draft-ietf-lpwan-schc-yang-data-model

Version 20210817

```
revision 2021-08-17 {
  description
    "Initial version from RFC XXXX ";
  reference
    "RFC XXX: Data Model for Static Context Header Compression (SCHC)";
}
```


Authors:

Laurent Toutain (laurent.toutain@imt-atlantique.fr)

Ana Minaburo (ana@ackl.io)

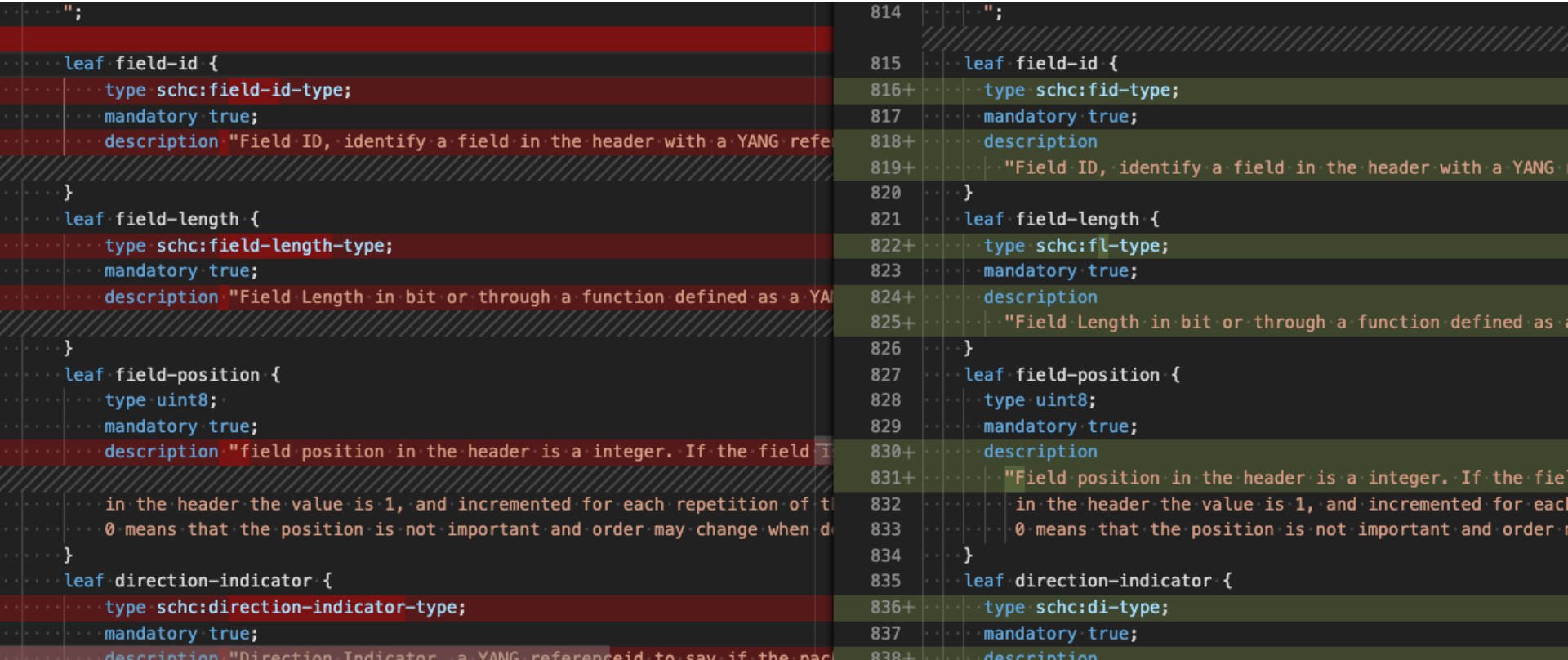
Cosmetic

Add a « - »

```
    }
    leaf dtag-size {
      type uint8;
      default "0";
      description
        "Size in bit of the DTag field";
    }
    leaf w-size {
      when "not(derived-from(../fragmentation-mode, 'fragmentation-mode-no-ack'))";
      type uint8;
      description
        "Size in bit of the window field";
    }
    leaf fcn-size {
      type uint8;
      mandatory true;
      description
        "Size in bit of the FCN field";
    }
```

Cosmetic

Rename types be use RFC8724 terminology and shorter IDs



```
";                                                                  ";

leaf field-id {                                                     leaf field-id {
    type schc:field-id-type;                                            type schc:fid-type;
    mandatory true;                                                     mandatory true;
    description "Field ID, identify a field in the header with a YANG       description
                                                                            "Field ID, identify a field in the header with a YANG
}                                                                   }
leaf field-length {                                                 leaf field-length {
    type schc:field-length-type;                                        type schc:fl-type;
    mandatory true;                                                     mandatory true;
    description "Field Length in bit or through a function defined as a YA      description
                                                                            "Field Length in bit or through a function defined as
}                                                                   }
leaf field-position {                                               leaf field-position {
    type uint8;                                                         type uint8;
    mandatory true;                                                     mandatory true;
    description "field position in the header is a integer. If the field i      description
                                                                            "Field position in the header is a integer. If the fie
    in the header the value is 1, and incremented for each repetition of t          in the header the value is 1, and incremented for eac
    0 means that the position is not important and order may change when d           0 means that the position is not important and order
}                                                                   }
leaf direction-indicator {                                          leaf direction-indicator {
    type schc:direction-indicator-type;                                 type schc:di-type;
    mandatory true;                                                     mandatory true;
    description "Direction Indicator, a YANG reference id to say if the pac     description
```

Compression

- Simplify TV (remove union)

```
776
777   // --------  RULE ENTRY DEFINITION  ------------
778
779+  grouping tv-struct {
780+    description
781+      "Define the target value element. Always a binary type, strings
782+      must be converted to binary. field-id allows the conversion to the appropriate
783+      type.";

784     leaf value {

785       type binary;




786     }
```

Compression

- Rename target-value

```
842+    list target-value {
843+      key "position";
844+      uses tv-struct;
845+      description
846+        "A list of value to compare with the header field value. If target value
847         is a singleton, position must be 0. For matching-list, should be consecutive po
848         values starting from 1.";
849     }
```

  - « rule » and « entry » are singular

No-Compression

- Add a new type of rule for no-compression
- Remove version field (not in RFC8724)

```
1036    container schc {
1037      list rule {
1038        key "rule-id-value rule-id-length";
1039        uses rule-id-type;
1040        choice nature {
1041          case fragmentation {
1042            if-feature "fragmentation";
1043            uses fragmentation-content;
1044          }
1045          case compression {
1046            uses compression-content;
1047          }
1048+         case no-compression {
1049+           description
1050+             "RFC8724 allows a rule for uncompressed headers";
1051          }
```

Fragmentation

- Add l2-word-size

```
899     grouping fragmentation-content {
900+      description
901+        "This grouping defines the fragmentation parameters for
902         all the modes (No Ack, Ack Always and Ack on Error) specified in
903         RFC 8724.";
904+      leaf l2-word-size {
905+        type uint8;
906+        default "8";
907+        description
908+          "Size in bit of the layer 2 word";
909+      }
```

Added Compound Ack

```
identity bitmap-format-base-type {
    description
      "Define how the bitmap is defined in ACK messages.";
}

identity bitmap-RFC8724 {
    base bitmap-format-base-type;
    description
      "Bitmap as defined in RFC8724.";
}

identity bitmap-compound-ack {
    base bitmap-format-base-type;
    description
      "Compound Ack.";
}

typedef bitmap-format-type {
    type identityref {
      base bitmap-format-base-type;
    }
    description
      "type used in rules";
}
```

- See draft

Added Compound Ack

```
1002+        leaf bitmap-format {
1003+          type schc:bitmap-format-type;
1004+          when "derived-from(../fragmentation-mode, 'fragmentation-mode-ack-on-error')";
1005+          default "schc:bitmap-RFC8724";
1006+          description
1007+            "How the bitmaps are included in the Ack message.";
1008         }
```

Is it usefull ?

- Has disappear from RFC 8724

```
812        leaf maximum-window-size {
813          type uint16;
814          description "by default 2^wsize - 1";
815        }
```

relations between values

- add MUST statement in compression rules

```
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 require 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";
}
```

  - `derived-from-or-self` to match identityref

relations between values

- add WHEN statement in fragmentation rules

```
982      case ack-on-error {
983        leaf tile-size {
984          type uint8;
985+         when "derived-from(../fragmentation-mode, 'fragmentation-mode-ack-on-error')";
986+         description
987+           "Size in bit of tiles, if not specified or set to 0: tile fills the fragment
988        }
989        leaf tile-in-All1 {
990          type schc:all1-data-type;
991+         when "derived-from(../fragmentation-mode, 'fragmentation-mode-ack-on-error')";
992+         description
993+           "When true, sender and receiver except a tile in All-1 frag";
994        }
995        leaf ack-behavior {
996          type schc:ack-behavior-type;
997+         when "derived-from(../fragmentation-mode, 'fragmentation-mode-ack-on-error')";
998+         description
999+           "Sender behavior to acknowledge, after All-0, All-1 or when the
1000            LPWAN allows it (Always)";
1001       }
1002+      leaf bitmap-format {
1003+        type schc:bitmap-format-type;
1004+        when "derived-from(../fragmentation-mode, 'fragmentation-mode-ack-on-error')";
1005+        default "schc:bitmap-RFC8724";
1006+        description
```

Other relations between values ?

- Test if MSB arg is shorter than field-length ?
  - How to deal with length functions ?
- Test is LSB/Map-send CDA with MSB/M-M MO ?
  - Not in the spec
- Any other ???

Conclusion

- New version is on github lp-wan repository
  - Check against RFC 8724 for fragmentation
- Used yangson to check rules
  - Done during Hackathon
    - Transform a openSCHC JSON file to a JSON following YANG DM
      - For Compression, Fragmentation To Be Done
      - CORECONF TBD
  - Openschc version will be released soon.

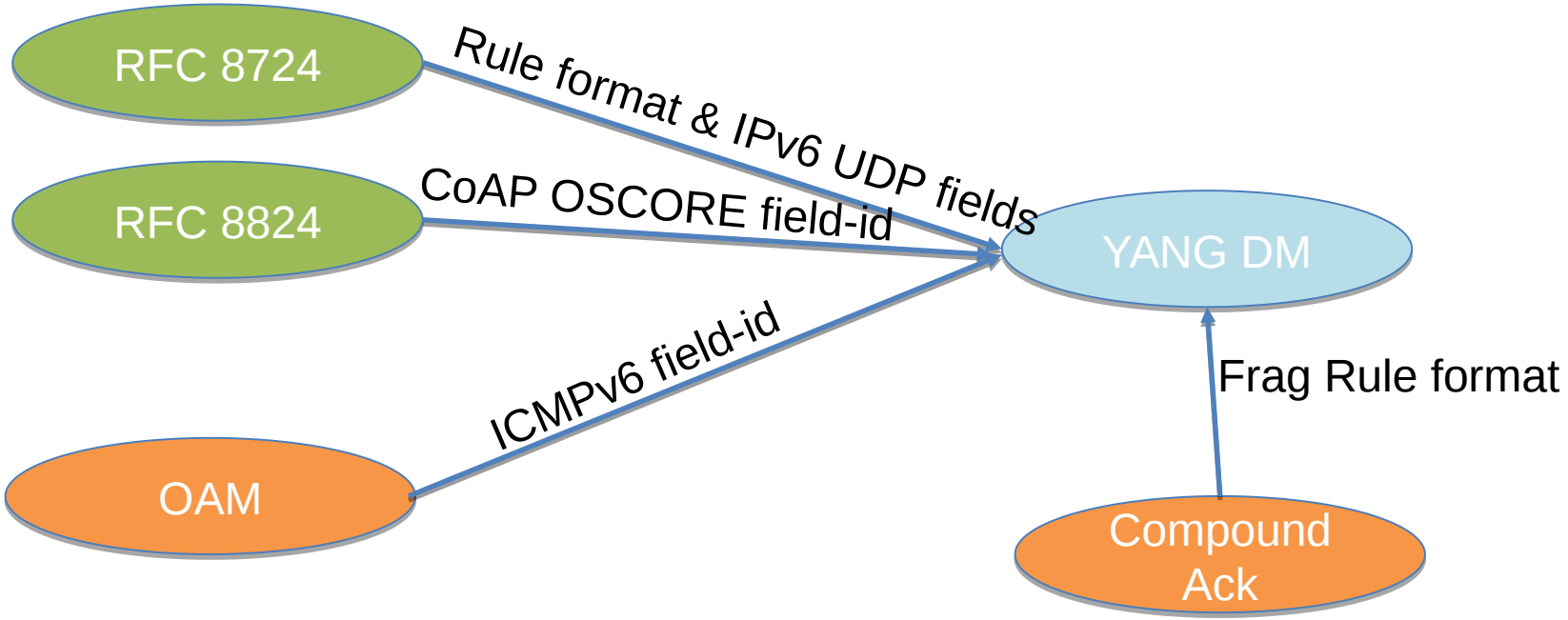
Example openSCHC

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	0000000000000001	EQUAL	NOT-SENT	
IPV6.APP_PREFIX	64	1	BI		IGNORE	VALUE-SENT	
IPV6.APP_IID	64	1	BI		IGNORE	VALUE-SENT	
ICMPV6.TYPE	8	1	DW	128	EQUAL	NOT-SENT	
ICMPV6.TYPE	8	1	UP	129	EQUAL	NOT-SENT	
ICMPV6.CODE	8	1	BI	0	MATCH-MAPPING	MAPPING-SENT	
.	.	.	.	1: .	.	.	
ICMPV6.CKSUM	16	1	BI	0	IGNORE	COMPUTE-CHECKSUM	
ICMPV6.IDENT	16	1	BI	0	MSB(8)	LSB	
ICMPV6.SEQNO	var	1	BI	0	IGNORE	VALUE-SENT	

Example openSCHC

```
{'ietf-schc:schc': {'rule': [{'entry': [{'comp-decomp-action': 'cda-not-sent',
                                         'direction-indicator': 'di-bidirectional',
                                         'field-id': 'fid-ipv6-version',
                                         'field-length': '4',
                                         'field-position': 1,
                                         'matching-operator': 'mo-equal',
                                         'target-value': [{'position': 0,
                                                           'value': b'AAY='}]},
                                        {'comp-decomp-action': 'cda-not-sent',
                                         'direction-indicator': 'di-bidirectional',
                                         'field-id': 'fid-ipv6-trafficclass',
                                         'field-length': '8',
                                         'field-position': 1,
                                         'matching-operator': 'mo-equal',
                                         'target-value': [{'position': 0,
                                                           'value': b'AA=='}]},
```

Dependancies



Tree

```
module: ietf-schc
  +--rw schc
     +--rw version?   uint64
     +--rw rule* [rule-id-value rule-id-length]
        +--rw rule-id-value                 uint32
        +--rw rule-id-length                uint8
        +--rw (nature)?
           +--:(fragmentation) {fragmentation}?
           |  +--rw l2-word-size?           uint8
           |  +--rw direction               schc:direction-indicator-type
           |  +--rw dtag-size?              uint8
           |  +--rw w-size?                 uint8
           |  +--rw fcn-size                uint8
           |  +--rw RCS-algorithm?          RCS-algorithm-type
           |  +--rw maximum-window-size?    uint16
           |  +--rw retransmission-timer?   uint64
           |  +--rw inactivity-timer?       uint64
           |  +--rw max-ack-requests?       uint8
           |  +--rw maximum-packet-size?    uint16
           |  +--rw fragmentation-mode      schc:fragmentation-mode-type
           |  +--rw (mode)?
           |     +--:(no-ack)
           |     +--:(ack-always)
           |     +--:(ack-on-error)
           |        +--rw tile-size?        uint8
           |        +--rw tile-in-All1?     schc:all1-data-type
           |        +--rw ack-behavior?     schc:ack-behavior-type
           |        +--rw bitmap-format?    schc:bitmap-format-type
           +--:(compression)
           |  +--rw entry* [field-id field-position direction-indicator]
           |     +--rw field-id                    schc:field-id-type
           |     +--rw field-length                schc:field-length-type
           |     +--rw field-position              uint8
           |     +--rw direction-indicator         schc:direction-indicator-type
           |     +--rw target-value* [position]
           |     |  +--rw value?      binary
           |     |  +--rw position    uint16
           |     +--rw matching-operator           schc:matching-operator-type
           |     +--rw matching-operator-value* [position]
           |     |  +--rw value?      binary
           |     |  +--rw position    uint16
           |     +--rw comp-decomp-action          schc:comp-decomp-action-type
           |     +--rw comp-decomp-action-value* [position]
           |        +--rw value?      binary
           |        +--rw position    uint16
           +--:(no-compression)
```