfc 9/003A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Reset the device and observe that the message is processed. Observe that both pages are erased after the message is deleted.

### Starting with unexpected byte on page 8

Setup with the following 3 commands:

1. Gec 8
2. Gec 9
3. Gfc 8/FF3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Reset the device and observe that the message is processed.

### Starting with unexpected byte on page 9

Setup with the following 3 commands:

1. Gec 8
2. Gec 9
3. Gfc 9/FF3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Reset the device and observe that the message is processed. Observe that both pages are erased after the message is deleted.

### Partial message on page 9 continued on page 8

Setup with the following 4 commands:

1. Gec 8
2. Gec 9
3. Gfc 9/3A35000000313B9AC9ED000000000D82800500
4. Gfc 8/FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Reset the device and observe that the message is processed. After the message is deleted observe that page 9 is erased page 8 starts with 40 00s.

### Partial message on startup, completed later

Setup with the following 3 commands:

1. Gec 8
2. Gec 9
3. Gfc 9/3A35000000313B9AC9ED000000000D82800500

Reset the device and observe that the message is **not** processed before it goes back to sleep. Observe that it is unchanged in flash.

Send the following command:

1. Gfc 8/FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Wait for the next callout, **do not reset the device**. Observe that the message is processed. After the message is deleted observe that page 9 is erased page 8 starts with 40 00s.

### Read message on page 9 followed by unread message on page 8.

Setup with the following 4 commands:

1. Gec 8
2. Gec 9
3. Gfc 9/141080003E0C3B9AC9ED00000000037CA201CDD200
4. Gfc 8/3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Reset the device and observe that only the second message is processed. Observe that page 9 is erased.

### Read messages on both pages

Setup with the following 4 commands:

1. Gec 8
2. Gec 9
3. Gfc 9/141080003E0C3B9AC9ED00000000037CA201CDD200
4. Gfc 8/3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE700

Reset the device and observe that page 9 is erased.

### Starting with an invalid unread message

Setup with the following 3 commands:

1. Gec 8
2. Gec 9
3. Gfc 8/3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84D00AA

Reset the device and observe that the message is **not** processed, but the ending AA marker is changed to 00.

### 2 unread messages

Setup with the following 4 commands:

1. Gec 8
2. Gec 9
3. Gfc 8/141080003E0C3B9AC9ED00000000037CA201CDD2AA
4. Gfc 8/3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Reset the device and observe that both messages are processed.

### Read message followed by unread message

Setup with the following 4 commands:

1. Gec 8
2. Gec 9
3. Gfc 8/141080003E0C3B9AC9ED00000000037CA201CDD200
4. Gfc 8/3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Reset the device and observe that only the second message is processed.

### Messages received after startup

Setup with the following 2 commands:

1. Gec 8
2. Gec 9

Reset the device and allow it to sleep again. Observe no messages processed.

Send the following command:

1. Gfc 8/141080003E0C3B9AC9ED00000000037CA201CDD2AA

Observe that the message is processed on the next callout, **do not reset the device**.

Send the following command:

1. Gfc 8/3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Observe that it is processed (may be on the next callout, **do not reset the device**).

### Page 8 full of messages going onto page 9

Setup with the following 2 commands:

1. Gec 8
2. Gec 9

Send the following command 71 times. There should be 1 partial and 1 full message on page 9.

1. Gfc 8/3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Reset the device and observe the messages being processed. After they are deleted, observe that both pages are erased.

### Both pages filled with random values

Setup with the following 2 commands:

1. Gec 8
2. Gec 9

Send the following command 128 times (data was generated by “dd if=/dev/random bs=64 count=1 | xxd -p” on cygwin):

1. Gfc 8/d07e6338cce810b8d1a10213268139f11ed84b55f31d3060518d84ea2a07c2fb3c7f4bf850bbb34c82d811fadcd401ea3122665b325f974c592fb1269d7f6aee

Reset the device and observe that both pages are erased.

NOTE: This will take longer than power from the AssetPack will last and may take many (30?) callout attempts to complete. Resetting to force a new callout attempt can be done to make it progress faster. Powering from a bench supply instead of the AssetPack is also an option. It will take approximately 40 minutes on continuous power. *It is not expected that this condition will occur during normal operation and no attempt has been made to make it efficient.*

### Garbage between unread messages

Setup with the following 5 commands:

1. Gec 8
2. Gec 9
3. Gfc 8/141080003E0C3B9AC9ED00000000037CA201CDD2AA
4. Gfc 8/0102030405
5. Gfc 8/3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Reset the device and observe that both messages are processed, and the garbage is overwritten with 00s.

### Garbage between read and unread messages

Setup with the following 5 commands:

1. Gec 8
2. Gec 9
3. Gfc 8/141080003E0C3B9AC9ED00000000037CA201CDD200
4. Gfc 8/0102030405
5. Gfc 8/3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Reset the device and observe that the second message is processed, and the garbage is overwritten with 00s.

### Read message following unread message

Setup with the following 5 commands:

1. Gec 8
2. Gec 9
3. Gfc 8/3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA
4. Gfc 8/141080003E0C3B9AC9ED00000000037CA201CDD200
5. Gfc 8/3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Reset the device and observe that the first and last messages are processed and the second one is overwritten with 0s.

### Message with invalid marker

Setup with the following 4 commands:

1. Gec 8
2. Gec 9
3. Gfc 8/141080003E0C3B9AC9ED00000000037CA201CDD255
4. Gfc 8/3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Reset the device and observe that the first message is overwritten with 00s and the second message is processed.

### Bad length for message

Setup with the following 4 commands:

1. Gec 8
2. Gec 9
3. Gfc 8/101080003E0C3B9AC9ED00000000037CAA
4. Gfc 8/3A35000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C84DE7AA

Reset the device and observe that the first message is overwritten with 00s and the second message is processed.

### Message > 64 bytes (needs 2 reads)

This message is constructed to be valid but was never sent to the device. It is a combination of 2 other messages.

Setup with the following 4 commands:

1. Gec 8
2. Gec 9
3. Gfc 8/4742000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C80C3B9AC9ED000000
4. Gfc 8/00037CA2015C66AA

Reset the device and observe that the message is processed.

### Message > 64 bytes crossing from page 9 to page 8 (needs 3 reads, 2 on page 9)

This message is constructed to be valid but was never sent to the device. It is a combination of 2 other messages.

Setup with the following 5 commands:

1. Gec 8
2. Gec 9
3. Gfc 9/4742000000313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C80C3B9AC9ED000000
4. Gfc 9/00037CA2
5. Gfc 8/015C66AA

Reset the device and observe that the message is processed.

### Message > 64 bytes crossing from page 9 to page 8 (needs 3 reads, 2 on page 8)

This message is constructed to be valid but was never sent to the device. It is a combination of 2 other messages.

Setup with the following 5 commands:

1. Gec 8
2. Gec 9
3. Gfc 9/47420000
4. Gfc 8/00313B9AC9ED000000000D82800500FFFFFFFF00FFFFFF0212000583927F3CB4001284BA0000000000000000001E0A100000C8C80C3B9AC9ED00000000037CA2
5. Gfc 8/015C66AA

Reset the device and observe that the message is processed.