Stream:	Internet Engineering Task Force (IETF)
RFC:	9442
Category:	Standards Track
Published:	July 2023
ISSN:	2070-1721
Authors:	JC. Zúñiga C. Gomez *Universitat Politècnica de Catalunya*

S. Aguilar *Universitat Politècnica de Catalunya*
L. Toutain *IMT-Atlantique*
S. Céspedes *Concordia University*
D. Wistuba *NIC Labs, Universidad de Chile*
J. Boite *Unabiz (Sigfox)*

RFC 9442 Static Context Header Compression (SCHC) over Sigfox Low-Power Wide Area Network (LPWAN)

Abstract

The Static Context Header Compression (SCHC) and fragmentation specification (RFC 8724) describes a generic framework for application header compression and fragmentation modes designed for Low-Power Wide Area Network (LPWAN) technologies. This document defines a profile of SCHC over Sigfox LPWAN and provides optimal parameter values and modes of operation.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Generic Framework for Static Context Header Compression (SCHC) and Fragmentation specification [RFC8724] can be used in conjunction with any of the four LPWAN technologies described in [RFC8376]. These LPWANs have similar characteristics, such as star-oriented topologies, network architecture, connected devices with built-in applications, etc.

SCHC offers a considerable degree of flexibility to accommodate all these LPWAN technologies. Even though there are a great number of similarities between them, some differences exist with respect to the transmission characteristics, payload sizes, etc. Hence, there are optimal parameters and modes of operation that can be used when SCHC is used in conjunction with a specific LPWAN technology.

Sigfox is an LPWAN technology that offers energy-efficient connectivity for devices at a very low cost. Complete Sigfox documentation can be found in [sigfox-docs]. Sigfox aims to provide a very wide area network composed of Base Stations that receive short Uplink messages (up to 12 bytes in size) sent by devices over the long-range Sigfox radio protocol, as described in [RFC8376]. Base Stations then forward messages to the Sigfox Cloud infrastructure for further processing (e.g., to offer geolocation services) and final delivery to the customer. Base Stations also relay Downlink messages (with a fixed 8-byte size) sent by the Sigfox Cloud to the devices, i.e., Downlink messages are being generated when devices explicitly request these messages with a flag in an Uplink message. With SCHC functionalities, the Sigfox network offers more reliable communications (including recovery of lost messages) and is able to convey extended-size payloads (allowing for fragmentation/reassembly of messages) [sigfox-spec].

This document describes the parameters, settings, and modes of operation to be used when SCHC is implemented over a Sigfox LPWAN. The set of parameters forms a "SCHC over Sigfox Profile". The SCHC over Sigfox Profile is applicable to the Sigfox Radio specification versions up to v1.6/ March 2022 [sigfox-spec] (support for future versions would have to be assessed).

2. Terminology

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

It is assumed that the reader is familiar with the terms and mechanisms defined in [RFC8376] and [RFC8724]. Also, it is assumed that the reader is familiar with Sigfox terminology [sigfox-spec].

3. SCHC over Sigfox

The Generic SCHC Framework described in [RFC8724] takes advantage of previous knowledge of traffic flows existing in LPWAN applications to avoid context synchronization.

Contexts need to be stored and pre-configured on both ends. This can be done either by using a provisioning protocol, by out-of-band means, or by pre-provisioning them (e.g., at manufacturing time). For example, the context exchange can be done by using the Network Configuration Protocol (NETCONF) [RFC6241] with Secure Shell (SSH), RESTCONF [RFC8040] with secure HTTP methods, and CoAP Management Interface (CORECONF) [CORE-COMI] with the Constrained Application Protocol (CoAP) [RFC7252] as provisioning protocols. The contexts can be encoded in XML under NETCONF, in JSON [RFC8259] under RESTCONF, and in Concise Binary Object Representation (CBOR) [RFC8949] under CORECONF. The way contexts are configured and stored on both ends is out of the scope of this document.

3.1. Network Architecture

Figure 1 represents the architecture for Compression/Decompression (C/D) and Fragmentation/Reassembly (F/R) based on the terminology defined in [RFC8376], where the Radio Gateway (RGW) is a Sigfox Base Station and the Network Gateway (NGW) is the Sigfox cloud-based Network.

```
  Sigfox Device                                           Application
+----------------+                                     +--------------+
| APP1 APP2 APP3 |                                     |APP1 APP2 APP3|
+----------------+                                     +--------------+
|   UDP  |       |                                     |     |  UDP   |
|  IPv6  |       |                                     |     | IPv6   |
+--------+       |                                     |     +--------+
| SCHC C/D & F/R |                                     |              |
|                |                                     |              |
+-------+--------+                                     +--------+-----+
        $                                                       .
        $   +---------+     +--------------+     +---------+    .
        $   |         |     |   Network    |     | Network |    .
        +~~ |Sigfox BS|     |   Gateway    |     |  SCHC   |    .
            |  (RGW)  | === |    (NGW)     | ... |C/D & F/R|.....
            |         |     | Sigfox Cloud |     |         |   IP-based
            +---------+     +--------------+     +---------+   Network
------- Uplink message ------>
                                       <------- Downlink message ------
Legend:
$, ~ : Radio link
= : Internal Sigfox Network
. : External IP-based Network
```

Figure 1: Network Architecture

In the case of the global Sigfox network, RGWs (or Base Stations) are distributed over multiple countries wherever the Sigfox LPWAN service is provided. The NGW (or cloud-based Sigfox Core Network) is a single entity that connects to all RGWs (Sigfox Base Stations) in the world, hence providing a global single star Network topology.

The Sigfox Device sends application packets that are compressed and/or fragmented by a SCHC C/D + F/R to reduce header size and/or fragment the packet. The resulting SCHC message is sent over a layer two (L2) Sigfox frame to the Sigfox Base Stations, which then forward the SCHC message to the NGW. The NGW then delivers the SCHC message and associated gathered metadata to the Network SCHC C/D + F/R.

The Sigfox cloud-based Network communicates with the Network SCHC C/D + F/R for compression/decompression and/or for fragmentation/reassembly. The Network SCHC C/D + F/R shares the same set of Rules as the device SCHC C/D + F/R. The Network SCHC C/D + F/R can be collocated with the NGW or it could be located in a different place, as long as a tunnel or secured communication is established between the NGW and the SCHC C/D + F/R functions. After decompression and/or reassembly, the packet can be forwarded over the Internet to one (or several) LPWAN Application Server(s) (App(s)).

The SCHC C/D + F/R processes are bidirectional, so the same principles are applicable on both Uplink (UL) and Downlink (DL).

3.2. Uplink

Uplink Sigfox transmissions occur in repetitions over different times and frequencies. Besides time and frequency diversities, the Sigfox network also provides spatial diversity, as potentially an Uplink message will be received by several Base Stations. The Uplink message application payload size can be up to 12 bytes.

Since all messages are self-contained and Base Stations forward all these messages back to the same Sigfox network, multiple input copies can be combined at the NGW, providing for extra reliability based on the triple diversity (i.e., time, space, and frequency).

A detailed description of the Sigfox radio protocol can be found in [sigfox-spec].

Messages sent from the device to the Network are delivered by the Sigfox cloud-based Network to the Network SCHC C/D + F/R through a callback/API with the following information:

- Device ID
- Message Sequence Number
- Message Payload
- Message Timestamp
- Device Geolocation (optional)
- Received Signal Strength Indicator (RSSI) (optional)
- Device Temperature (optional)
- Device Battery Voltage (optional)

The Device ID is a globally unique identifier assigned to the device, which is included in the Sigfox header of every message. The Message Sequence Number is a monotonically increasing number identifying the specific transmission of this Uplink message, and it is also part of the Sigfox header. The Message Payload corresponds to the payload that the device has sent in the Uplink transmission. Battery Voltage, Device Temperature, and RSSI values are sent in the confirmation control message, which is mandatorily sent by the device after the successful reception of a Downlink message (see [sigfox-callbacks], Section 5.2).

The Message Timestamp, Device Geolocation, RSSI, Device Temperature, and Device Battery Voltage are metadata parameters provided by the Network.

A detailed description of the Sigfox callbacks/APIs can be found in [sigfox-callbacks].

Only messages that have passed the L2 Cyclic Redundancy Check (CRC) at Network reception are delivered by the Sigfox network to the Network SCHC C/D + F/R.

The L2 Word size used by Sigfox is 1 byte (8 bits).

Figure 2 shows a SCHC message sent over Sigfox, where the SCHC message could be a full SCHC Packet (e.g., compressed) or a SCHC Fragment (e.g., a piece of a bigger SCHC Packet).

```
                      | Sigfox Header | Sigfox Payload  |
                      +---------------+---------------- +
                                      |   SCHC Message  |
```

Figure 2: SCHC Message in Sigfox

3.3. Downlink

Downlink transmissions are device-driven and can only take place following an Uplink communication that indicates Downlink communication can be performed. Hence, a Sigfox Device explicitly indicates its intention to receive a Downlink message (with a size of 8 bytes) using a Downlink request flag when sending the preceding Uplink message to the Network. The Downlink request flag is part of the Sigfox protocol headers. After completing the Uplink transmission, the device opens a fixed window for Downlink reception. The delay and duration of the reception opportunity window have fixed values. If there is a Downlink message to be sent for this given device (e.g., either a response to the Uplink message or queued information waiting to be transmitted), the Network transmits this message to the device during the reception window. If no message is received by the device after the reception opportunity window has elapsed, the device closes the reception window opportunity and gets back to the normal mode (e.g., continue Uplink transmissions, sleep, standby, etc.).

When a Downlink message is sent to a device, a reception acknowledgement is generated by the device, sent back to the Network through the Sigfox radio protocol, and reported in the Sigfox network backend.

A detailed description of the Sigfox radio protocol can be found in [sigfox-spec], and a detailed description of the Sigfox callbacks/APIs can be found in [sigfox-callbacks]. A Downlink request flag can be included in the information exchange between the Sigfox network and Network SCHC.

3.3.1. SCHC ACK on Downlink

As explained previously, Downlink transmissions are driven by devices and can only take place following a specific Uplink transmission that indicates and allows a following Downlink opportunity. For this reason, when SCHC bidirectional services are used (e.g., ACK-on-Error fragmentation mode), the SCHC protocol implementation needs to consider the times when a Downlink message (e.g., SCHC Acknowledgement (ACK)) can be sent and/or received.

For the Uplink ACK-on-Error fragmentation mode, a Downlink opportunity **MUST** be indicated by the last fragment of every window, which is signalled by a specific value of the Fragment Compressed Number (FCN) value, i.e., FCN = All-0 or FCN = All-1. The FCN is the tile index in a specific window. The combination of the FCN and the window number uniquely identifies a SCHC Fragment, as explained in [RFC8724]. The device sends the fragments in sequence and, after transmitting FCN = All-0 or FCN = All-1, it opens up a reception opportunity. The Network SCHC can then decide to respond at that opportunity (or wait for a further one) with a SCHC ACK, indicating that there are missing fragments from the current or previous windows. If there is no SCHC ACK to be sent, or if the Network decides to wait for a further Downlink transmission opportunity, then no Downlink transmission takes place at that opportunity and the Uplink transmissions continue after a timeout. Intermediate SCHC Fragments with FCNs that are different from All-0 or All-1 **MUST NOT** use the Downlink request flag to request a SCHC ACK.

3.4. SCHC Rules

The RuleID **MUST** be included in the SCHC header. The total number of Rules to be used directly affects the RuleID field size, and therefore the total size of the fragmentation header. For this reason, it is **RECOMMENDED** to keep the number of Rules that are defined for a specific device to the minimum possible. Large RuleID sizes (and thus larger fragmentation headers) are acceptable for devices without significant energy constraints (e.g., a sensor that is powered by the electricity grid).

RuleIDs can be used to differentiate data traffic classes (e.g., QoS, control vs. data, etc.) and data sessions. They can also be used to interleave simultaneous fragmentation sessions between a device and the Network.

3.5. Fragmentation

The SCHC specification [RFC8724] defines a generic fragmentation functionality that allows sending data packets or files larger than the maximum size of a Sigfox payload. The functionality also defines a mechanism to reliably send multiple messages by allowing to selectively resend any lost fragments.

The SCHC fragmentation supports several modes of operation. These modes have different advantages and disadvantages, depending on the specifics of the underlying LPWAN technology and application use case. This section describes how the SCHC fragmentation functionality should optimally be implemented when used over a Sigfox LPWAN for the most typical use case applications.

As described in Section 8.2.3 of [RFC8724], the integrity of the fragmentation-reassembly process of a SCHC Packet **MUST** be checked at the receiver end. Since only Uplink/Downlink messages/fragments that have passed the Sigfox CRC-check are delivered to the Network/Sigfox Device SCHC C/D + F/R, integrity can be guaranteed when no consecutive messages are missing from the sequence and all FCN bitmaps are complete. With this functionality in mind, and in order to save protocol and processing overhead, the use of a Reassembly Check Sequence (RCS), as described in Section 3.5.1.5, **MUST** be used.

3.5.1. Uplink Fragmentation

Sigfox Uplink transmissions are completely asynchronous and take place in any random frequency of the allowed Uplink bandwidth allocation. In addition, devices may go to deep sleep mode and then wake up and transmit whenever there is a need to send information to the Network, as there is no need to perform any Network attachment, synchronization, or other procedures before transmitting a data packet.

Since Uplink transmissions are asynchronous, a SCHC Fragment can be transmitted at any given time by the device. Sigfox Uplink messages are fixed in size, and as described in [RFC8376], they can carry a payload of 0-12 bytes (0-96 bits). Hence, a single SCHC Tile size, per fragmentation mode, can be defined so that every Sigfox message always carries one SCHC Tile.

When the ACK-on-Error mode is used for Uplink fragmentation, the SCHC Compound ACK defined in [RFC9441] **MUST** be used in the Downlink responses.

3.5.1.1. SCHC Sender-Abort

As defined in [RFC8724], a SCHC Sender-Abort can be triggered when the number of SCHC ACK REQ attempts is greater than or equal to MAX_ACK_REQUESTS. In the case of SCHC over Sigfox, a SCHC Sender-Abort **MUST** be sent if the number of repeated All-1s sent in sequence, without a Compound ACK reception in between, is greater than or equal to MAX_ACK_REQUESTS.

3.5.1.2. SCHC Receiver-Abort

As defined in [RFC8724], a SCHC Receiver-Abort is triggered when the receiver has no RuleID and DTag pairs available for a new session. In the case of this profile, a SCHC Receiver-Abort **MUST** be sent if, for a single device, all the RuleIDs are being processed by the receiver (i.e., have an active session) at a certain time and a new one is requested or if the RuleID of the fragment is not valid.

A SCHC Receiver-Abort **MUST** be triggered when the Inactivity Timer expires.

MAX_ACK_REQUESTS can be increased when facing high error rates.

Although a SCHC Receiver-Abort can be triggered at any point in time, a SCHC Receiver-Abort Downlink message **MUST** only be sent when there is a Downlink transmission opportunity.

3.5.1.3. Single-Byte SCHC Header for Uplink Fragmentation

3.5.1.3.1. Uplink No-ACK Mode: Single-Byte SCHC Header

Single-byte SCHC Header No-ACK mode **MUST** be used for transmitting short, non-critical packets that require fragmentation and do not require full reliability. This mode can be used by Uplink-only devices that do not support Downlink communications or by bidirectional devices when they send non-critical data. Note that sending non-critical data by using a reliable fragmentation mode (which is only possible for bidirectional devices) may incur unnecessary overhead.

Since there are no multiple windows in the No-ACK mode, the W bit is not present. However, it **MUST** use the FCN field to indicate the size of the data packet. In this sense, the data packet would need to be split into X fragments and, similarly to the other fragmentation modes, the first transmitted fragment would need to be marked with FCN = X-1. Consecutive fragments **MUST** be marked with decreasing FCN values, having the last fragment marked with FCN = (All-1). Hence, even though the No-ACK mode does not allow recovering missing fragments, it allows implicitly indicating the size of the expected packet to the Network and hence detects whether all fragments have been received or not at the receiver side. In case the FCN field is not used to indicate the size of the data packet, the Network can detect whether all fragments have been received or not by using the integrity check.

When using the Single-byte SCHC Header for Uplink fragmentation, the fragmentation header **MUST** be 8 bits in size and is composed as follows:

- RuleID size: 3 bits
- DTag size (T): 0 bits
- Fragment Compressed Number (FCN) size (N): 5 bits

Other F/R parameters **MUST** be configured as follows:

- As per [RFC8724], in the No-ACK mode, the W (window) field is not present.
- Regular tile size: 11 bytes
- All-1 tile size: 0 to 10 bytes
- Inactivity Timer: Application-dependent. The default value is 12 hours.
- RCS size: 5 bits

The maximum SCHC Packet size is 340 bytes.

Section 3.6.1 presents SCHC Fragment format examples, and Section 5.1 provides fragmentation examples, using Single-byte SCHC Header No-ACK mode.

3.5.1.3.2. Uplink ACK-on-Error Mode: Single-Byte SCHC Header

ACK-on-Error with a single-byte header **MUST** be used for short- to medium-sized packets that need to be sent reliably. ACK-on-Error is optimal for reliable SCHC Packet transmission over Sigfox transmissions, since it leads to a reduced number of ACKs in the lower-capacity Downlink channel. Also, Downlink messages can be sent asynchronously and opportunistically. In contrast, ACK-Always would not minimize the number of ACKs, and No-ACK would not allow reliable transmission.

Allowing transmission of packets/files up to 300 bytes long, the SCHC Uplink fragmentation header size is 8 bits in size and is composed as follows:

- RuleID size: 3 bits
- DTag size (T): 0 bits
- Window index (W) size (M): 2 bits
- Fragment Compressed Number (FCN) size (N): 3 bits

Other F/R parameters **MUST** be configured as follows:

- MAX_ACK_REQUESTS: 5
- WINDOW_SIZE: 7 (i.e., the maximum FCN value is 0b110)
- Regular tile size: 11 bytes
- All-1 tile size: 0 to 10 bytes
- Retransmission Timer: Application-dependent. The default value is 12 hours.
- Inactivity Timer: Application-dependent. The default value is 12 hours.
- RCS size: 3 bits

Section 3.6.2 presents SCHC Fragment format examples, and Section 5.2 provides fragmentation examples, using ACK-on-Error with a single-byte header.

3.5.1.4. Two-Byte SCHC Header for Uplink Fragmentation

ACK-on-Error with a two-byte header **MUST** be used for medium- to large-sized packets that need to be sent reliably. ACK-on-Error is optimal for reliable SCHC Packet transmission over Sigfox, since it leads to a reduced number of ACKs in the lower-capacity Downlink channel. Also, Downlink messages can be sent asynchronously and opportunistically. In contrast, ACK-Always would not minimize the number of ACKs, and No-ACK would not allow reliable transmission.

3.5.1.4.1. Uplink ACK-on-Error Mode: Two-Byte SCHC Header Option 1

In order to allow transmission of medium to large packets/files up to 480 bytes long, the SCHC Uplink fragmentation header size is 16 bits in size and is composed as follows:

- RuleID size: 6 bits
- DTag size (T): 0 bits
- Window index (W) size (M): 2 bits
- Fragment Compressed Number (FCN) size (N): 4 bits
- RCS size: 4 bits

Other F/R parameters **MUST** be configured as follows:

- MAX_ACK_REQUESTS: 5
- WINDOW_SIZE: 12 (with a maximum value of FCN=0b1011)
- Regular tile size: 10 bytes
- All-1 tile size: 1 to 10 bytes
- Retransmission Timer: Application-dependent. The default value is 12 hours.
- Inactivity Timer: Application-dependent. The default value is 12 hours.

Note that WINDOW_SIZE is limited to 12. This is because 4 windows (M = 2) with bitmaps of size 12 can be fitted in a single SCHC Compound ACK.

Section 3.6.3 presents SCHC Fragment format examples, using ACK-on-Error with two-byte header Option 1.

3.5.1.4.2. Uplink ACK-on-Error Mode: Two-Byte SCHC Header Option 2

In order to allow transmission of very large packets/files up to 2400 bytes long, the SCHC Uplink fragmentation header size is 16 bits in size and is composed as follows:

- RuleID size: 8 bits
- DTag size (T): 0 bits
- Window index (W) size (M): 3 bits
- Fragment Compressed Number (FCN) size (N): 5 bits
- RCS size: 5 bits

Other F/R parameters **MUST** be configured as follows:

- MAX_ACK_REQUESTS: 5
- WINDOW_SIZE: 31 (with a maximum value of FCN=0b11110)
- Regular tile size: 10 bytes
- All-1 tile size: 0 to 9 bytes
- Retransmission Timer: Application-dependent. The default value is 12 hours.
- Inactivity Timer: Application-dependent. The default value is 12 hours.

Section 3.6.4 presents SCHC Fragment format examples, using ACK-on-Error with two-byte header Option 2.

3.5.1.5. All-1 SCHC Fragment and RCS Behavior

For ACK-on-Error, as defined in [RFC8724], it is expected that the last SCHC Fragment of the last window will always be delivered with an All-1 FCN. Since this last window may not be full (i.e., it may be composed of fewer than WINDOW_SIZE fragments), an All-1 fragment may follow a value of FCN higher than 1 (0b01). In this case, the receiver cannot determine from the FCN values alone whether there are or are not any missing fragments right before the All-1 fragment.

For Rules where the number of fragments in the last window is unknown, an RCS field **MUST** be used, indicating the number of fragments in the last window, including the All-1. With this RCS value, the receiver can detect if there are missing fragments before the All-1 and hence construct the corresponding SCHC ACK Bitmap accordingly and send it in response to the All-1.

3.5.2. Downlink Fragmentation

In some LPWAN technologies, as part of energy-saving techniques, Downlink transmission is only possible immediately after an Uplink transmission. This allows the device to go in a very deep sleep mode and preserve battery without the need to listen to any information from the Network. This is the case for Sigfox-enabled devices, which can only listen to Downlink communications after performing an Uplink transmission and requesting a Downlink.

When there are fragments to be transmitted in the Downlink, an Uplink message is required to trigger the Downlink communication. In order to avoid a potentially high delay for fragmented datagram transmission in the Downlink, the fragment receiver **MAY** perform an Uplink transmission as soon as possible after reception of a Downlink fragment that is not the last one. Such an Uplink transmission **MAY** be triggered by sending a SCHC message, such as a SCHC ACK. However, other data messages can equally be used to trigger Downlink communications. The fragment receiver **MUST** send an Uplink transmission (e.g., empty message) and request a Downlink every 24 hours when no SCHC session is started. Whether this Uplink transmission is used (and the transmission rate, if used) depends on application-specific requirements.

Sigfox Downlink messages are fixed in size, and as described in [RFC8376] they can carry a payload of 0-8 bytes (0-64 bits). Hence, a single SCHC Tile size per mode can be defined so that every Sigfox message always carries one SCHC Tile.

For reliable Downlink fragment transmission, the ACK-Always mode **SHOULD** be used. Note that ACK-on-Error does not guarantee Uplink feedback (since no SCHC ACK will be sent when no errors occur in a window), and No-ACK would not allow reliable transmission.

The SCHC Downlink fragmentation header size is 8 bits in size and is composed as follows:

- RuleID size: 3 bits
- DTag size (T): 0 bits
- Window index (W) size (M): 0 bits
- Fragment Compressed Number (FCN) size (N): 5 bits

Other F/R parameters **MUST** be configured as follows:

- MAX_ACK_REQUESTS: 5
- WINDOW_SIZE: 31 (with a maximum value of FCN=0b11110)
- Regular tile size: 7 bytes
- All-1 tile size: 0 to 6 bytes
- Retransmission Timer: Application-dependent. The default value is 12 hours.
- Inactivity Timer: Application-dependent. The default value is 12 hours.
- RCS size: 5 bits

3.6. SCHC over Sigfox F/R Message Formats

This section depicts the different formats of SCHC Fragment, SCHC ACK (including the SCHC Compound ACK defined in [RFC9441]), and SCHC Abort used in SCHC over Sigfox.

3.6.1. Uplink No-ACK Mode: Single-Byte SCHC Header

3.6.1.1. Regular SCHC Fragment

Figure 3 shows an example of a Regular SCHC Fragment for all fragments except the last one. As tiles are 11 bytes in size, padding **MUST NOT** be added. The penultimate tile of a SCHC Packet is of regular size.

```
|- SCHC Fragment Header -|
+------------------------+---------+
|   RuleID   |    FCN    | Payload |
+------------+-----------+---------+
|   3 bits   |  5 bits   | 88 bits |
```

Figure 3: Regular SCHC Fragment Format for All Fragments except the Last One

3.6.1.2. All-1 SCHC Fragment

Figure 4 shows an example of the All-1 message. The All-1 message **MAY** contain the last tile of the SCHC Packet. Padding **MUST NOT** be added, as the resulting size is a multiple of an L2 Word.

The All-1 messages Fragment Header includes a 5-bit RCS, and 3 bits are added as padding to complete 2 bytes. The payload size of the All-1 message ranges from 0 to 80 bits.

```
|--------  SCHC Fragment Header -------|
+--------------------------------------+--------------+
| RuleID | FCN=ALL-1 |  RCS    |  b'000 |   Payload    |
+--------+-----------+--------+--------+--------------+
| 3 bits |  5 bits   | 5 bits | 3 bits | 0 to 80 bits |
```

Figure 4: All-1 SCHC Message Format with the Last Tile

As per [RFC8724], the All-1 must be distinguishable from a SCHC Sender-Abort message (with the same RuleID and N values). The All-1 **MAY** have the last tile of the SCHC Packet. The SCHC Sender-Abort message header size is 1 byte with no padding bits.

For the All-1 message to be distinguishable from the Sender-Abort message, the Sender-Abort message **MUST** be 1 byte (only header with no padding). This way, the minimum size of the All-1 is 2 bytes, and the Sender-Abort message is 1 byte.

3.6.1.3. SCHC Sender-Abort Message Format

```
     Sender-Abort
|------ Header ------|
+--------------------+
| RuleID | FCN=ALL-1 |
+--------+-----------+
| 3 bits |  5 bits   |
```

Figure 5: SCHC Sender-Abort Message Format

3.6.2. Uplink ACK-on-Error Mode: Single-Byte SCHC Header

3.6.2.1. Regular SCHC Fragment

Figure 6 shows an example of a Regular SCHC Fragment for all fragments except the last one. As tiles are 11 bytes in size, padding **MUST NOT** be added.

```
|-- SCHC Fragment Header --|
+--------------------------+---------+
| RuleID |   W    |  FCN   | Payload |
+--------+--------+--------+---------+
| 3 bits | 2 bits | 3 bits | 88 bits |
```

Figure 6: Regular SCHC Fragment Format for All Fragments except the Last One

The SCHC ACK REQ **MUST NOT** be used, instead the All-1 SCHC Fragment **MUST** be used to request a SCHC ACK from the receiver (Network SCHC). As per [RFC8724], the All-0 message is distinguishable from the SCHC ACK REQ (All-1 message). The penultimate tile of a SCHC Packet is of regular size.

3.6.2.2. All-1 SCHC Fragment

Figure 7 shows an example of the All-1 message. The All-1 message **MAY** contain the last tile of the SCHC Packet. Padding **MUST NOT** be added, as the resulting size is L2-word-multiple.

```
        |-------------  SCHC Fragment Header -----------|
        +-----------------------------------------------+--------------+
        | RuleID |   W    | FCN=ALL-1 |  RCS   |b'00000 |   Payload    |
        +--------+--------+-----------+--------+--------+--------------+
        | 3 bits | 2 bits |  3 bits   | 3 bits | 5 bits | 0 to 80 bits |
```

Figure 7: All-1 SCHC Message Format with the Last Tile

As per [RFC8724], the All-1 must be distinguishable from a SCHC Sender-Abort message (with same RuleID, M, and N values). The All-1 **MAY** have the last tile of the SCHC Packet. The SCHC Sender-Abort message header size is 1 byte with no padding bits.

For the All-1 message to be distinguishable from the Sender-Abort message, the Sender-Abort message **MUST** be 1 byte (only header with no padding). This way, the minimum size of the All-1 is 2 bytes, and the Sender-Abort message is 1 byte.

3.6.2.3. SCHC ACK Format

Figure 8 shows the SCHC ACK format when all fragments have been correctly received (C=1). Padding **MUST** be added to complete the 64-bit Sigfox Downlink frame payload size.

```
                     |---- SCHC ACK Header ----|
                     +-------------------------+---------+
                     | RuleID |    W   | C=b'1 | b'0-pad |
                     +--------+--------+-------+---------+
                     | 3 bits | 2 bits | 1 bit | 58 bits |
```

Figure 8: SCHC Success ACK Message Format

In case SCHC Fragment losses are found in any of the windows of the SCHC Packet (C=0), the SCHC Compound ACK defined in [RFC9441] **MUST** be used. The SCHC Compound ACK message format is shown in Figure 9.

```
    |--- SCHC ACK Header ---|- W=w1 -|...|----- W=wi ------|
    +------+--------+-------+--------+...+--------+--------+------+-------+
    |RuleID| W=b'w1 | C=b'0 | Bitmap |...| W=b'wi | Bitmap | b'00 |b'0-pad|
    +------+--------+-------+--------+...+--------+--------+------+-------+
    |3 bits| 2 bits | 1 bit | 7 bits |...| 2 bits | 7 bits |2 bits|
```

Figure 9: SCHC Compound ACK Message Format

Losses are found in windows W = w1,...,wi, where w1 < w2 <...< wi.

3.6.2.4. SCHC Sender-Abort Message Format

```
                  |---- Sender-Abort Header ----|
                  +-----------------------------+
                  | RuleID | W=b'11 | FCN=ALL-1 |
                  +--------+--------+-----------+
                  | 3 bits | 2 bits |  3 bits   |
```

Figure 10: SCHC Sender-Abort Message Format

3.6.2.5. SCHC Receiver-Abort Message Format

```
   |- Receiver-Abort Header -|
   +---------------------------------+-----------------+---------+
   | RuleID | W=b'11 | C=b'1 |  b'11 |  0xFF (all 1's) | b'0-pad |
   +--------+--------+-------+-------+-----------------+---------+
   | 3 bits | 2 bits | 1 bit | 2 bit |  8 bit          | 48 bits |
             next L2 Word boundary ->| <-- L2 Word --> |
```

Figure 11: SCHC Receiver-Abort Message Format

3.6.3. Uplink ACK-on-Error Mode: Two-Byte SCHC Header Option 1

3.6.3.1. Regular SCHC Fragment

Figure 12 shows an example of a Regular SCHC Fragment for all fragments except the last one. The penultimate tile of a SCHC Packet is of the regular size.

```
           |------- SCHC Fragment Header ------|
           +-----------------------------------+---------+
           | RuleID |    W   |  FCN   | b'0000 | Payload |
           +--------+--------+--------+--------+---------+
           | 6 bits | 2 bits | 4 bits | 4 bits | 80 bits |
```

Figure 12: Regular SCHC Fragment Format for All Fragments except the Last One

The SCHC ACK REQ **MUST NOT** be used, instead the All-1 SCHC Fragment **MUST** be used to request a SCHC ACK from the receiver (Network SCHC). As per [RFC8724], the All-0 message is distinguishable from the SCHC ACK REQ (All-1 message).

3.6.3.2. All-1 SCHC Fragment

Figure 13 shows an example of the All-1 message. The All-1 message **MUST** contain the last tile of the SCHC Packet.

The All-1 message Fragment Header contains an RCS of 4 bits to complete the two-byte size. The size of the last tile ranges from 8 to 80 bits.

```
|--------- SCHC Fragment Header -------|
+--------------------------------------+--------------+
| RuleID |    W    | FCN=ALL-1 |  RCS   |    Payload   |
+--------+--------+-----------+--------+--------------+
| 6 bits | 2 bits |  4 bits   | 4 bits | 8 to 80 bits |
```

Figure 13: All-1 SCHC Message Format with the Last Tile

As per [RFC8724], the All-1 must be distinguishable from the SCHC Sender-Abort message (with same RuleID, M, and N values). The All-1 **MUST** have the last tile of the SCHC Packet that **MUST** be at least 1 byte. The SCHC Sender-Abort message header size is 2 bytes with no padding bits.

For the All-1 message to be distinguishable from the Sender-Abort message, the Sender-Abort message **MUST** be 2 bytes (only header with no padding). This way, the minimum size of the All-1 is 3 bytes, and the Sender-Abort message is 2 bytes.

3.6.3.3. SCHC ACK Format

Figure 14 shows the SCHC ACK format when all fragments have been correctly received (C=1). Padding **MUST** be added to complete the 64-bit Sigfox Downlink frame payload size.

```
|---- SCHC ACK Header ----|
+-------------------------+---------+
| RuleID |    W    | C=b'1 | b'0-pad |
+--------+--------+-------+---------+
| 6 bits | 2 bits | 1 bit | 55 bits |
```

Figure 14: SCHC Success ACK Message Format

The SCHC Compound ACK message **MUST** be used in case SCHC Fragment losses are found in any window of the SCHC Packet (C=0). The SCHC Compound ACK message format is shown in Figure 15. The SCHC Compound ACK can report up to 4 windows with losses, as shown in Figure 16.

When sent in the Downlink, the SCHC Compound ACK **MUST** be 0 padded (padding bits must be 0) to complement the 64 bits required by the Sigfox payload.

```
|--- SCHC ACK Header ---|- W=w1 -|...|---- W=wi -----|
+--------+------+-------+--------+...+------+--------+------+-------+
| RuleID |W=b'w1| C=b'0 | Bitmap |...|W=b'wi| Bitmap | b'00 |b'0-pad|
+--------+------+-------+--------+...+------+--------+------+-------+
| 6 bits |2 bits| 1 bit | 12 bits|...|2 bits| 12 bits|2 bits|
```

Figure 15: SCHC Compound ACK Message Format

Losses are found in windows W = w1,...,wi, where w1 < w2 <...< wi.

```
            |- SCHC ACK Header -|- W=0 -|      |- W=1 -|...
            +------+------+-----+-------+------+-------+...
            |RuleID|W=b'00|C=b'0|Bitmap |W=b'01|Bitmap |...
            +------+------+-----+-------+------+-------+...
            |6 bits|2 bits|1 bit|12 bits|2 bits|12 bits|...

                  ...       |- W=2 -|      |- W=3 -|
                  ...+------+-------+------+-------+---+
                  ...|W=b'10|Bitmap |W=b'11|Bitmap |b'0|
                  ...+------+-------+------+-------+---+
                  ...|2 bits|12 bits|2 bits|12 bits|
```

Figure 16: SCHC Compound ACK Message Format Example with Losses in All Windows

Losses are found in windows W = w1,...,wi, where w1 < w2 <...< wi.

3.6.3.4. SCHC Sender-Abort Message Format

```
               |---- Sender-Abort Header ----|
               +-----------------------------+
               | RuleID |   W    | FCN=ALL-1 |
               +--------+--------+-----------+
               | 6 bits | 2 bits |  4 bits   |
```

Figure 17: SCHC Sender-Abort Message Format

3.6.3.5. SCHC Receiver-Abort Message Format

```
     |- Receiver-Abort Header -|
     +---------------------------------+-----------------+---------+
     | RuleID | W=b'11 | C=b'1 |  0x7F |  0xFF (all 1's) | b'0-pad |
     +--------+--------+-------+-------+-----------------+---------+
     | 6 bits | 2 bits | 1 bit | 7 bit |  8 bit          | 40 bits |
               next L2 Word boundary ->| <-- L2 Word --> |
```

Figure 18: SCHC Receiver-Abort Message Format

3.6.4. Uplink ACK-on-Error Mode: Two-Byte SCHC Header Option 2

3.6.4.1. Regular SCHC Fragment

Figure 19 shows an example of a Regular SCHC Fragment for all fragments except the last one. The penultimate tile of a SCHC Packet is of the regular size.

```
              |-- SCHC Fragment Header --|
              +--------------------------+---------+
              | RuleID |   W    | FCN    | Payload |
              +--------+--------+--------+---------+
              | 8 bits | 3 bits | 5 bits | 80 bits |
```

Figure 19: Regular SCHC Fragment Format for All Fragments except the Last One

The SCHC ACK REQ **MUST NOT** be used, instead the All-1 SCHC Fragment **MUST** be used to request a SCHC ACK from the receiver (Network SCHC). As per [RFC8724], the All-0 message is distinguishable from the SCHC ACK REQ (All-1 message).

3.6.4.2. All-1 SCHC Fragment

Figure 20 shows an example of the All-1 message. The All-1 message **MAY** contain the last tile of the SCHC Packet.

The All-1 message Fragment Header contains an RCS of 5 bits and 3 padding bits to complete a 3-byte Fragment Header. The size of the last tile, if present, ranges from 8 to 72 bits.

```
        |-------------- SCHC Fragment Header -----------|
        +-----------------------------------------------+---------------+
        | RuleID |    W    | FCN=ALL-1 |  RCS    | b'000  |    Payload    |
        +--------+--------+-----------+--------+--------+---------------+
        | 8 bits | 3 bits |   5 bits   | 5 bits | 3 bits | 8 to 72 bits |
```

Figure 20: All-1 SCHC Message Format with the Last Tile

As per [RFC8724], the All-1 must be distinguishable from the SCHC Sender-Abort message (with same RuleID, M, and N values). The SCHC Sender-Abort message header size is 2 bytes with no padding bits.

For the All-1 message to be distinguishable from the Sender-Abort message, the Sender-Abort message **MUST** be 2 bytes (only header with no padding). This way, the minimum size of the All-1 is 3 bytes, and the Sender-Abort message is 2 bytes.

3.6.4.3. SCHC ACK Format

Figure 21 shows the SCHC ACK format when all fragments have been correctly received (C=1). Padding **MUST** be added to complete the 64-bit Sigfox Downlink frame payload size.

```
                     |---- SCHC ACK Header ----|
                     +-----------------------+---------+
                     | RuleID |    W   | C=b'1 | b'0-pad |
                     +--------+--------+-------+---------+
                     | 8 bits | 3 bits | 1 bit | 52 bits |
```

Figure 21: SCHC Success ACK Message Format

The SCHC Compound ACK message **MUST** be used in case SCHC Fragment losses are found in any window of the SCHC Packet (C=0). The SCHC Compound ACK message format is shown in Figure 22. The SCHC Compound ACK can report up to 3 windows with losses.

When sent in the Downlink, the SCHC Compound ACK **MUST** be 0 padded (padding bits must be 0) to complement the 64 bits required by the Sigfox payload.

```
      |-- SCHC ACK Header --|- W=w1 -|...|---- W=wi -----|
      +------+------+-------+--------+...+------+--------+------+-------+
      |RuleID|W=b'w1| C=b'0 | Bitmap |...|W=b'wi| Bitmap | 000  |b'0-pad|
      +------+------+-------+--------+...+------+--------+------+-------+
      |8 bits|3 bits| 1 bit | 31 bits|...|3 bits| 31 bits|3 bits|
```

Figure 22: SCHC Compound ACK Message Format

Losses are found in windows W = w1,...,wi, where w1 < w2 <...< wi.

3.6.4.4. SCHC Sender-Abort Message Format

```
                        |---- Sender-Abort Header ----|
                        +-----------------------------+
                        | RuleID |   W    | FCN=ALL-1 |
                        +--------+--------+-----------+
                        | 8 bits | 3 bits |  5 bits   |
```

Figure 23: SCHC Sender-Abort Message Format

3.6.4.5. SCHC Receiver-Abort Message Format

```
       |-- Receiver-Abort Header -|
       +---------------------------------+-----------------+---------+
       | RuleID | W=b'111 | C=b'1 | b'1111 |  0xFF (all 1's) | b'0-pad |
       +--------+---------+-------+--------+-----------------+---------+
       | 8 bits |  3 bits | 1 bit | 4 bit  |  8 bit          | 40 bits |
                  next L2 Word boundary ->| <-- L2 Word --> |
```

Figure 24: SCHC Receiver-Abort Message Format

3.6.5. Downlink ACK-Always Mode: Single-Byte SCHC Header

3.6.5.1. Regular SCHC Fragment

Figure 25 shows an example of a Regular SCHC Fragment for all fragments except the last one. The penultimate tile of a SCHC Packet is of the regular size.

```
                            SCHC Fragment
                         |--    Header   --|
                         +-----------------+---------+
                         | RuleID |  FCN   | Payload |
                         +--------+--------+---------+
                         | 3 bits | 5 bits | 56 bits |
```

Figure 25: Regular SCHC Fragment Format for All Fragments except the Last One

The SCHC ACK **MUST NOT** be used, instead the All-1 SCHC Fragment **MUST** be used to request a SCHC ACK from the receiver. As per [RFC8724], the All-0 message is distinguishable from the SCHC ACK REQ (All-1 message).

3.6.5.2. All-1 SCHC Fragment

Figure 26 shows an example of the All-1 message. The All-1 message **MAY** contain the last tile of the SCHC Packet.

The All-1 message Fragment Header contains an RCS of 5 bits and 3 padding bits to complete a 2-byte Fragment Header. The size of the last tile, if present, ranges from 8 to 48 bits.

```
|--------- SCHC Fragment Header -------|
+--------------------------------------+--------------+
| RuleID | FCN=ALL-1 |  RCS   | b'000  |    Payload   |
+--------+-----------+--------+--------+--------------+
| 3 bits |  5 bits   | 5 bits | 3 bits | 0 to 48 bits |
```

Figure 26: All-1 SCHC Message Format with the Last Tile

As per [RFC8724], the All-1 must be distinguishable from the SCHC Sender-Abort message (with same RuleID and N values). The SCHC Sender-Abort message header size is 1 byte with no padding bits.

For the All-1 message to be distinguishable from the Sender-Abort message, the Sender-Abort message **MUST** be 1 byte (only header with no padding). This way, the minimum size of the All-1 is 2 bytes, and the Sender-Abort message is 1 bytes.

3.6.5.3. SCHC ACK Format

Figure 27 shows the SCHC ACK format when all fragments have been correctly received (C=1). Padding **MUST** be added to complete 2 bytes.

```
          SCHC ACK
|--   Header   --|
+----------------+---------+
| RuleID | C=b'1 | b'0-pad |
+--------+-------+---------+
| 3 bits | 1 bit |  4 bits |
```

Figure 27: SCHC Success ACK Message Format

The SCHC ACK message format is shown in Figure 28.

```
|---- SCHC ACK Header ----|
+--------+-------+--------+---------+
| RuleID | C=b'0 | Bitmap | b'0-pad |
+--------+-------+--------+---------+
| 3 bits | 1 bit | 31 bits|  5 bits |
```

Figure 28: SCHC Compound ACK Message Format

3.6.5.4. SCHC Sender-Abort Message Format

```
          Sender-Abort
      |----   Header   ----|
      +--------------------+
      | RuleID | FCN=ALL-1 |
      +--------+-----------+
      | 3 bits |  5 bits   |
```

Figure 29: SCHC Sender-Abort Message Format

3.6.5.5. SCHC Receiver-Abort Message Format

```
  Receiver-Abort
|---  Header  ---|
+----------------+--------+-----------------+
| RuleID | C=b'1 | b'1111 |  0xFF (all 1's) |
+--------+-------+--------+-----------------+
| 3 bits | 1 bit | 4 bit  |  8 bit          |
```

Figure 30: SCHC Receiver-Abort Message Format

3.7. Padding

The Sigfox payload fields have different characteristics in Uplink and Downlink.

Uplink messages can contain a payload size from 0 to 12 bytes. The Sigfox radio protocol allows sending zero bits, one single bit of information for binary applications (e.g., status), or an integer number of bytes. Therefore, for 2 or more bits of payload, it is required to add padding to the next integer number of bytes. The reason for this flexibility is to optimize transmission time and hence save battery consumption at the device.

On the other hand, Downlink frames have a fixed length. The payload length **MUST** be 64 bits (i.e., 8 bytes). Hence, if less information bits are to be transmitted, padding **MUST** be used with bits equal to 0. The receiver **MUST** remove the added padding bits before the SCHC reassembly process.

4. Fragmentation Rules Examples

This section provides an example of RuleID configuration for interoperability between the F/R modes presented in this document. Note that the RuleID space for Uplink F/R is different than the one for Downlink F/R; therefore, this section is divided in two subsections: Rules for Uplink fragmentation and Rules for Downlink fragmentation.

For Uplink F/R, multiple header lengths were described in Section 3.5. All of them are part of the SCHC over Sigfox Profile and offer not only low protocol overhead for small payloads (single byte header) but also extensibility to transport larger payloads with more overhead (2-byte header, Options 1 and 2). The usage of the RuleID space for each header length is an implementation choice, but we provide an example of it in the following section. This illustrates implementation choices made in order to 1) identify the different header length and 2) finally parse the RuleID field to identify the RuleID value and execute the associated treatment.

4.1. Uplink Fragmentation Rules Examples

The RuleID field for Uplink F/R modes has different sizes depending on the header length. In order to identify the header length and then the value of the RuleID, the RuleID field is interpreted as follows:

- The RuleID field is the first one to be parsed in the SCHC header, starting from the leftmost bits.
- For Single-byte SCHC Header F/R modes, a RuleID field of 3 bits is expected:
  - If the first 3 leftmost bits have a value different than 0b'111, then it signals a Single-byte SCHC Header F/R mode.
  - If their value is 0b'111, then it signals a Two-byte SCHC Header F/R mode.
- For Single-byte SCHC Header F/R modes:
  - There are 7 RuleIDs available (with values from 0b'000-0b'110); the RuleID with value 0b'111 is reserved to indicate a Two-byte SCHC Header.
  - This set of Rules is called "standard rules", and it is used to implement Single-byte SCHC Header modes.
  - Each RuleID is associated with a set of properties defining if Uplink F/R is used and which Uplink F/R mode is used. As an example, the RuleID 0b'000 is mapped onto Uplink No-ACK Mode: Single-byte SCHC Header, and the RuleIDs 0b'001 and 0b'002 are mapped onto Uplink ACK-on-Error mode: Single-byte SCHC Header (2 RuleIDs to allow for SCHC Packet interleaving).
- For Two-byte SCHC Header F/R modes, at least 6 bits for the RuleID field are expected:
  - The 3 first leftmost bits are always 0b'111.
    - If the following 3 bits have a different value than 0b'111, then it signals the Two-byte SCHC Header Option 1.
    - If the following 3 bits are 0b'111, then it signals the Two-byte SCHC Header Option 2.
  - For the Two-byte SCHC Header Option 1, there are 7 RuleIDs available (0b'111000-0b'111110), 0b'111111 being reserved to indicate the Two-byte SCHC Header Option 2. This set of Rules is called "extended rules", and it is used to implement the Uplink ACK-on-Error mode: Two-byte SCHC Header Option 1.
  - For the Two-byte SCHC Header Option 2, there are 2 additional bits to parse as the RuleID, so 4 RuleIDs are available (0b'11111100-0b'11111111). This set of Rules is used to cover specific cases that previous RuleIDs do not cover. As an example, RuleID 0b'00111111 is used to transport uncompressed IPv6 packets using the Uplink ACK-on-Error mode: Two-byte SCHC Header Option 2.

4.2. Downlink Fragmentation Rules Example

For the Downlink ACK-Always Mode: Single-byte SCHC Header, RuleIDs can get values in ranges from 0b'000 to 0b'111.

5. Fragmentation Sequence Examples

In this section, some sequence diagrams depict message exchanges for different fragmentation modes and use cases are shown. In the examples, 'Seq' indicates the Sigfox Sequence Number of the frame carrying a fragment.

5.1. Uplink No-ACK Examples

The FCN field indicates the size of the data packet. The first fragment is marked with FCN = X-1, where X is the number of fragments the message is split into. All fragments are marked with decreasing FCN values. The last packet fragment is marked with FCN = All-1 (1111).

Case No Losses - All fragments are sent and received successfully.

```
        Sender                     Receiver
          |-------FCN=6,Seq=1-------->|
          |-------FCN=5,Seq=2-------->|
          |-------FCN=4,Seq=3-------->|
          |-------FCN=3,Seq=4-------->|
          |-------FCN=2,Seq=5-------->|
          |-------FCN=1,Seq=6-------->|
          |-------FCN=15,Seq=7------->| All fragments received
        (End)
```

Figure 31: Uplink No-ACK No-Losses

When the first SCHC Fragment is received, the receiver can calculate the total number of SCHC Fragments that the SCHC Packet is composed of. For example, if the first fragment is numbered with FCN=6, the receiver can expect six more messages/fragments (i.e., with FCN going from 5 downwards and the last fragment with an FCN equal to 15).

Case Losses on Any Fragment except the First

```
     Sender                     Receiver
       |-------FCN=6,Seq=1-------->|
       |-------FCN=5,Seq=2----X    |
       |-------FCN=4,Seq=3-------->|
       |-------FCN=3,Seq=4-------->|
       |-------FCN=2,Seq=5-------->|
       |-------FCN=1,Seq=6-------->|
       |-------FCN=15,Seq=7------->| Missing Fragment Unable to reassemble
     (End)
```

Figure 32: Uplink No-ACK Losses (Scenario 1)

5.2. Uplink ACK-on-Error Examples: Single-Byte SCHC Header

The Single-byte SCHC Header ACK-on-Error mode allows sending up to 28 fragments and packet sizes up to 300 bytes. The SCHC Fragments may be delivered asynchronously, and Downlink ACK can be sent opportunistically.

Case No Losses

The Downlink flag must be enabled in the sender Uplink message to allow a Downlink message from the receiver. The Downlink Enable in the figures shows where the sender **MUST** enable the Downlink and wait for an ACK.

```
                 Sender                   Receiver
                   |-----W=0,FCN=6,Seq=1----->|
                   |-----W=0,FCN=5,Seq=2----->|
                   |-----W=0,FCN=4,Seq=3----->|
                   |-----W=0,FCN=3,Seq=4----->|
                   |-----W=0,FCN=2,Seq=5----->|
                   |-----W=0,FCN=1,Seq=6----->|
         DL Enable |-----W=0,FCN=0,Seq=7----->|
               (no ACK)
                   |-----W=1,FCN=6,Seq=8----->|
                   |-----W=1,FCN=5,Seq=9----->|
                   |-----W=1,FCN=4,Seq=10---->|
         DL Enable |-----W=1,FCN=7,Seq=11---->| All fragments received
                   |<- Compound ACK,W=1,C=1 --| C=1
                 (End)
```

Figure 33: Uplink ACK-on-Error No-Losses

Case Fragment Losses in the First Window

In this case, fragments are lost in the first window (W=0). After the first All-0 message arrives, the receiver leverages the opportunity and sends a SCHC ACK with the corresponding bitmap and C=0.

After the loss fragments from the first window (W=0) are resent, the sender continues transmitting the fragments of the following window (W=1) without opening a reception opportunity. Finally, the All-1 fragment is sent, the Downlink is enabled, and the SCHC ACK is received with C=1. Note that the SCHC Compound ACK also uses a Sequence Number.

```
            Sender                    Receiver
              |-----W=0,FCN=6,Seq=1----->|
              |-----W=0,FCN=5,Seq=2--X   |
              |-----W=0,FCN=4,Seq=3----->|
              |-----W=0,FCN=3,Seq=4----->|
              |-----W=0,FCN=2,Seq=5--X   |                    __
              |-----W=0,FCN=1,Seq=6----->|                   | W=0
    DL Enable |-----W=0,FCN=0,Seq=7----->| Missing Fragments<- FCN=5,Seq=2
              |<- Compound ACK,W=0,C=0 --| Bitmap:1011011    | FCN=2,Seq=5
              |-----W=0,FCN=5,Seq=9----->|                    --
              |-----W=0,FCN=2,Seq=10---->|
              |-----W=1,FCN=6,Seq=11---->|
              |-----W=1,FCN=5,Seq=12---->|
              |-----W=1,FCN=4,Seq=13---->|
    DL Enable |-----W=1,FCN=7,Seq=14---->| All fragments received
              |<-Compound ACK,W=1,C=1 ---| C=1
            (End)
```

Figure 34: Uplink ACK-on-Error Losses in the First Window

Case Fragment All-0 Lost in the First Window (W=0)

In this example, the All-0 of the first window (W=0) is lost. Therefore, the receiver waits for the next All-0 message of intermediate windows or All-1 message of last window to generate the corresponding SCHC ACK, which indicates that the All-0 of window 0 is absent.

The sender resends the missing All-0 messages (with any other missing fragment from window 0) without opening a reception opportunity.

```
             Sender                    Receiver
               |-----W=0,FCN=6,Seq=1----->|
               |-----W=0,FCN=5,Seq=2----->|
               |-----W=0,FCN=4,Seq=3----->|
               |-----W=0,FCN=3,Seq=4----->|
               |-----W=0,FCN=2,Seq=5----->|
               |-----W=0,FCN=1,Seq=6----->| DL Enable
               |-----W=0,FCN=0,Seq=7--X   |
           (no ACK)
               |-----W=1,FCN=6,Seq=8----->|
               |-----W=1,FCN=5,Seq=9----->|                    __
               |-----W=1,FCN=4,Seq=10---->|                   |W=0
     DL Enable |-----W=1,FCN=7,Seq=11---->| Missing Fragment<- FCN=0,Seq=7
               |<-Compound ACK,W=0,C=0 ---| Bitmap:1111110    |__
               |-----W=0,FCN=0,Seq=13---->| All fragments received
     DL Enable |-----W=1,FCN=7,Seq=14---->|
               |<-Compound ACK,W=1,C=1 ---| C=1
             (End)
```

Figure 35: Uplink ACK-on-Error All-0 Lost in the First Window

In the following diagram, besides the All-0, there are other fragment losses in the first window (W=0).

```
             Sender                    Receiver
               |-----W=0,FCN=6,Seq=1----->|
               |-----W=0,FCN=5,Seq=2--X   |
               |-----W=0,FCN=4,Seq=3----->|
               |-----W=0,FCN=3,Seq=4--X   |
               |-----W=0,FCN=2,Seq=5----->|
               |-----W=0,FCN=1,Seq=6----->|
     DL Enable |-----W=0,FCN=0,Seq=7--X   |
           (no ACK)
               |-----W=1,FCN=6,Seq=8----->|
               |-----W=1,FCN=5,Seq=9----->|                    __
               |-----W=1,FCN=4,Seq=10---->|                   |W=0
     DL Enable |-----W=1,FCN=7,Seq=11---->| Missing Fragment<- FCN=5,Seq=2
               |<--Compound ACK,W=0,C=0 --| Bitmap:1010110    |FCN=3,Seq=4
               |-----W=0,FCN=5,Seq=13---->|                   |FCN=0,Seq=7
               |-----W=0,FCN=3,Seq=14---->|                    --
               |-----W=0,FCN=0,Seq=15---->| All fragments received
     DL Enable |-----W=1,FCN=7,Seq=16---->|
               |<-Compound ACK,W=1,C=1 ---| C=1
             (End)
```

Figure 36: Uplink ACK-on-Error All-0 and Other Fragments Lost in the First Window

In the next examples, there are fragment losses in both the first (W=0) and second (W=1) windows. The retransmission cycles after the All-1 is sent (i.e., not in intermediate windows) **MUST** always finish with an All-1, as it serves as an ACK Request message to confirm the correct reception of the retransmitted fragments.

```
           Sender                    Receiver
             |-----W=0,FCN=6,Seq=1----->|
             |-----W=0,FCN=5,Seq=2--X   |
             |-----W=0,FCN=4,Seq=3----->|
             |-----W=0,FCN=3,Seq=4--X   |                    __
             |-----W=0,FCN=2,Seq=5----->|                   |W=0
             |-----W=0,FCN=1,Seq=6----->|                   |FCN=5,Seq=2
   DL Enable |-----W=0,FCN=0,Seq=7--X   |                   |FCN=3,Seq=4
        (no ACK)                                            |FCN=0,Seq=7
             |-----W=1,FCN=6,Seq=8--X   |                   |W=1
             |-----W=1,FCN=5,Seq=9----->|                   |FCN=6,Seq=8
             |-----W=1,FCN=4,Seq=10-X   |                   |FCN=4,Seq=10
   DL Enable |-----W=1,FCN=7,Seq=11---->| Missing Fragment<-|__
             |<-Compound ACK,W=0,1,C=0--| Bitmap W=0:1010110
             |-----W=0,FCN=5,Seq=13---->|        W=1:0100001
             |-----W=0,FCN=3,Seq=14---->|
             |-----W=0,FCN=0,Seq=15---->|
             |-----W=1,FCN=6,Seq=16---->|
             |-----W=1,FCN=4,Seq=17---->| All fragments received
   DL Enable |-----W=1,FCN=7,Seq=18---->|
             |<-Compound ACK,W=1,C=1----| C=1
           (End)
```

Figure 37: Uplink ACK-on-Error All-0 and Other Fragments Lost in the First and Second Windows (1)

The figure below is a similar case as above but with fewer fragments in the second window (W=1).

```
           Sender                    Receiver
             |-----W=0,FCN=6,Seq=1----->|
             |-----W=0,FCN=5,Seq=2--X   |
             |-----W=0,FCN=4,Seq=3----->|
             |-----W=0,FCN=3,Seq=4--X   |
             |-----W=0,FCN=2,Seq=5----->|                     __
             |-----W=0,FCN=1,Seq=6----->|                    |W=0
   DL Enable |-----W=0,FCN=0,Seq=7--X   |                    |FCN=5,Seq=2
          (no ACK)                                           |FCN=3,Seq=4
             |-----W=1,FCN=6,Seq=8--X   |                    |FCN=0,Seq=7
   DL Enable |-----W=1,FCN=7,Seq=9----->| Missing Fragment--> W=1
             |<-Compound ACK,W=0,1, C=0-| Bitmap W=0:1010110,|FCN=6,Seq=8
             |-----W=0,FCN=5,Seq=11---->|        W=1:0000001 |__
             |-----W=0,FCN=3,Seq=12---->|
             |-----W=0,FCN=0,Seq=13---->|
             |-----W=1,FCN=6,Seq=14---->| All fragments received
   DL Enable |-----W=1,FCN=7,Seq=15---->|
             |<-Compound ACK, W=1,C=1---| C=1
           (End)
```

Figure 38: Uplink ACK-on-Error All-0 and Other Fragments Lost in the First and Second Windows (2)

Case SCHC ACK is Lost

SCHC over Sigfox does not implement the SCHC ACK REQ message. Instead, it uses the SCHC All-1 message to request a SCHC ACK when required.

```
               Sender                      Receiver
                  |-----W=0,FCN=6,Seq=1----->|
                  |-----W=0,FCN=5,Seq=2----->|
                  |-----W=0,FCN=4,Seq=3----->|
                  |-----W=0,FCN=3,Seq=4----->|
                  |-----W=0,FCN=2,Seq=5----->|
                  |-----W=0,FCN=1,Seq=6----->|
        DL Enable |-----W=0,FCN=0,Seq=7----->|
              (no ACK)
                  |-----W=1,FCN=6,Seq=8----->|
                  |-----W=1,FCN=5,Seq=9----->|
                  |-----W=1,FCN=4,Seq=10---->|
        DL Enable |-----W=1,FCN=7,Seq=11---->| All fragments received
                  | X--Compound ACK,W=1,C=1 -| C=1
        DL Enable |-----W=1,FCN=7,Seq=13---->| RESEND ACK
                  |<-Compound ACK,W=1,C=1 ---| C=1
                (End)
```

Figure 39: Uplink ACK-on-Error ACK Lost

Case SCHC Compound ACK at the End

In this example, SCHC Fragment losses are found in both windows 0 and 1. However, the sender does not send a SCHC Compound ACK after the All-0 of window 0. Instead, it sends a SCHC Compound ACK indicating fragment losses on both windows.

```
            Sender                    Receiver
              |-----W=0,FCN=6,Seq=1----->|
              |-----W=0,FCN=5,Seq=2--X   |
              |-----W=0,FCN=4,Seq=3----->|
              |-----W=0,FCN=3,Seq=4--X   |
              |-----W=0,FCN=2,Seq=5----->|
              |-----W=0,FCN=1,Seq=6----->|                     __
    DL Enable |-----W=0,FCN=0,Seq=7----->| Waits for          |W=0
           (no ACK)                       next DL opportunity |FCN=5,Seq=2
              |-----W=1,FCN=6,Seq=8--X   |                    |FCN=3,Seq=4
    DL Enable |-----W=1,FCN=7,Seq=9----->| Missing Fragment<-- W=1
              |<-Compound ACK,W=0,1, C=0-| Bitmap W=0:1010110 |FCN=6,Seq=8
              |-----W=0,FCN=5,Seq=11---->|        W=1:0000001  --
              |-----W=0,FCN=3,Seq=12---->|
              |-----W=1,FCN=6,Seq=13---->| All fragments received
    DL Enable |-----W=1,FCN=7,Seq=14---->|
              |<-Compound ACK, W=1, C=1 -| C=1
            (End)
```

Figure 40: Uplink ACK-on-Error Fragments Lost in the First and Second Windows with One Compound ACK

The number of times the same SCHC ACK message will be retransmitted is determined by the MAX_ACK_REQUESTS.

5.3. SCHC Abort Examples

Case SCHC Sender-Abort

The sender may need to send a Sender-Abort to stop the current communication. For example, this may happen if the All-1 has been sent MAX_ACK_REQUESTS times.

```
                Sender                    Receiver
                  |-----W=0,FCN=6,Seq=1----->|
                  |-----W=0,FCN=5,Seq=2----->|
                  |-----W=0,FCN=4,Seq=3----->|
                  |-----W=0,FCN=3,Seq=4----->|
                  |-----W=0,FCN=2,Seq=5----->|
                  |-----W=0,FCN=1,Seq=6----->|
        DL Enable |-----W=0,FCN=0,Seq=7----->|
              (no ACK)
                  |-----W=1,FCN=6,Seq=8----->|
                  |-----W=1,FCN=5,Seq=9----->|
                  |-----W=1,FCN=4,Seq=10---->|
        DL Enable |-----W=1,FCN=7,Seq=11---->| All fragments received
                  | X--Compound ACK,W=1,C=1 -| C=1
        DL Enable |-----W=1,FCN=7,Seq=13---->| RESEND ACK  (1)
                  | X--Compound ACK,W=1,C=1 -| C=1
        DL Enable |-----W=1,FCN=7,Seq=15---->| RESEND ACK  (2)
                  | X--Compound ACK,W=1,C=1 -| C=1
        DL Enable |-----W=1,FCN=7,Seq=17---->| RESEND ACK  (3)
                  | X--Compound ACK,W=1,C=1 -| C=1
        DL Enable |-----W=1,FCN=7,Seq=18---->| RESEND ACK  (4)
                  | X--Compound ACK,W=1,C=1 -| C=1
        DL Enable |-----W=1,FCN=7,Seq=19---->| RESEND ACK  (5)
                  | X--Compound ACK,W=1,C=1 -| C=1
        DL Enable |----Sender-Abort,Seq=20-->| exit with error condition
                (End)
```

Figure 41: Uplink ACK-on-Error Sender-Abort

Case Receiver-Abort

The receiver may need to send a Receiver-Abort to stop the current communication. This message can only be sent after a Downlink Enable.

```
                   Sender                     Receiver
                     |-----W=0,FCN=6,Seq=1----->|
                     |-----W=0,FCN=5,Seq=2----->|
                     |-----W=0,FCN=4,Seq=3----->|
                     |-----W=0,FCN=3,Seq=4----->|
                     |-----W=0,FCN=2,Seq=5----->|
                     |-----W=0,FCN=1,Seq=6----->|
           DL Enable |-----W=0,FCN=0,Seq=7----->|
                     |<------  RECV ABORT ------| under-resourced
                  (Error)
```

Figure 42: Uplink ACK-on-Error Receiver-Abort

6. Security Considerations

The radio protocol authenticates and ensures the integrity of each message. This is achieved by using a unique Device ID and an AES-128-based message authentication code, ensuring that the message has been generated and sent by the device (see [sigfox-spec], Section 3.8) or Network (see [sigfox-spec], Section 4.3) with the ID claimed in the message [sigfox-spec].

Application data may or may not be encrypted at the application layer, depending on the criticality of the use case. This flexibility allows a balance between cost and effort versus risk. AES-128 in counter mode is used for encryption. Cryptographic keys are independent for each device. These keys are associated with the Device ID, and separate integrity and encryption keys are pre-provisioned. An encryption key is only provisioned if confidentiality is to be used (see [sigfox-spec], Section 5.3; note that further documentation is available at Sigfox upon request).

The radio protocol has protections against replay attacks, and the cloud-based core Network provides firewall protection against undesired incoming communications [sigfox-spec].

The previously described security mechanisms do not guarantee end-to-end security between the device SCHC C/D + F/R and the Network SCHC C/D + F/R; potential security threats described in [RFC8724] are applicable to the profile specified in this document.

In some circumstances, sending device location information is privacy sensitive. The Device Geolocation parameter provided by the Network is optional; therefore, it can be omitted to protect this aspect of the device privacy.

7. IANA Considerations

This document has no IANA actions.

8. References

8.1. Normative References

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8724] Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC. Zuniga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation", RFC 8724, DOI 10.17487/RFC8724, April 2020, <https://www.rfc-editor.org/info/rfc8724>.

[RFC9441] Zúñiga, JC., Gomez, C., Aguilar, S., Toutain, L., Céspedes, S., and D. Wistuba, "Static Context Header Compression (SCHC) Compound Acknowledgement (ACK)", RFC 9441, DOI 10.17487/RFC9441, July 2023, <https://www.rfc-editor.org/info/rfc9441>.

[sigfox-spec] Sigfox, "Sigfox Device Radio Specifications", <https://build.sigfox.com/sigfox-device-radio-specifications>.

8.2. Informative References

[CORE-COMI] Veillette, M., Ed., van der Stok, P., Ed., Pelov, A., Bierman, A., and C. Bormann, Ed., "CoAP Management Interface (CORECONF)", Work in Progress, Internet-Draft, draft-ietf-core-comi-12, 13 March 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-core-comi-12>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8376] Farrell, S., Ed., "Low-Power Wide Area Network (LPWAN) Overview", RFC 8376, DOI 10.17487/RFC8376, May 2018, <https://www.rfc-editor.org/info/rfc8376>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[sigfox-callbacks] Sigfox, "Sigfox Callbacks", <https://support.sigfox.com/docs/callbacks-documentation>.

[sigfox-docs] Sigfox, "Sigfox Documentation", <https://support.sigfox.com/docs>.

Acknowledgements

Carles Gomez has been funded in part by the Spanish Government through the TEC2016-79988-P grant and the PID2019-106808RA-I00 grant (funded by MCIN / AEI / 10.13039/501100011033) and by Secretaria d'Universitats i Recerca del Departament d'Empresa i Coneixement de la Generalitat de Catalunya through 2017 grant SGR 376 and 2021 grant SGR 00330.

Sergio Aguilar has been funded by the ERDF and the Spanish Government through project TEC2016-79988-P and project PID2019-106808RA-I00, AEI/FEDER, EU (funded by MCIN / AEI / 10.13039/501100011033).

Sandra Cespedes has been funded in part by the ANID Chile Project FONDECYT Regular 1201893 and Basal Project FB0008.

Diego Wistuba has been funded by the ANID Chile Project FONDECYT Regular 1201893.

The authors would like to thank Ana Minaburo, Clement Mannequin, Rafael Vidal, Julien Boite, Renaud Marty, and Antonis Platis for their useful comments and implementation design considerations.

Authors' Addresses

Juan Carlos Zúñiga
Montreal QC
Canada
Email: j.c.zuniga@ieee.org

Carles Gomez
Universitat Politècnica de Catalunya
C/Esteve Terradas, 7
08860 Castelldefels
Spain
Email: carles.gomez@upc.edu

Sergio Aguilar
Universitat Politècnica de Catalunya
C/Esteve Terradas, 7
08860 Castelldefels
Spain
Email: sergio.aguilar.romero@upc.edu

Laurent Toutain
IMT-Atlantique
CS 17607
2 rue de la Chataigneraie
35576 Cesson-Sevigne Cedex
France
Email: Laurent.Toutain@imt-atlantique.fr

Sandra Céspedes
Concordia University
1455 De Maisonneuve Blvd. W.
Montreal QC H3G 1M8
Canada
Email: sandra.cespedes@concordia.ca

Diego Wistuba
NIC Labs, Universidad de Chile
Av. Almte. Blanco Encalada 1975
Santiago
Chile
Email: research@witu.cl

Julien Boite
Unabiz (Sigfox)
Labege
France
Email: juboite@free.fr
URI: https://www.sigfox.com/