Workgroup: lpwan Working Group
Internet-Draft:
draft-ietf-lpwan-schc-yang-data-model-19
Published: 9 October 2022
Intended Status: Standards Track
Expires: 12 April 2023
Authors: A. Minaburo L. Toutain
Acklio Institut MINES TELECOM; IMT Atlantique

Data Model for Static Context Header Compression (SCHC)

Abstract

This document describes a YANG data model for the SCHC (Static Context Header Compression) compression and fragmentation rules.

This document formalizes the description of the rules for better interoperability between SCHC instances either to exchange a set of rules or to modify some rules parameters.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

SCHC is a compression and fragmentation mechanism for constrained networks defined in [RFC8724]. It is based on a static context shared by two entities at the boundary of the constrained network. [RFC8724] provides an informal representation of the rules used either for compression/decompression (or C/D) or fragmentation/reassembly (or F/R). The goal of this document is to formalize the description of the rules to offer:

* the same definition on both ends, even if the internal representation is different;

* an update of the other end to set up some specific values (e.g. IPv6 prefix, destination address,...).

[I-D.ietf-lpwan-architecture] illustrates the exchange of rules using the YANG data model.

This document defines a YANG module [RFC7950] to represent both compression and fragmentation rules, which leads to common representation for values for all the rules elements.

2. Requirements Language

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.

3. Terminology

This section defines the terminology and acronyms used in this document. It extends the terminology of [RFC8376].

* App: LPWAN Application, as defined by [RFC8376]. An application sending/receiving packets to/from the Dev.

* Bi: Bidirectional. Characterizes a Field Descriptor that applies to headers of packets traveling in either direction (Up and Dw, see this glossary).

* CDA: Compression/Decompression Action. Describes the pair of actions that are performed at the compressor to compress a header field and at the decompressor to recover the original value of the header field.

* Context: A set of Rules used to compress/decompress headers.

* Dev: Device, as defined by [RFC8376].

* DevIID: Device Interface Identifier. The IID that identifies the Dev interface.

* DI: Direction Indicator. This field tells which direction of packet travel (Up, Dw or Bi) a Field Description applies to. This allows for asymmetric processing, using the same Rule.

* Dw: Downlink direction for compression/decompression, from SCHC C/D in the network to SCHC C/D in the Dev.

* FID: Field Identifier. This identifies the protocol and field a Field Description applies to.

* FL: Field Length is the length of the original packet header field. It is expressed as a number of bits for header fields of fixed lengths or as a type (e.g., variable, token length, ...) for field lengths that are unknown at the time of Rule creation. The length of a header field is defined in the corresponding protocol specification (such as IPv6 or UDP).

* FP: when a Field is expected to appear multiple times in a header, Field Position specifies the occurrence this Field Description applies to (for example, first uri-path option, second uri-path, etc. in a CoAP header), counting from 1. The value 0 is special and means "don't care", see [RFC8724] Section 7.2.

* IID: Interface Identifier. See the IPv6 addressing architecture [RFC7136].

* L2 Word: this is the minimum subdivision of payload data that the L2 will carry. In most L2 technologies, the L2 Word is an octet. In bit-oriented radio technologies, the L2 Word might be a single bit. The L2 Word size is assumed to be constant over time for each device.

* MO: Matching Operator. An operator used to match a value contained in a header field with a value contained in a Rule.

* Rule ID (Rule Identifier): An identifier for a Rule. SCHC C/D on both sides share the same Rule ID for a given packet. A set of Rule IDs are used to support SCHC F/R functionality.

* TV: Target value. A value contained in a Rule that will be matched with the value of a header field.

* Up: Uplink direction for compression/decompression, from the Dev SCHC C/D to the network SCHC C/D.

4. SCHC rules

SCHC compression is generic, the main mechanism does not refer to a specific protocol. Any header field is abstracted through an Field

Identifier (FID), a position (FP), a direction (DI), and a value that can be a numerical value or a string. [RFC8724] and [RFC8824] specify fields for IPv6 [RFC8200], UDP[RFC0768], CoAP [RFC7252] including options defined for no server response [RFC7967] and OSCORE [RFC8613]. For the latter [RFC8824] splits this field into sub-fields.

SCHC fragmentation requires a set of common parameters that are included in a rule. These parameters are defined in [RFC8724].

The YANG data model enables the compression and the fragmentation selection using the feature statement.

4.1. Compression Rules

[RFC8724] proposes an informal representation of the compression rule. A compression context for a device is composed of a set of rules. Each rule contains information to describe a specific field in the header to be compressed.

```
  +-----------------------------------------------------------------+
  |                      Rule N                                     |
 +-----------------------------------------------------------------+|
 |                    Rule i                                       ||
+-----------------------------------------------------------------+||
|  (FID)            Rule 1                                        |||
|+-------+--+--+--+------------+-----------------+---------------+|||
||Field 1|FL|FP|DI|Target Value|Matching Operator|Comp/Decomp Act||||
|+-------+--+--+--+------------+-----------------+---------------+|||
||Field 2|FL|FP|DI|Target Value|Matching Operator|Comp/Decomp Act||||
|+-------+--+--+--+------------+-----------------+---------------+|||
||...    |..|..|..|   ...      | ...             | ...           ||||
|+-------+--+--+--+------------+-----------------+---------------+||/
||Field N|FL|FP|DI|Target Value|Matching Operator|Comp/Decomp Act|||
|+-------+--+--+--+------------+-----------------+---------------+|/
|                                                                 |
\-----------------------------------------------------------------/
```

Figure 1: Compression Decompression Context

4.2. Identifier generation

Identifiers used in the SCHC YANG data model are from the identityref statement to ensure global uniqueness and easy augmentation if needed. The principle to define a new type based on a group of identityref is the following:

*define a main identity ending with the keyword base-type.

* derive all the identities used in the Data Model from this base type.

* create a typedef from this base type.

The example (Figure 2) shows how an identityref is created for RCS (Reassembly Check Sequence) algorithms used during SCHC fragmentation.

```
identity rcs-algorithm-base-type {
  description
    "Identify which algorithm is used to compute RCS.
     The algorithm also defines the size of the RCS field.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
               Compression and Fragmentation";
}

identity rcs-crc32 {
  base rcs-algorithm-base-type;
  description
    "CRC 32 defined as default RCS in RFC8724. This RCS is
     4 bytes long.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
               Compression and Fragmentation";
}

typedef rcs-algorithm-type {
  type identityref {
    base rcs-algorithm-base-type;
  }
  description
    "Define the type for RCS algorithm in rules.";
}
```

Figure 2: Principle to define a type based on identityref.

4.3. Convention for Field Identifier

In the process of compression, the headers of the original packet are first parsed to create a list of fields. This list of fields is matched against the rules to find the appropriate rule and apply compression. [RFC8724] does not state how the field ID value is constructed. In examples, identification is done through a string indexed by the protocol name (e.g. IPv6.version, CoAP.version,...).

The current YANG data model includes fields definitions found in [RFC8724], [RFC8824].

Using the YANG data model, each field MUST be identified through a global YANG identityref.
A YANG field ID for the protocol is always derived from the fid-base-type. Then an identity for each protocol is specified using the naming convention fid-<<protocol name>>-base-type. All possible fields for this protocol MUST derive from the protocol identity. The naming convention is "fid-" followed by the protocol name and the field name. If a field has to be divided into sub-fields, the field identity serves as a base.

The full field-id definition is found in Section 6. A type is defined for IPv6 protocol, and each field is based on it. Note that the DiffServ bits derive from the Traffic Class identity.

4.4. Convention for Field length

Field length is either an integer giving the size of a field in bits or a specific function. [RFC8724] defines the "var" function which allows variable length fields (whose length is expressed in bytes) and [RFC8824] defines the "tkl" function for managing the CoAP Token length field.

The naming convention is "fl-" followed by the function name.

The field length function can be defined as an identityref as described in Section 6. Therefore, the type for field length is a union between an integer giving the size of the length in bits and the identityref.

4.5. Convention for Field position

Field position is a positive integer which gives the occurrence times of a specific field from the header start. The default value is 1, and incremented at each repetition. Value 0 indicates that the position is not important and is not considered during the rule selection process.

Field position is a positive integer. The type is uint8.

4.6. Convention for Direction Indicator

The Direction Indicator (di) is used to tell if a field appears in both directions (Bi) or only uplink (Up) or Downlink (Dw). The naming convention is "di" followed by the Direction Indicator name.

The type is "di-type".

4.7. Convention for Target Value

The Target Value is a list of binary sequences of any length, aligned to the left. In the rule, the structure will be used as a list, with index as a key. The highest index value is used to compute the size of the index sent in residue for the match-mapping CDA (Compression Decompression Action). The index can specify several values:

* For Equal and MSB, Target Value contains a single element. Therefore, the index is set to 0.

* For match-mapping, Target Value can contain several elements. Index values MUST start from 0 and MUST be contiguous.

If the header field contains text, the binary sequence uses the same encoding.

4.8. Convention for Matching Operator

Matching Operator (MO) is a function applied between a field value provided by the parsed header and the target value. [RFC8724] defines 4 MO.

The naming convention is "mo-" followed by the MO name.

The type is "mo-type"

4.8.1. Matching Operator arguments

They are viewed as a list, built with a tv-struct (see Section 4.7).

4.9. Convention for Compression Decompression Actions

Compression Decompression Action (CDA) identifies the function to use for compression or decompression. [RFC8724] defines 6 CDA.

The naming convention is "cda-" followed by the CDA name.

4.9.1. Compression Decompression Action arguments

Currently no CDA requires arguments, but in the future some CDA may require one or several arguments. They are viewed as a list, of target-value type.

4.10. Fragmentation rule

Fragmentation is optional in the data model and depends on the presence of the "fragmentation" feature.

Most of the fragmentation parameters are listed in Annex D of [RFC8724].

Since fragmentation rules work for a specific direction, they MUST contain a mandatory direction indicator. The type is the same as the one used in compression entries, but bidirectional MUST NOT be used.

4.10.1. Fragmentation mode

[RFC8724] defines 3 fragmentation modes:

* No Ack: this mode is unidirectional, no acknowledgment is sent back.

* Ack Always: each fragmentation window must be explicitly acknowledged before going to the next.

* Ack on Error: A window is acknowledged only when the receiver detects some missing fragments.

The type is "fragmentation-mode-type". The naming convention is "fragmentation-mode-" followed by the fragmentation mode name.

4.10.2. Fragmentation Header

A data fragment header, starting with the rule ID, can be sent in the fragmentation direction. [RFC8724] indicates that the SCHC header may be composed of (cf. Figure 3):

* a Datagram Tag (Dtag) identifying the datagram being fragmented if the fragmentation applies concurrently on several datagrams. This field is optional and its length is defined by the rule.

* a Window (W) used in Ack-Always and Ack-on-Error modes. In Ack-Always, its size is 1. In Ack-on-Error, it depends on the rule. This field is not needed in No-Ack mode.

* a Fragment Compressed Number (FCN) indicating the fragment/tile position within the window. This field is mandatory on all modes defined in [RFC8724], its size is defined by the rule.

```
|-- SCHC Fragment Header ----|
         |-- T --|-M-|-- N --|
+-- ... -+- ... -+---+- ... -+--------...-------+~~~~~~~~~~~~~~~~~~~~
| RuleID | DTag  | W |  FCN  | Fragment Payload | padding (as needed)
+-- ... -+- ... -+---+- ... -+--------...-------+~~~~~~~~~~~~~~~~~~~~
```

Figure 3: Data fragment header from RFC8724

4.10.3. Last fragment format

The last fragment of a datagram is sent with an RCS (Reassembly Check Sequence) field to detect residual transmission error and possible losses in the last window. [RFC8724] defines a single algorithm based on Ethernet CRC computation.

The naming convention is "rcs-" followed by the algorithm name.

For Ack-on-Error mode, the All-1 fragment may just contain the RCS or can include a tile. The parameters define the behavior:

* all-1-data-no: the last fragment contains no data, just the RCS

* all-1-data-yes: the last fragment includes a single tile and the RCS

* all-1-data-sender-choice: the last fragment may or may not contain a single tile. The receiver can detect if a tile is present.

The naming convention is "all-1-data-" followed by the behavior identifier.

4.10.4. Acknowledgment behavior

The acknowledgment fragment header goes in the opposite direction of data. [RFC8724] defines the header, composed of (see Figure 4):

* a Dtag (if present).

* a mandatory window as in the data fragment.

* a C bit giving the status of RCS validation. In case of failure, a bitmap follows, indicating the received tile.

```
|--- SCHC ACK Header ----|
         |-- T --|-M-| 1 |
+-- ... -+- ... -+---+---+~~~~~~~~~~~~~~~~~~
| RuleID |  DTag | W |C=1| padding as needed                (success)
+-- ... -+- ... -+---+---+~~~~~~~~~~~~~~~~~~

+-- ... -+- ... -+---+---+------ ... ------+~~~~~~~~~~~~~~~
| RuleID |  DTag | W |C=0|Compressed Bitmap| pad. as needed (failure)
+-- ... -+- ... -+---+---+------ ... ------+~~~~~~~~~~~~~~~
```

Figure 4: Acknowledgment fragment header for RFC8724

For Ack-on-Error, SCHC defines when an acknowledgment can be sent. This can be at any time defined by the layer 2, at the end of a window (FCN all-0) or as a response to receiving the last fragment (FCN all-1). The naming convention is "ack-behavior" followed by the algorithm name.

4.10.5. Timer values

The state machine requires some common values to handle fragmentation correctly.

* retransmission-timer gives the duration before sending an ack request (cf. section 8.2.2.4. of [RFC8724]). If specified, value MUST be strictly positive.

* inactivity-timer gives the duration before aborting a fragmentation session (cf. section 8.2.2.4. of [RFC8724]). The value 0 explicitly indicates that this timer is disabled.

[RFC8724] do not specify any range for these timers. [RFC9011] recommends a duration of 12 hours. In fact, the value range should be between milliseconds for real time systems to several days. To allow a large range of applications, two parameters must be specified:

* the duration of a tick. It is computed by this formula 2^tick-duration/10^6. When tick-duration is set to 0, the unit is the microsecond. The default value of 20 leads to a unit of 1.048575 second. A value of 32 leads to a tick duration of about 1 hour 11 minutes.

* the number of ticks in the predefined unit. With the default tick-duration value of 20, the timers can cover a range between 1.0 sec and 19 hours covering [RFC9011] recommendation.

4.10.6. Fragmentation Parameter

The SCHC fragmentation protocol specifies the number of attempts before aborting through the parameter:

* max-ack-requests (cf. section 8.2.2.4. of [RFC8724]).

4.10.7. Layer 2 parameters

The data model includes two parameters needed for fragmentation:

* l2-word-size: [RFC8724] base fragmentation, in bits, on a layer 2 word which can be of any length. The default value is 8 and correspond to the default value for byte aligned layer 2. A value

of 1 will indicate that there is no alignment and no need for padding.

* maximum-packet-size: defines the maximum size of an uncompressed datagram. By default, the value is set to 1280 bytes.

They are defined as unsigned integers, see Section 6.

5. Rule definition

A rule is identified by a unique rule identifier (rule ID) comprising both a Rule ID value and a Rule ID length. The YANG grouping rule-id-type defines the structure used to represent a rule ID. A length of 0 is allowed to represent an implicit rule.

Three natures of rules are defined in [RFC8724]:

* Compression: a compression rule is associated with the rule ID.

* No compression: this identifies the default rule used to send a packet integrally when no compression rule was found (see [RFC8724] section 6).

* Fragmentation: fragmentation parameters are associated with the rule ID. Fragmentation is optional and feature "fragmentation" should be set.

The YANG data model introduces respectively these three identities :

* nature-compression

* nature-no-compression

* nature-fragmentation

The naming convention is "nature-" followed by the nature identifier.

To access a specific rule, the rule ID length and value are used as a key. The rule is either a compression or a fragmentation rule.

5.1. Compression rule

A compression rule is composed of entries describing its processing. An entry contains all the information defined in Figure 1 with the types defined above.

The compression rule described Figure 1 is defined by compression-content. It defines a list of compression-rule-entry, indexed by their field id, position and direction. The compression-rule-entry element represent a line of the table Figure 1. Their type reflects the identifier types defined in Section 4.1

Some checks are performed on the values:

* target value MUST be present for MO different from ignore.

* when MSB MO is specified, the matching-operator-value must be present

5.2. Fragmentation rule

A Fragmentation rule is composed of entries describing the protocol behavior. Some on them are numerical entries, others are identifiers defined in Section 4.10.

5.3. YANG Tree

The YANG data model described in this document conforms to the Network Management Datastore Architecture defined in [RFC8342].

```
module: ietf-schc
  +--rw schc
     +--rw rule* [rule-id-value rule-id-length]
        +--rw rule-id-value                   uint32
        +--rw rule-id-length                  uint8
        +--rw rule-nature                     nature-type
        +--rw (nature)?
           +--:(fragmentation) {fragmentation}?
           |  +--rw fragmentation-mode
           |  |       schc:fragmentation-mode-type
           |  +--rw l2-word-size?             uint8
           |  +--rw direction                 schc:di-type
           |  +--rw dtag-size?                uint8
           |  +--rw w-size?                   uint8
           |  +--rw fcn-size                  uint8
           |  +--rw rcs-algorithm?            rcs-algorithm-type
           |  +--rw maximum-packet-size?      uint16
           |  +--rw window-size?              uint16
           |  +--rw max-interleaved-frames?   uint8
           |  +--rw inactivity-timer
           |  |  +--rw ticks-duration?   uint8
           |  |  +--rw ticks-numbers?    uint16
           |  +--rw retransmission-timer
           |  |  +--rw ticks-duration?   uint8
           |  |  +--rw ticks-numbers?    uint16
           |  +--rw max-ack-requests?         uint8
           |  +--rw (mode)?
           |     +--:(no-ack)
           |     +--:(ack-always)
           |     +--:(ack-on-error)
           |        +--rw tile-size?          uint8
           |        +--rw tile-in-all-1?      schc:all-1-data-type
           |        +--rw ack-behavior?       schc:ack-behavior-type
           +--:(compression) {compression}?
              +--rw entry*
                      [field-id field-position direction-indicator]
                 +--rw field-id                    schc:fid-type
                 +--rw field-length                schc:fl-type
                 +--rw field-position              uint8
                 +--rw direction-indicator         schc:di-type
                 +--rw target-value* [index]
                 |  +--rw index    uint16
                 |  +--rw value?   binary
                 +--rw matching-operator           schc:mo-type
                 +--rw matching-operator-value* [index]
                 |  +--rw index    uint16
                 |  +--rw value?   binary
                 +--rw comp-decomp-action          schc:cda-type
                 +--rw comp-decomp-action-value* [index]
```

```
+--rw index    uint16
+--rw value?   binary
```

Figure 5: Overview of SCHC data model

6. YANG Module

```
<CODE BEGINS> file "ietf-schc@2022-10-09.yang"

module ietf-schc {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-schc";
  prefix schc;

  organization
    "IETF IPv6 over Low Power Wide-Area Networks (lpwan) working
     group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/lpwan/about/>
     WG List:  <mailto:lp-wan@ietf.org>
     Editor:   Laurent Toutain
       <mailto:laurent.toutain@imt-atlantique.fr>
     Editor:   Ana Minaburo
       <mailto:ana@ackl.io>";
  description
    "
     Copyright (c) 2022 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject to
     the license terms contained in, the Revised BSD License set
     forth in section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX
     (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
     for full legal notices.

     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 (RFC 2119) (RFC 8174) when, and only when,
     they appear in all capitals, as shown here.

     ***************************************************************

     Generic Data model for Static Context Header Compression Rule
     for SCHC, based on RFC 8724 and RFC8824. Include compression,
     no compression and fragmentation rules.

     This module is a YANG model for SCHC rules (RFC 8724 and
     RFC8824). RFC 8724 describes compression rules in a abstract
     way through a table.

 |-----------------------------------------------------------------|
```

```
|  (FID)            Rule 1                                         |
|+-------+--+--+--+------------+-----------------+---------------+|
||Field 1|FL|FP|DI|Target Value|Matching Operator|Comp/Decomp Act||
|+-------+--+--+--+------------+-----------------+---------------+|
||Field 2|FL|FP|DI|Target Value|Matching Operator|Comp/Decomp Act||
|+-------+--+--+--+------------+-----------------+---------------+|
||...    |..|..|..|   ...      | ...             | ...           ||
|+-------+--+--+--+------------+-----------------+---------------+|
||Field N|FL|FP|DI|Target Value|Matching Operator|Comp/Decomp Act||
|+-------+--+--+--+------------+-----------------+---------------+|
|-----------------------------------------------------------------|

    This module specifies a global data model that can be used for
    rule exchanges or modification. It specifies both the data model
    format and the global identifiers used to describe some
    operations in fields.
    This data model applies to both compression and fragmentation.";

revision 2022-10-09 {
  description
    "Initial version from RFC XXXX.";
  reference
    "RFC XXX: Data Model for Static Context Header Compression
     (SCHC)";
}

feature compression {
  description
    "SCHC compression capabilities are taken into account.";
}

feature fragmentation {
  description
    "SCHC fragmentation capabilities are taken into account.";
}

// -------------------------
//  Field ID type definition
//--------------------------
// generic value TV definition

identity fid-base-type {
  description
    "Field ID base type for all fields.";
}

identity fid-ipv6-base-type {
  base fid-base-type;
  description
```

```
    "Field ID base type for IPv6 headers described in RFC 8200.";
  reference
    "RFC 8200 Internet Protocol, Version 6 (IPv6) Specification";
}

identity fid-ipv6-version {
  base fid-ipv6-base-type;
  description
    "IPv6 version field.";
  reference
    "RFC 8200 Internet Protocol, Version 6 (IPv6) Specification";
}

identity fid-ipv6-trafficclass {
  base fid-ipv6-base-type;
  description
    "IPv6 Traffic Class field.";
  reference
    "RFC 8200 Internet Protocol, Version 6 (IPv6) Specification";
}

identity fid-ipv6-trafficclass-ds {
  base fid-ipv6-trafficclass;
  description
    "IPv6 Traffic Class field: DiffServ field.";
  reference
    "RFC 8200 Internet Protocol, Version 6 (IPv6) Specification,
     RFC 3168 The Addition of Explicit Congestion Notification
              (ECN) to IP";
}

identity fid-ipv6-trafficclass-ecn {
  base fid-ipv6-trafficclass;
  description
    "IPv6 Traffic Class field: ECN field.";
  reference
    "RFC 8200 Internet Protocol, Version 6 (IPv6) Specification,
     RFC 3168 The Addition of Explicit Congestion Notification
              (ECN) to IP";
}

identity fid-ipv6-flowlabel {
  base fid-ipv6-base-type;
  description
    "IPv6 Flow Label field.";
  reference
    "RFC 8200 Internet Protocol, Version 6 (IPv6) Specification";
}
```

```
identity fid-ipv6-payload-length {
  base fid-ipv6-base-type;
  description
    "IPv6 Payload Length field.";
  reference
    "RFC 8200 Internet Protocol, Version 6 (IPv6) Specification";
}

identity fid-ipv6-nextheader {
  base fid-ipv6-base-type;
  description
    "IPv6 Next Header field.";
  reference
    "RFC 8200 Internet Protocol, Version 6 (IPv6) Specification";
}

identity fid-ipv6-hoplimit {
  base fid-ipv6-base-type;
  description
    "IPv6 Next Header field.";
  reference
    "RFC 8200 Internet Protocol, Version 6 (IPv6) Specification";
}

identity fid-ipv6-devprefix {
  base fid-ipv6-base-type;
  description
    "Corresponds to either the source address or the destination
     address prefix of RFC 8200 depending on whether it is an
     uplink or a downlink message.";
  reference
    "RFC 8200 Internet Protocol, Version 6 (IPv6) Specification";
}

identity fid-ipv6-deviid {
  base fid-ipv6-base-type;
  description
    "Corresponds to either the source address or the destination
     address IID of RFC 8200 depending on whether it is an uplink
     or a downlink message.";
  reference
    "RFC 8200 Internet Protocol, Version 6 (IPv6) Specification";
}

identity fid-ipv6-appprefix {
  base fid-ipv6-base-type;
  description
    "Corresponds to either the source address or the destination
     address prefix of RFC 8200 depending on whether it is an
```

```
     uplink or a downlink message.";
  reference
    "RFC 8200 Internet Protocol, Version 6 (IPv6) Specification";
}

identity fid-ipv6-appiid {
  base fid-ipv6-base-type;
  description
    "Corresponds to either the source address or the destination
     address IID of RFC 8200 depending on whether it is an uplink
     or a downlink message.";
  reference
    "RFC 8200 Internet Protocol, Version 6 (IPv6) Specification";
}

identity fid-udp-base-type {
  base fid-base-type;
  description
    "Field ID base type for UDP headers described in RFC 768.";
  reference
    "RFC 768 User Datagram Protocol";
}

identity fid-udp-dev-port {
  base fid-udp-base-type;
  description
    "UDP source or destination port, if uplink or downlink
     communication, respectively.";
  reference
    "RFC 768 User Datagram Protocol";
}

identity fid-udp-app-port {
  base fid-udp-base-type;
  description
    "UDP destination or source port, if uplink or downlink
     communication, respectively.";
  reference
    "RFC 768 User Datagram Protocol";
}

identity fid-udp-length {
  base fid-udp-base-type;
  description
    "UDP length.";
  reference
    "RFC 768 User Datagram Protocol";
}
```

```
identity fid-udp-checksum {
  base fid-udp-base-type;
  description
    "UDP length.";
  reference
    "RFC 768 User Datagram Protocol";
}

identity fid-coap-base-type {
  base fid-base-type;
  description
    "Field ID base type for UDP headers described.";
  reference
    "RFC 7252 The Constrained Application Protocol (CoAP)";
}

identity fid-coap-version {
  base fid-coap-base-type;
  description
    "CoAP version.";
  reference
    "RFC 7252 The Constrained Application Protocol (CoAP)";
}

identity fid-coap-type {
  base fid-coap-base-type;
  description
    "CoAP type.";
  reference
    "RFC 7252 The Constrained Application Protocol (CoAP)";
}

identity fid-coap-tkl {
  base fid-coap-base-type;
  description
    "CoAP token length.";
  reference
    "RFC 7252 The Constrained Application Protocol (CoAP)";
}

identity fid-coap-code {
  base fid-coap-base-type;
  description
    "CoAP code.";
  reference
    "RFC 7252 The Constrained Application Protocol (CoAP)";
}

identity fid-coap-code-class {
```

```
    base fid-coap-code;
    description
      "CoAP code class.";
    reference
      "RFC 7252 The Constrained Application Protocol (CoAP)";
  }

  identity fid-coap-code-detail {
    base fid-coap-code;
    description
      "CoAP code detail.";
    reference
      "RFC 7252 The Constrained Application Protocol (CoAP)";
  }

  identity fid-coap-mid {
    base fid-coap-base-type;
    description
      "CoAP message ID.";
    reference
      "RFC 7252 The Constrained Application Protocol (CoAP)";
  }

  identity fid-coap-token {
    base fid-coap-base-type;
    description
      "CoAP token.";
    reference
      "RFC 7252 The Constrained Application Protocol (CoAP)";
  }

  identity fid-coap-option-if-match {
    base fid-coap-base-type;
    description
      "CoAP option If-Match.";
    reference
      "RFC 7252 The Constrained Application Protocol (CoAP)";
  }

  identity fid-coap-option-uri-host {
    base fid-coap-base-type;
    description
      "CoAP option URI-Host.";
    reference
      "RFC 7252 The Constrained Application Protocol (CoAP)";
  }

  identity fid-coap-option-etag {
    base fid-coap-base-type;
```

```
    description
      "CoAP option Etag.";
    reference
      "RFC 7252 The Constrained Application Protocol (CoAP)";
  }

  identity fid-coap-option-if-none-match {
    base fid-coap-base-type;
    description
      "CoAP option if-none-match.";
    reference
      "RFC 7252 The Constrained Application Protocol (CoAP)";
  }

  identity fid-coap-option-observe {
    base fid-coap-base-type;
    description
      "CoAP option Observe.";
    reference
      "RFC 7252 The Constrained Application Protocol (CoAP)";
  }

  identity fid-coap-option-uri-port {
    base fid-coap-base-type;
    description
      "CoAP option Uri-Port.";
    reference
      "RFC 7252 The Constrained Application Protocol (CoAP)";
  }

  identity fid-coap-option-location-path {
    base fid-coap-base-type;
    description
      "CoAP option Location-Path.";
    reference
      "RFC 7252 The Constrained Application Protocol (CoAP)";
  }

  identity fid-coap-option-uri-path {
    base fid-coap-base-type;
    description
      "CoAP option Uri-Path.";
    reference
      "RFC 7252 The Constrained Application Protocol (CoAP)";
  }

  identity fid-coap-option-content-format {
    base fid-coap-base-type;
    description
```

```
    "CoAP option Content Format.";
  reference
    "RFC 7252 The Constrained Application Protocol (CoAP)";
}

identity fid-coap-option-max-age {
  base fid-coap-base-type;
  description
    "CoAP option Max-Age.";
  reference
    "RFC 7252 The Constrained Application Protocol (CoAP)";
}

identity fid-coap-option-uri-query {
  base fid-coap-base-type;
  description
    "CoAP option Uri-Query.";
  reference
    "RFC 7252 The Constrained Application Protocol (CoAP)";
}

identity fid-coap-option-accept {
  base fid-coap-base-type;
  description
    "CoAP option Accept.";
  reference
    "RFC 7252 The Constrained Application Protocol (CoAP)";
}

identity fid-coap-option-location-query {
  base fid-coap-base-type;
  description
    "CoAP option Location-Query.";
  reference
    "RFC 7252 The Constrained Application Protocol (CoAP)";
}

identity fid-coap-option-block2 {
  base fid-coap-base-type;
  description
    "CoAP option Block2.";
  reference
    "RFC 7959 Block-Wise Transfers in the Constrained
              Application Protocol (CoAP)";
}

identity fid-coap-option-block1 {
  base fid-coap-base-type;
  description
```

```
    "CoAP option Block1.";
  reference
    "RFC 7959 Block-Wise Transfers in the Constrained
              Application Protocol (CoAP)";
}

identity fid-coap-option-size2 {
  base fid-coap-base-type;
  description
    "CoAP option size2.";
  reference
    "RFC 7959 Block-Wise Transfers in the Constrained
              Application Protocol (CoAP)";
}

identity fid-coap-option-proxy-uri {
  base fid-coap-base-type;
  description
    "CoAP option Proxy-Uri.";
  reference
    "RFC 7252 The Constrained Application Protocol (CoAP)";
}

identity fid-coap-option-proxy-scheme {
  base fid-coap-base-type;
  description
    "CoAP option Proxy-scheme.";
  reference
    "RFC 7252 The Constrained Application Protocol (CoAP)";
}

identity fid-coap-option-size1 {
  base fid-coap-base-type;
  description
    "CoAP option Size1.";
  reference
    "RFC 7252 The Constrained Application Protocol (CoAP)";
}

identity fid-coap-option-no-response {
  base fid-coap-base-type;
  description
    "CoAP option No response.";
  reference
    "RFC 7967 Constrained Application Protocol (CoAP)
              Option for No Server Response";
}

identity fid-oscore-base-type {
```

```
  base fid-coap-type;
  description
    "OSCORE options (RFC8613) split in sub options.";
  reference
    "RFC 8824 Static Context Header Compression (SCHC) for the
              Constrained Application Protocol (CoAP)";
}

identity fid-coap-option-oscore-flags {
  base fid-oscore-base-type;
  description
    "CoAP option oscore flags.";
  reference
    "RFC 8824 Static Context Header Compression (SCHC) for the
              Constrained Application Protocol (CoAP) (see
              section 6.4)";
}

identity fid-coap-option-oscore-piv {
  base fid-oscore-base-type;
  description
    "CoAP option oscore flags.";
  reference
    "RFC 8824 Static Context Header Compression (SCHC) for the
              Constrained Application Protocol (CoAP) (see
              section 6.4)";
}

identity fid-coap-option-oscore-kid {
  base fid-oscore-base-type;
  description
    "CoAP option oscore flags.";
  reference
    "RFC 8824 Static Context Header Compression (SCHC) for the
              Constrained Application Protocol (CoAP) (see
              section 6.4)";
}

identity fid-coap-option-oscore-kidctx {
  base fid-oscore-base-type;
  description
    "CoAP option oscore flags.";
  reference
    "RFC 8824 Static Context Header Compression (SCHC) for the
              Constrained Application Protocol (CoAP)(see
              section 6.4)";
}

//----------------------------------
```

```
// Field Length type definition
//---------------------------------

identity fl-base-type {
  description
    "Used to extend field length functions.";
}

identity fl-variable {
  base fl-base-type;
  description
    "Residue length in Byte is sent as defined for CoAP.";
  reference
    "RFC 8824 Static Context Header Compression (SCHC) for the
              Constrained Application Protocol (CoAP) (see
              section 5.3)";
}

identity fl-token-length {
  base fl-base-type;
  description
    "Residue length in Byte is sent as defined for CoAP.";
  reference
    "RFC 8824 Static Context Header Compression (SCHC) for the
              Constrained Application Protocol (CoAP) (see
              section 4.5)";
}

//---------------------------------
// Direction Indicator type
//---------------------------------

identity di-base-type {
  description
    "Used to extend direction indicators.";
}

identity di-bidirectional {
  base di-base-type;
  description
    "Direction Indication of bidirectionality.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context
              Header Compression and Fragmentation (see
              section 7.1.)";
}

identity di-up {
  base di-base-type;
```

```
    description
      "Direction Indication of uplink.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context
                Header Compression and Fragmentation (see
                section 7.1).";
  }

  identity di-down {
    base di-base-type;
    description
      "Direction Indication of downlink.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context
                Header Compression and Fragmentation (see
                section 7.1).";
  }

  //----------------------------------
  // Matching Operator type definition
  //----------------------------------

  identity mo-base-type {
    description
      "Matching Operator: used in the rule selection process
       to check is a Target Value matches the field's value.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context
                Header Compression and Fragmentation (see*
                section 7.2).";
  }

  identity mo-equal {
    base mo-base-type;
    description
      "equal MO.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context
                Header Compression and Fragmentation (see
                section 7.3).";
  }

  identity mo-ignore {
    base mo-base-type;
    description
      "ignore MO.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context
                Header Compression and Fragmentation (see
```

```
                section 7.3).";
  }

  identity mo-msb {
    base mo-base-type;
    description
      "MSB MO.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context
                Header Compression and Fragmentation (see
                section 7.3).";
  }

  identity mo-match-mapping {
    base mo-base-type;
    description
      "match-mapping MO.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context
                Header Compression and Fragmentation (see
                section 7.3).";
  }

  //----------------------------------
  // CDA type definition
  //----------------------------------

  identity cda-base-type {
    description
      "Compression Decompression Actions. Specify the action to
       be applied to the field's value in a specific rule.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context
                Header Compression and Fragmentation (see
                section 7.2).";
  }

  identity cda-not-sent {
    base cda-base-type;
    description
      "not-sent CDA.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context
                Header Compression and Fragmentation (see
                section 7.4).";
  }

  identity cda-value-sent {
    base cda-base-type;
```

```
    description
      "value-sent CDA.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context
                Header Compression and Fragmentation (see
                section 7.4).";
  }

  identity cda-lsb {
    base cda-base-type;
    description
      "LSB CDA.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context
                Header Compression and Fragmentation (see
                section 7.4).";
  }

  identity cda-mapping-sent {
    base cda-base-type;
    description
      "mapping-sent CDA.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context
                Header Compression and Fragmentation (see
                section 7.4).";
  }

  identity cda-compute {
    base cda-base-type;
    description
      "compute-* CDA.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context
                Header Compression and Fragmentation (see
                section 7.4).";
  }

  identity cda-deviid {
    base cda-base-type;
    description
      "DevIID CDA.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context
                Header Compression and Fragmentation (see
                section 7.4).";
  }

  identity cda-appiid {
```

```
  base cda-base-type;
  description
    "AppIID CDA.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context
               Header Compression and Fragmentation (see
               section 7.4).";
}

// -- type definition

typedef fid-type {
  type identityref {
    base fid-base-type;
  }
  description
    "Field ID generic type.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
               Compression and Fragmentation";
}

typedef fl-type {
  type union {
    type uint64 {
      range 1..max;
    }
    type identityref {
      base fl-base-type;
    }
  }
  description
    "Field length either a positive integer expressing the size in
     bits or a function defined through an identityref.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
               Compression and Fragmentation";
}

typedef di-type {
  type identityref {
    base di-base-type;
  }
  description
    "Direction in LPWAN network, up when emitted by the device,
     down when received by the device, bi when emitted or
     received by the device.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
```

```
               Compression and Fragmentation";
}

typedef mo-type {
  type identityref {
    base mo-base-type;
  }
  description
    "Matching Operator (MO) to compare fields values with
     target values.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
               Compression and Fragmentation";
}

typedef cda-type {
  type identityref {
    base cda-base-type;
  }
  description
    "Compression Decompression Action to compression or
     decompress a field.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
               Compression and Fragmentation";
}

// -- FRAGMENTATION TYPE
// -- fragmentation modes

identity fragmentation-mode-base-type {
  description
    "Define the fragmentation mode.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
               Compression and Fragmentation";
}

identity fragmentation-mode-no-ack {
  base fragmentation-mode-base-type;
  description
    "No-ACK mode.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
               Compression and Fragmentation";
}

identity fragmentation-mode-ack-always {
  base fragmentation-mode-base-type;
```

```
  description
    "ACK-Always mode.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
              Compression and Fragmentation";
}

identity fragmentation-mode-ack-on-error {
  base fragmentation-mode-base-type;
  description
    "ACK-on-Error mode.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
              Compression and Fragmentation";
}

typedef fragmentation-mode-type {
  type identityref {
    base fragmentation-mode-base-type;
  }
  description
    "Define the type used for fragmentation mode in rules.";
}

// -- Ack behavior

identity ack-behavior-base-type {
  description
    "Define when to send an Acknowledgment .";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
              Compression and Fragmentation";
}

identity ack-behavior-after-all-0 {
  base ack-behavior-base-type;
  description
    "Fragmentation expects Ack after sending All-0 fragment.";
}

identity ack-behavior-after-all-1 {
  base ack-behavior-base-type;
  description
    "Fragmentation expects Ack after sending All-1 fragment.";
}

identity ack-behavior-by-layer2 {
  base ack-behavior-base-type;
  description
```

```
    "Layer 2 defines when to send an Ack.";
}

typedef ack-behavior-type {
  type identityref {
    base ack-behavior-base-type;
  }
  description
    "Define the type used for Ack behavior in rules.";
}

// -- All-1 with data types

identity all-1-data-base-type {
  description
    "Type to define when to send an Acknowledgment message.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
              Compression and Fragmentation";
}

identity all-1-data-no {
  base all-1-data-base-type;
  description
    "All-1 contains no tiles.";
}

identity all-1-data-yes {
  base all-1-data-base-type;
  description
    "All-1 MUST contain a tile.";
}

identity all-1-data-sender-choice {
  base all-1-data-base-type;
  description
    "Fragmentation process chooses to send tiles or not in All-1.";
}

typedef all-1-data-type {
  type identityref {
    base all-1-data-base-type;
  }
  description
    "Define the type used for All-1 format in rules.";
}

// -- RCS algorithm types

identity rcs-algorithm-base-type {
```

```
    description
      "Identify which algorithm is used to compute RCS.
       The algorithm also defines the size of the RCS field.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context Header
                 Compression and Fragmentation";
  }

  identity rcs-crc32 {
    base rcs-algorithm-base-type;
    description
      "CRC 32 defined as default RCS in RFC8724. This RCS is
       4 bytes long.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context Header
                 Compression and Fragmentation";
  }

  typedef rcs-algorithm-type {
    type identityref {
      base rcs-algorithm-base-type;
    }
    description
      "Define the type for RCS algorithm in rules.";
  }

  // --------  RULE ENTRY DEFINITION ------------

  grouping tv-struct {
    description
      "Defines the target value element. If the header field
       contains a text, the binary sequence uses the same encoding.
       field-id allows the conversion to the appropriate type.";
    leaf index {
      type uint16;
      description
        "Index gives the position in the matching-list. If only one
         element is present, index is 0. Otherwise, index is the
         the order in the matching list, starting at 0.";
    }
    leaf value {
      type binary;
      description
        "Target Value content as an untyped binary value.";
    }
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context Header
                 Compression and Fragmentation";
  }
```

```
grouping compression-rule-entry {
  description
    "These entries defines a compression entry (i.e. a line)
     as defined in RFC 8724.

 +-------+--+--+--+------------+-----------------+---------------+
 |Field 1|FL|FP|DI|Target Value|Matching Operator|Comp/Decomp Act|
 +-------+--+--+--+------------+-----------------+---------------+

     An entry in a compression rule is composed of 7 elements:
     - Field ID: The header field to be compressed.
     - Field Length : Either a positive integer of a function.
     - Field Position: A positive (and possibly equal to 0)
       integer.
     - Direction Indicator: An indication in which direction
       compression and decompression process is effective.
     - Target value: A value against which the header Field is
       compared.
     - Matching Operator: The comparison operation and optional
       associate parameters.
     - Comp./Decomp. Action: The compression or decompression
       action, and optional parameters.
    ";
  leaf field-id {
    type schc:fid-type;
    mandatory true;
    description
      "Field ID, identify a field in the header with a YANG
       identity reference.";
  }
  leaf field-length {
    type schc:fl-type;
    mandatory true;
    description
      "Field Length, expressed in number of bits if the length is
       known when the Rule is created or through a specific
       function if the length is variable.";
  }
  leaf field-position {
    type uint8;
    mandatory true;
    description
      "Field position in the header is an integer. Position 1
       matches the first occurrence of a field in the header,
       while incremented position values match subsequent
       occurrences.
       Position 0 means that this entry matches a field
       irrespective of its position of occurrence in the
```

```
        header.
        Be aware that the decompressed header may have
        position-0 fields ordered differently than they
        appeared in the original packet.";
   }
   leaf direction-indicator {
     type schc:di-type;
     mandatory true;
     description
       "Direction Indicator, indicate if this field must be
        considered for rule selection or ignored based on the
        direction (bi directionnal, only uplink, or only
        downlink).";
   }
   list target-value {
     key "index";
     uses tv-struct;
     description
       "A list of value to compare with the header field value.
        If target value is a singleton, position must be 0.
        For use as a matching list for the mo-match-mapping matching
        operator, index should take consecutive values starting
        from 0.";
   }
   leaf matching-operator {
     type schc:mo-type;
     must "../target-value or derived-from-or-self(.,
                                                   'mo-ignore')" {
       error-message
         "mo-equal, mo-msb and mo-match-mapping need target-value";
       description
         "target-value is not required for mo-ignore.";
     }
     must "not (derived-from-or-self(., 'mo-msb')) or
           ../matching-operator-value" {
       error-message "mo-msb requires length value";
     }
     mandatory true;
     description
       "MO: Matching Operator.";
     reference
       "RFC 8724 SCHC: Generic Framework for Static Context Header
                 Compression and Fragmentation (see Section 7.3).";
   }
   list matching-operator-value {
     key "index";
     uses tv-struct;
     description
       "Matching Operator Arguments, based on TV structure to allow
```

```
          several arguments.
          In RFC 8724, only the MSB matching operator needs arguments
          (a single argument, which is the number of most significant
          bits to be matched).";
     }
     leaf comp-decomp-action {
       type schc:cda-type;
       must "../target-value or
                 derived-from-or-self(., 'cda-value-sent') or
                 derived-from-or-self(., 'cda-compute') or
                 derived-from-or-self(., 'cda-appiid') or
                 derived-from-or-self(., 'cda-deviid')" {
         error-message
           "cda-not-sent, cda-lsb, cda-mapping-sent need
            target-value";
         description
           "target-value is not required for some CDA.";
         }
       mandatory true;
       description
         "CDA: Compression Decompression Action.";
       reference
         "RFC 8724 SCHC: Generic Framework for Static Context Header
                   Compression and Fragmentation (see section 7.4)";
     }
     list comp-decomp-action-value {
       key "index";
       uses tv-struct;
       description
         "CDA arguments, based on a TV structure, in order to allow
          for several arguments. The CDAs specified in RFC 8724
          require no argument.";
     }

   }

   // --Rule nature

   identity nature-base-type {
     description
       "A rule, identified by its RuleID, are used for a single
        purpose. RFC 8724 defines 2 natures:
        compression, no compression and fragmentation.";
     reference
       "RFC 8724 SCHC: Generic Framework for Static Context Header
                 Compression and Fragmentation (see section 6).";
   }

   identity nature-compression {
```

```
  base nature-base-type;
  description
    "Identify a compression rule.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
              Compression and Fragmentation (see section 6).";
}

identity nature-no-compression {
  base nature-base-type;
  description
    "Identify a no compression rule.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
              Compression and Fragmentation (see section 6).";
}

identity nature-fragmentation {
  base nature-base-type;
  description
    "Identify a fragmentation rule.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
              Compression and Fragmentation (see section 6).";
}

typedef nature-type {
  type identityref {
    base nature-base-type;
  }
  description
    "defines the type to indicate the nature of the rule.";
}

grouping compression-content {
  list entry {
    must "derived-from-or-self(../rule-nature,
                                    'nature-compression')" {
      error-message "Rule nature must be compression";
    }
    key "field-id field-position direction-indicator";
    uses compression-rule-entry;
    description
      "A compression rule is a list of rule entries, each
       describing a header field. An entry is identified
       through a field-id, its position in the packet, and
       its direction.";
  }
  description
```

```
      "Define a compression rule composed of a list of entries.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context Header
                Compression and Fragmentation";
  }

  grouping fragmentation-content {
    description
      "This grouping defines the fragmentation parameters for
       all the modes (No-ACK, ACK-Always and ACK-on-Error) specified
       in RFC 8724.";
    leaf fragmentation-mode {
      type schc:fragmentation-mode-type;
      must "derived-from-or-self(../rule-nature,
                                   'nature-fragmentation')" {
        error-message "Rule nature must be fragmentation";
      }
      mandatory true;
      description
        "Which fragmentation mode is used (No-Ack, ACK-Always,
         ACK-on-Error).";
    }
    leaf l2-word-size {
      type uint8;
      default "8";
      description
        "Size, in bits, of the layer 2 word.";
    }
    leaf direction {
      type schc:di-type;
      must "derived-from-or-self(., 'di-up') or
            derived-from-or-self(., 'di-down')" {
        error-message
          "Direction for fragmentation rules are up or down.";
      }
      mandatory true;
      description
        "MUST be up or down, bidirectional MUST NOT be used.";
    }
    // SCHC Frag header format
    leaf dtag-size {
      type uint8;
      default "0";
      description
        "Size, in bits, of the DTag field (T variable from
         RFC8724).";
    }
    leaf w-size {
      when "derived-from-or-self(../fragmentation-mode,
```

```
                           'fragmentation-mode-ack-on-error')
        or
        derived-from-or-self(../fragmentation-mode,
                           'fragmentation-mode-ack-always') ";
  type uint8;
  description
    "Size, in bits, of the window field (M variable from
     RFC8724).";
}
leaf fcn-size {
  type uint8;
  mandatory true;
  description
    "Size, in bits, of the FCN field (N variable from RFC8724).";
}
leaf rcs-algorithm {
  type rcs-algorithm-type;
  default "schc:rcs-crc32";
  description
    "Algorithm used for RCS. The algorithm specifies the RCS
     size.";
}
// SCHC fragmentation protocol parameters
leaf maximum-packet-size {
  type uint16;
  default "1280";
  description
    "When decompression is done, packet size must not
     strictly exceed this limit, expressed in bytes.";
}
leaf window-size {
  type uint16;
  description
    "By default, if not specified 2^w-size - 1. Should not exceed
     this value. Possible FCN values are between 0 and
     window-size - 1.";
}
leaf max-interleaved-frames {
  type uint8;
  default "1";
  description
    "Maximum of simultaneously fragmented frames. Maximum value
     is 2^dtag-size. All DTAG values can be used, but more than
     max-interleaved-frames MUST NOT be active at any time";
}
container inactivity-timer {
  leaf ticks-duration {
    type uint8;
    default "20";
```

```
      description
        "Duration of one tick in micro-seconds:
            2^ticks-duration/10^6 = 1.048s.";
    }
    leaf ticks-numbers {
      type uint16 {
        range "0..max";
      }
      description
        "Timer duration = ticks-numbers*2^ticks-duration / 10^6.";
    }

    description
      "Duration is seconds of the inactivity timer, 0 indicates
       that the timer is disabled.

       Allows a precision from micro-second to year by sending the
       tick-duration value. For instance:

     tick-duration /  smallest value          highest value
     v
     20: 00y 000d 00h 00m 01s.048575<->00y 000d 19h 05m 18s.428159
     21: 00y 000d 00h 00m 02s.097151<->00y 001d 14h 10m 36s.856319
     22: 00y 000d 00h 00m 04s.194303<->00y 003d 04h 21m 13s.712639
     23: 00y 000d 00h 00m 08s.388607<->00y 006d 08h 42m 27s.425279
     24: 00y 000d 00h 00m 16s.777215<->00y 012d 17h 24m 54s.850559
     25: 00y 000d 00h 00m 33s.554431<->00y 025d 10h 49m 49s.701119

      Note that the smallest value is also the incrementation step,
      so the timer precision.";
  }
  container retransmission-timer {
    leaf ticks-duration {
      type uint8;
      default "20";
      description
        "Duration of one tick in micro-seconds:
            2^ticks-duration/10^6 = 1.048s.";
    }
    leaf ticks-numbers {
      type uint16 {
        range "1..max";
      }
      description
        "Timer duration = ticks-numbers*2^ticks-duration / 10^6.";
    }

    when "derived-from-or-self(../fragmentation-mode,
                              'fragmentation-mode-ack-on-error')
```

```
        or
        derived-from-or-self(../fragmentation-mode,
                            'fragmentation-mode-ack-always') ";
  description
    "Duration in seconds of the retransmission timer.
     See inactivity timer.";
}
leaf max-ack-requests {
  when "derived-from-or-self(../fragmentation-mode,
                            'fragmentation-mode-ack-on-error')
        or
        derived-from-or-self(../fragmentation-mode,
                            'fragmentation-mode-ack-always') ";
  type uint8 {
    range "1..max";
  }
  description
    "The maximum number of retries for a specific SCHC ACK.";
}
choice mode {
  case no-ack;
  case ack-always;
  case ack-on-error {
    leaf tile-size {
      when "derived-from-or-self(../fragmentation-mode,
                         'fragmentation-mode-ack-on-error')";
      type uint8;
      description
        "Size, in bits, of tiles. If not specified or set to 0,
         tiles fill the fragment.";
    }
    leaf tile-in-all-1 {
      when "derived-from-or-self(../fragmentation-mode,
                         'fragmentation-mode-ack-on-error')";
      type schc:all-1-data-type;
      description
        "Defines whether the sender and receiver expect a tile in
         All-1 fragments or not, or if it is left to the sender's
         choice.";
    }
    leaf ack-behavior {
      when "derived-from-or-self(../fragmentation-mode,
                         'fragmentation-mode-ack-on-error')";
      type schc:ack-behavior-type;
      description
        "Sender behavior to acknowledge, after All-0, All-1 or
         when the LPWAN allows it.";
    }
  }
```

```
    description
      "RFC 8724 defines 3 fragmentation modes.";
  }
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
              Compression and Fragmentation";
}

// Define rule ID. Rule ID is composed of a RuleID value and a
// Rule ID Length

grouping rule-id-type {
  leaf rule-id-value {
    type uint32;
    description
      "Rule ID value, this value must be unique, considering its
       length.";
  }
  leaf rule-id-length {
    type uint8 {
      range "0..32";
    }
    description
      "Rule ID length, in bits. The value 0 is for implicit
       rules.";
  }
  description
    "A rule ID is composed of a value and a length, expressed in
     bits.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
              Compression and Fragmentation";
}

// SCHC table for a specific device.

container schc {
  list rule {
    key "rule-id-value rule-id-length";
    uses rule-id-type;
    leaf rule-nature {
      type nature-type;
      mandatory true;
      description
        "Specify the rule's nature.";
    }
    choice nature {
      case fragmentation {
        if-feature "fragmentation";
```

```
          uses fragmentation-content;
        }
        case compression {
          if-feature "compression";
          uses compression-content;
        }
        description
          "A rule is for compression, for no-compression or for
           fragmentation.";
      }
      description
        "Set of rules compression, no compression or fragmentation
         rules identified by their rule-id.";
    }
    description
      "A SCHC set of rules is composed of a list of rules which are
       used for compression, no-compression or fragmentation.";
    reference
      "RFC 8724 SCHC: Generic Framework for Static Context Header
                Compression and Fragmentation";
  }
}

<CODE ENDS>
```

Figure 6

7. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

* Openschc is implementing the conversion between the local rule representation and the representation conforming to the data model in JSON and CBOR (following -08 draft).

8. IANA Considerations

This document registers one URI and one YANG modules.

8.1. URI Registration

This document requests IANA to register the following URI in the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-schc

Registrant Contact: The IESG.

XML: N/A; the requested URI is an XML namespace.

8.2. YANG Module Name Registration

This document registers the following one YANG modules in the "YANG Module Names" registry [RFC6020].

name: ietf-schc

namespace: urn:ietf:params:xml:ns:yang:ietf-schc

prefix: schc

reference: RFC XXXX Data Model for Static Context Header Compression (SCHC)

9. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

This data model formalizes the rules elements described in [RFC8724] for compression, and fragmentation. As explained in the architecture document [I-D.ietf-lpwan-architecture], a rule can be read, created, updated or deleted in response to a management request. These actions can be done between two instances of SCHC or between a SCHC instance and a rule repository.

```
                  create
       (-------)  read   +=======+ *
       ( rules )<------->|Rule   |<--|-------->
       (-------)  update |Manager|   NETCONF, RESTCONF,...
          . read  delete +=======+   request
          .
       +-------+
   <===| R & D |<===
   ===>| C & F |===>
       +-------+
```

The rule contains sensitive information such as the application IPv6 address where the device's data will be sent after decompression. A device may try to modify other devices' rules by changing the application address and may block communication or allows traffic eavesdropping. Therefore, a device must be allowed to modify only its own rules on the remote SCHC instance. The identity of the requester must be validated. This can be done through certificates or access lists. By reading a module, an attacker may know the traffic a device can generate and learn about application addresses or REST API.

The full tree is sensitive, since it represents all the elements that can be managed. This module aims to be encapsulated into a YANG module including access controls and identities.

10. Annex A : Example

The informal rules given Figure 7 will represented in XML as shown in Figure 8.

```
/-------------------------\
|Rule 6/3            110  |
|---------------+---+--+--+----------------+-------+----------------\
|IPV6.VER       |  4| 1|BI|               6|EQUAL  |NOT-SENT        |
|IPV6.TC        |  8| 1|BI|               0|EQUAL  |NOT-SENT        |
|IPV6.FL        | 20| 1|BI|               0|IGNORE |NOT-SENT        |
|IPV6.LEN       | 16| 1|BI|                |IGNORE |COMPUTE-LENGTH  |
|IPV6.NXT       |  8| 1|BI|              58|EQUAL  |NOT-SENT        |
|IPV6.HOP_LMT   |  8| 1|BI|             255|IGNORE |NOT-SENT        |
|IPV6.DEV_PREFIX| 64| 1|BI|200104701f2101d2|EQUAL  |NOT-SENT        |
|IPV6.DEV_IID   | 64| 1|BI|0000000000000003|EQUAL  |NOT-SENT        |
|IPV6.APP_PREFIX| 64| 1|BI|                |IGNORE |VALUE-SENT      |
|IPV6.APP_IID   | 64| 1|BI|                |IGNORE |VALUE-SENT      |
\---------------+---+--+--+----------------+-------+----------------/
/-------------------------\
|Rule 12/11     00001100  |
!=========================+=========================================\
!^ Fragmentation mode : NoAck   header dtag 2 Window  0 FCN  3  UP ^!
!^ No Tile size specified                                          ^!
!^ RCS Algorithm: RCS_CRC32                                        ^!
\===================================================================/
/-------------------------\
|Rule 100/8     01100100  |
| NO COMPRESSION RULE     |
\-------------------------/
```

Figure 7: Rules example

```xml
&lt;?xml version='1.0' encoding='UTF-8'?&gt;
  &lt;schc xmlns="urn:ietf:params:xml:ns:yang:ietf-schc"&gt;
  &lt;rule&gt;
    &lt;rule-id-value&gt;6&lt;/rule-id-value&gt;
    &lt;rule-id-length&gt;3&lt;/rule-id-length&gt;
    &lt;rule-nature&gt;nature-compression&lt;/rule-nature&gt;
    &lt;entry&gt;
      &lt;field-id&gt;fid-ipv6-version&lt;/field-id&gt;
      &lt;field-length&gt;4&lt;/field-length&gt;
      &lt;field-position&gt;1&lt;/field-position&gt;
      &lt;direction-indicator&gt;di-bidirectional&lt;/direction-indicator&gt;
      &lt;matching-operator&gt;mo-equal&lt;/matching-operator&gt;
      &lt;comp-decomp-action&gt;cda-not-sent&lt;/comp-decomp-action&gt;
      &lt;target-value&gt;
        &lt;index&gt;0&lt;/index&gt;
        &lt;value&gt;AAY=&lt;/value&gt;
      &lt;/target-value&gt;
    &lt;/entry&gt;
    &lt;entry&gt;
      &lt;field-id&gt;fid-ipv6-trafficclass&lt;/field-id&gt;
      &lt;field-length&gt;8&lt;/field-length&gt;
      &lt;field-position&gt;1&lt;/field-position&gt;
      &lt;direction-indicator&gt;di-bidirectional&lt;/direction-indicator&gt;
      &lt;matching-operator&gt;mo-equal&lt;/matching-operator&gt;
      &lt;comp-decomp-action&gt;cda-not-sent&lt;/comp-decomp-action&gt;
      &lt;target-value&gt;
        &lt;index&gt;0&lt;/index&gt;
        &lt;value&gt;AA==&lt;/value&gt;
      &lt;/target-value&gt;
    &lt;/entry&gt;
    &lt;entry&gt;
      &lt;field-id&gt;fid-ipv6-flowlabel&lt;/field-id&gt;
      &lt;field-length&gt;20&lt;/field-length&gt;
      &lt;field-position&gt;1&lt;/field-position&gt;
      &lt;direction-indicator&gt;di-bidirectional&lt;/direction-indicator&gt;
      &lt;matching-operator&gt;mo-ignore&lt;/matching-operator&gt;
      &lt;comp-decomp-action&gt;cda-not-sent&lt;/comp-decomp-action&gt;
      &lt;target-value&gt;
        &lt;index&gt;0&lt;/index&gt;
        &lt;value&gt;AA==&lt;/value&gt;
      &lt;/target-value&gt;
    &lt;/entry&gt;
    &lt;entry&gt;
      &lt;field-id&gt;fid-ipv6-payload-length&lt;/field-id&gt;
      &lt;field-length&gt;16&lt;/field-length&gt;
      &lt;field-position&gt;1&lt;/field-position&gt;
      &lt;direction-indicator&gt;di-bidirectional&lt;/direction-indicator&gt;
      &lt;matching-operator&gt;mo-ignore&lt;/matching-operator&gt;
      &lt;comp-decomp-action&gt;cda-compute&lt;/comp-decomp-action&gt;
```

```
&lt;/entry&gt;
&lt;entry&gt;
  &lt;field-id&gt;fid-ipv6-nextheader&lt;/field-id&gt;
  &lt;field-length&gt;8&lt;/field-length&gt;
  &lt;field-position&gt;1&lt;/field-position&gt;
  &lt;direction-indicator&gt;di-bidirectional&lt;/direction-indicator&gt;
  &lt;matching-operator&gt;mo-equal&lt;/matching-operator&gt;
  &lt;comp-decomp-action&gt;cda-not-sent&lt;/comp-decomp-action&gt;
  &lt;target-value&gt;
    &lt;index&gt;0&lt;/index&gt;
    &lt;value&gt;ADo=&lt;/value&gt;
  &lt;/target-value&gt;
&lt;/entry&gt;
&lt;entry&gt;
  &lt;field-id&gt;fid-ipv6-hoplimit&lt;/field-id&gt;
  &lt;field-length&gt;8&lt;/field-length&gt;
  &lt;field-position&gt;1&lt;/field-position&gt;
  &lt;direction-indicator&gt;di-bidirectional&lt;/direction-indicator&gt;
  &lt;matching-operator&gt;mo-ignore&lt;/matching-operator&gt;
  &lt;comp-decomp-action&gt;cda-not-sent&lt;/comp-decomp-action&gt;
  &lt;target-value&gt;
    &lt;index&gt;0&lt;/index&gt;
    &lt;value&gt;AP8=&lt;/value&gt;
  &lt;/target-value&gt;
&lt;/entry&gt;
&lt;entry&gt;
  &lt;field-id&gt;fid-ipv6-devprefix&lt;/field-id&gt;
  &lt;field-length&gt;64&lt;/field-length&gt;
  &lt;field-position&gt;1&lt;/field-position&gt;
  &lt;direction-indicator&gt;di-bidirectional&lt;/direction-indicator&gt;
  &lt;matching-operator&gt;mo-equal&lt;/matching-operator&gt;
  &lt;comp-decomp-action&gt;cda-not-sent&lt;/comp-decomp-action&gt;
  &lt;target-value&gt;
    &lt;index&gt;0&lt;/index&gt;
    &lt;value&gt;IAEEcB8hAdI=&lt;/value&gt;
  &lt;/target-value&gt;
&lt;/entry&gt;
&lt;entry&gt;
  &lt;field-id&gt;fid-ipv6-deviid&lt;/field-id&gt;
  &lt;field-length&gt;64&lt;/field-length&gt;
  &lt;field-position&gt;1&lt;/field-position&gt;
  &lt;direction-indicator&gt;di-bidirectional&lt;/direction-indicator&gt;
  &lt;matching-operator&gt;mo-equal&lt;/matching-operator&gt;
  &lt;comp-decomp-action&gt;cda-not-sent&lt;/comp-decomp-action&gt;
  &lt;target-value&gt;
    &lt;index&gt;0&lt;/index&gt;
    &lt;value&gt;AAAAAAAAAAM=&lt;/value&gt;
  &lt;/target-value&gt;
&lt;/entry&gt;
```

```
      <entry>
        <field-id>fid-ipv6-appprefix</field-id>
        <field-length>64</field-length>
        <field-position>1</field-position>
        <direction-indicator>di-bidirectional</direction-indicator>
        <matching-operator>mo-ignore</matching-operator>
        <comp-decomp-action>cda-value-sent</comp-decomp-action>
      </entry>
      <entry>
        <field-id>fid-ipv6-appiid</field-id>
        <field-length>64</field-length>
        <field-position>1</field-position>
        <direction-indicator>di-bidirectional</direction-indicator>
        <matching-operator>mo-ignore</matching-operator>
        <comp-decomp-action>cda-value-sent</comp-decomp-action>
      </entry>
    </rule>
    <rule>
      <rule-id-value>12</rule-id-value>
      <rule-id-length>11</rule-id-length>
      <rule-nature>nature-fragmentation</rule-nature>
      <direction>di-up</direction>
      <rcs-algorithm>rcs-crc32</rcs-algorithm>
      <dtag-size>2</dtag-size>
      <fcn-size>3</fcn-size>
      <fragmentation-mode>fragmentation-mode-no-ack</fragmentation-mode>
    </rule>
    <rule>
      <rule-id-value>100</rule-id-value>
      <rule-id-length>8</rule-id-length>
      <rule-nature>nature-no-compression</rule-nature>
    </rule>
  </schc>
```

Figure 8: XML representation of the rules.

11. Acknowledgements

The authors would like to thank Dominique Barthel, Carsten Bormann, Ivan Martinez, Alexander Pelov for their careful reading and valuable inputs. A special thanks for Joe Clarke, Carl Moberg, Tom Petch, Martin Thomson, and Eric Vyncke for their explanations and wise advices when building the model.

12. References

12.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[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>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC7136] Carpenter, B. and S. Jiang, "Significance of IPv6 Interface Identifiers", RFC 7136, DOI 10.17487/RFC7136, February 2014, <https://www.rfc-editor.org/info/rfc7136>.

[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>.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/info/rfc8613>.

[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>.

[RFC8824] Minaburo, A., Toutain, L., and R. Andreasen, "Static Context Header Compression (SCHC) for the Constrained Application Protocol (CoAP)", RFC 8824, DOI 10.17487/RFC8824, June 2021, <https://www.rfc-editor.org/info/rfc8824>.

12.2. Informative References

[I-D.ietf-lpwan-architecture] Pelov, A., Thubert, P., and A. Minaburo, "LPWAN Static Context Header Compression (SCHC) Architecture", Work in Progress, Internet-Draft, draft-ietf-lpwan-architecture-02, 30 June 2022, <https://www.ietf.org/archive/id/draft-ietf-lpwan-architecture-02.txt>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC7967] Bhattacharyya, A., Bandyopadhyay, S., Pal, A., and T. Bose, "Constrained Application Protocol (CoAP) Option for No Server Response", RFC 7967, DOI 10.17487/RFC7967, August 2016, <https://www.rfc-editor.org/info/rfc7967>.

[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>.

[RFC9011] Gimenez, O., Ed. and I. Petrov, Ed., "Static Context Header Compression and Fragmentation (SCHC) over LoRaWAN", RFC 9011, DOI 10.17487/RFC9011, April 2021, <https://www.rfc-editor.org/info/rfc9011>.

Authors' Addresses

Ana Minaburo
Acklio
1137A avenue des Champs Blancs
35510 Cesson-Sevigne Cedex
France

Email: ana@ackl.io

Laurent Toutain
Institut MINES TELECOM; IMT Atlantique
2 rue de la Chataigneraie
CS 17607
35576 Cesson-Sevigne Cedex
France

Email: Laurent.Toutain@imt-atlantique.fr