SCHC Working Group
Internet-Draft
Intended status: Standards Track
Expires: 25 November 2025

A. Minaburo
Consultant
L. Toutain
C. Banier
IMT Atlantique
M. Dumay
Orange

24 May 2025

CORECONF Rule management for SCHC
draft-toutain-schc-coreconf-management-00

Abstract

This document describes how CORECONF management can be applied to SCHC Context.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 25 November 2025.

Copyright Notice

Table of Contents

1. Introduction{#intro} 2
2. Applicability statement 3
2.1. Architecture 3
2.2. CoAP Profile 4
2.3. Rule modification 4
2.4. Rule creation 5
2.5. Rule deletion 5
3. Management messages 5
3.1. YANG Data Model 5
3.1.1. Nature Management 5
3.1.2. Guard 6
3.1.3. YANG tree representation 6
3.2. Set of Rules Editing 7
3.3. Errors in Management 7
3.4. Methods 8
3.4.1. FETCH 8
3.4.2. iPATCH 8
3.4.3. Optimization 11
3.5. RPC 12
4. Protocol Stack 13
4.1. Compression Rules 13
5. OSCORE 15
5.1. Compression Rules 15
6. DTLS 15
6.1. Compression Rules 15
7. Example CORECONF usage in Python 15
7.1. Deletion cases 15
7.2. Update cases 19
7.3. Addition cases 20
8. Normative References 23
Appendix A. YANG DM 24
Acknowledgments 25
Authors' Addresses 25

1. Introduction{#intro}

[RFC9363] defines the YANG Data Model for a SCHC context (a.k.a Set of Rules). [I-D.ietf-lpwan-architecture] proposes the architecture for rule management. Some rules must be clearly dedicated to the modification of the context.

[RFC9254] defines a way to serialize data issued from a YANG DM into a CBOR representation and [I-D.ietf-core-comi] defines the CoAP interface.

This document describes in which condition management can be done, how to manage rules inside an SCHC instance using CORECONF and proposes some compression rules for the protocol headers.

2. Applicability statement

2.1. Architecture

SCHC instance management allows the two end-points to modify the common SoR, by:

* modifying rules values (such as TV, MO or CDA) in existing rules,

* adding a new rule,

* removing an existing rule.

The rule management uses the CORECONF interface [I-D.ietf-core-comi] based on CoAP. The management traffic is carried as SCHC compressed packets tagged to some specific rule IDs. They are identified as M rules in Figure Figure 1. Management Rules (or M rules) can be either Compression rules or No compression rules. Only M rules can modify the SoR.

Management procedures uses their own IPv6 stack, independent of the rest of the system.

SCHC Packets using M Rules MUST be encrypted either by the underlying layer (for instance in a QUIC stream dedicated to management inside a QUIC connection) or directly using OSCORE or DTLS.

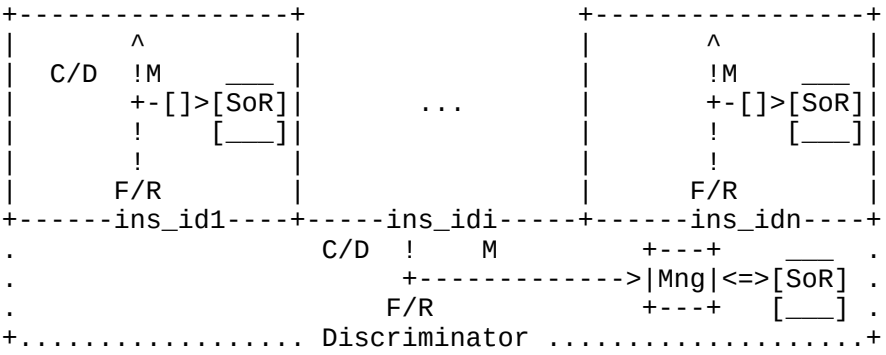

Figure 1: Inband Management

2.2. CoAP Profile

Management requests MUST be protected against packet loss. It is RECOMMENDED to use CONfirmable requests and no Token. If the management request is too large regarding the MTU, SCHC Fragmentation SHOULD be used instead of the Block option. As shown in figure Figure 1 fragmentation can be common to Management rules and other rules.

2.3. Rule modification

SCHC imposes that both ends share exactly the same SoR, therefore, a new or modified rule cannot be used while it remains in candidate status until the other end has validated the modification. A candidate rule cannot be used, either in C/D or F/R. A SCHC PDU MUST NOT be generated with a candidate rule ID and received PDUs containing a candidate rule ID must be dropped.

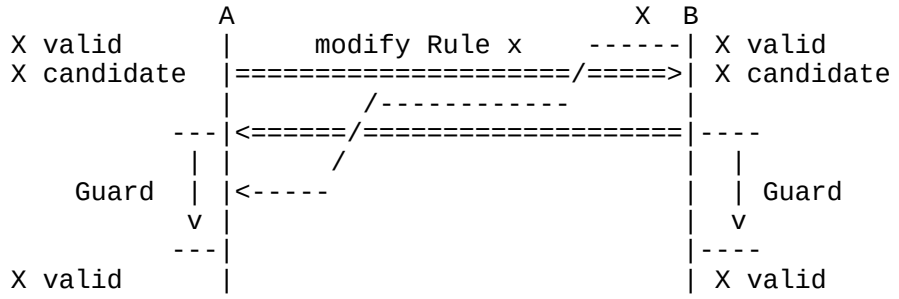

Figure 2: Modifying a rule

Figure 2 illustrates a Rule modification. The left-hand side entity A wants to make rule x evolve. It send an acknowledged CoAP message to the other end. Host A change the status of the rule to "candidate", indicating that the rule can no longer be used for SCHC procedures. The receiving entity B, acknowledges the message and continues to maintain the "candidate" status for a Guard period. At the reception of the acknowledgement, A set also a Guard period before rule x becomes valid again.

The Guard period is used to avoid SCHC message with a rule ID to appear at the other end after a rule modification. The Guard period appears only once during the rule management and is depends on the out-of-sequence messages expected between both ends.

2.4. Rule creation

Rule creation do not require a Guard period, and acknowledgement is RECOMMENDED. Figure Figure 3 gives an example, where the Acknowledgment is lost. Entity A sends a management message to create a new rule. Since its a new rule, the Guard period is not set and the new rule becomes immediately valid on B. The Acknowledgement does not reach A, so the rule stays in the candidate state, but the reception of a SCHC PDU carrying the RulE ID X, informs that the message has been correctly received by B. So X becomes valid in A.

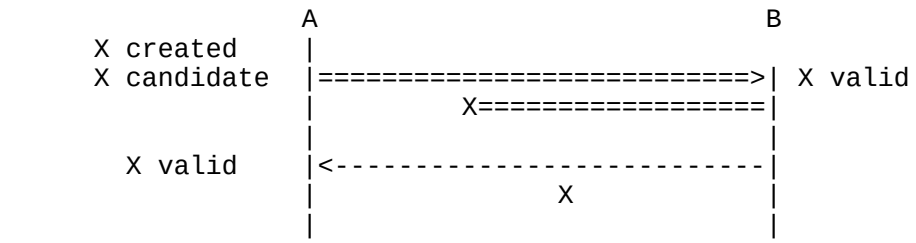

Figure 3: Modifying a rule

2.5. Rule deletion

After the rule deletion, a Guard period is established. During that period, a rule with the same ID cannot be created, and SCHC PDU corrying the Rule ID are dropped.

3. Management messages

3.1. YANG Data Model

CORECONF proposes an interface to manage data structured with a YANG Data Model. RFC 9363 defines a YANG Data Model for SCHC Rules. SCHC Instance Management MUST use FETCH to read a rule and iPATCH to create, modify or delete a rule. In order to accomplish management, the YANG Data Model has been updated.

3.1.1. Nature Management

M Rules have to be marked in a way that allows quickly identifying which rules in a SoR are responsible for management. Therefore, a new "nature-management" type has been defined. This nature is actually a specialization of "nature-compression" for management purposes and compression needs to be available and activated to do management.

3.1.2. Guard

To determine if a rule is considered available or not during the Guard period, a rule needs to have a status which determines if it can be used. Basically, an available rule MUST associate the key "rule-status" with the value "status-active". Conversely, during the Guard period, "rule-status" MUST be set to "status-candidate".

3.1.3. YANG tree representation

The YANG tree represents the Rule structure as defined in RFC 9363 with the two updates described above:

```
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-status                     status-type
        +--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 or management}?
              +--rw entry* [field-id field-position direction-indicator]
```

```
                +--rw field-id                    schc:fid-type
                +--rw field-length                union
                +--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 4: Updated YANG Data Model for CORECONF

3.2. Set of Rules Editing

For clarity reasons, this document will use YANG Identifiers in quotes instead of the SID values. In the YANG tree, all the lines of the tree have a SID number. Each level of the hierarchy is accessible through one or several keys. For example, to access the hierarchy under "rule", "rule-id-value" and "rule-id-length" must be specified. To access the hierarchy describing an entry in a compression rule, "rule-id-value" and "rule-id-length" must be followed by "field-id", "field-position" and "direction-indicator". Since the TV, MO-value and CDA-value are stored as lists, "index" must be added to access a specific element.

Therefore, to access a specific element in a hierarchy, the SID of this element has to be specified, followed by the keys needed to access it.

For example, ["target-value/value", 5, 3, "fid-ipv6-version", 1, "di-bidirectional", 0] is used to access the first value (0) of TV for the IPv6 Version of Rule 5/3.

3.3. Errors in Management

There is different level of error detection:

* CORECONF Errors: these errors are directly generated at the CORECONF level. For instance, retrieving a value with a wrong key.

* YANG validation errors: the data model is not conforming with the constraints such as "must" or "mandatory". This check is optional, since it may require a lot of resources on a device.

* SCHC errors: Errors on the Data Model that cannot be detected at the YANG level, for example, the rule numbering does not respect a binary tree.

3.4. Methods

3.4.1. FETCH

As mentioned in [I-D.ietf-core-comi], FETCH request helps to retrieve at least one instance-value.

Example : Fetching TV, MO and CDA of the Entry fid-ipv6-version/1/ bidirectional from Rule 6/3.

```
REQ: FETCH /c
     (Content-Format: application/yang-identifiers+cbor-seq)
["target-value",       6, 3, "fid-ipv6-version", 1, "di-bidirectional"],
["matching-operator",  6, 3, "fid-ipv6-version", 1, "di-bidirectional"],
["comp-decomp-action", 6, 3, "fid-ipv6-version", 1, "di-bidirectional"]

RES: 2.05 Content
     (Content-Format: application/yang-instances+cbor-seq)
{
  {"target-value"       : [{"index" : 0, "value" : h"06"}]},
  {"matching-operator"  : "mo-equal"},
  {"comp-decomp-action" : "cda-not-sent"}
}
```

3.4.2. iPATCH

To write an iPATCH request, several methods could be used. For instance, in a Rule 7/8, an entry for a field was set to ignore/ value-sent and the target-value was not set, the following command specify a new TV and change the MO and CDA. It is possible to set up individually each field, as given in the following example:

* Specified all conserned fields :

```
  iPATCH /c
  {
    ["target-value", 7, 8, field, 1, "di-bidirectional"] : [{delta_TV: 0, delta_value: va
lue}],
    ["matching-operator", 7, 8, field, 1, "bi-directional"] : "mo-equal",
    ["comp-decomp-action", 7, 8, field, 1, "bi-directional"] : "cda-not-sent"
  }
```

But if the changes concerns the same subtree, it is RECOMMENDED to regroup the changes in a unique fetch, as given in the following example:

~~~ iPATCH /c { ["entry", 7, 8, field, 1, "di-bidirectional"] : { delta_target-value : [{delta_index : 0, delta_value : value}], delta_matching-operator : "mo-equal", delta_comp-decomp-action : "cda-not-sent" } } ~~~

The same principle is applied to rules and "leaf-list".

3.4.2.1. Add

If the target object doesn’t exist in the context, then it is appended. It supports three main cases: * Adding a new key-value pair to an existing object * Adding a new object to an existing list

One important specification is that for every leaf-list, the YANG Data Model describes that every index should be incremental. In CORECONF, we trust the user/system.

Example: - Add TV into fid-ipv6-payload-length/1/di-bidirectional in Rule 0/3

```
REQ: iPATCH /c
    (Content-Format: application/yang-identifiers+cbor-seq)
{
  ["target-value", 0, 3, "fid-ipv6-payload-length", 1, "di-bidirectional"] : [
      {delta_index : 0, delta_value : h"50"},
      {delta_index : 1, delta_value : h"55"}
  ]
}

RES: 2.04 Changed
```

3.4.2.2. Update

A request can be considered as an update if the target associated with the various keys is present in the context. Otherwise, it could be consider as an add or an error.

Example : - The Entry fid-ipv6-version/1/di-bidirectional is in Rule 6/3.
~~~

```
REQ: iPATCH /c
     (Content-Format: application/yang-identifiers+cbor-seq)
{
  ["entry", 6, 3, "fid-ipv6-version", 1, "di-bidirectional"] : {
      {"delta_target-value": []},
      {"delta_matching-operator": "mo-ignore"},
      {"delta_comp-decomp-action": "cda-value-sent"}
  }
}

RES: 2.04 Changed
```

* The Entry fid-ipv6-version/1/di-bidirectional is in not in Rule 7/8 but Rule 7/8 exist.

```
REQ: iPATCH /c
     (Content-Format: application/yang-identifiers+cbor-seq)
{
  ["entry", 7, 8, "fid-ipv6-version", 1, "di-bidirectional"] : {
      {"delta_target-value"       : []},
      {"delta_matching-operator"  : "mo-ignore"},
      {"delta_comp-decomp-action" : "cda-value-sent"}
  }
}

RES: 2.04 Changed
```

* The Entry fid-ipv6-version/1/di-bidirectional is not in Rule 5/8, and Rule 5/8 does not exist. Therefore, Rule 5/8 cannot be added in order to include the Entry fid-ipv6-version/1/di-bidirectional because other fields, which are not keys, cannot be deducted at every depth of the context.

```
REQ: iPATCH /c
     (Content-Format: application/yang-identifiers+cbor-seq)
{
  ["entry", 5, 8, "fid-ipv6-version", 1, "di-bidirectional"] : {
      {"delta_target-value"       : []},
      {"delta_matching-operator"  : "mo-ignore"},
      {"delta_comp-decomp-action" : "cda-value-sent"}
  }
}

RES: 4.04 Not Found
```

3.4.2.3. Delete

To remove an object we use "null" value.

Example: - Delete Rule 7/8

```
  REQ: iPATCH /c
       (Content-Format: application/yang-identifiers+cbor-seq)
  {
    ["rule", 7, 8]: null
  }

  RES: 2.04 Changed
```

For deletion, we limit the actions and consider a minimal CORECONF representation as {"ietf-schc:schc" : {"rule" : []}}. Therefore, a request trying to delete "ietf-schc:schc" will set the CORECONF representation to the minimal one. Additionally, while updates are authorized, deleting a protected key is forbidden.

Example: - Delete rule-id-value of Rule 0/3

```
  REQ: iPATCH /c
       (Content-Format: application/yang-identifiers+cbor-seq)
  {
    ["rule-id-value", 0, 3]: null
  }

  RES: 4.00 Bad Request
```

3.4.3. Optimization

This process imposes to send the full rule in the value part, so an optimization can be done by deriving an existing rule and modify some parameters.

[I-D.toutain-schc-universal-option] augments the data model for universal options. This add to compression rules a new entry format where a field is indexed with: * a space-id, a YANG identifier referring to the protocol containing options (CoAP, QUIC, TCP,...) * the option used in the protocol * the position

```
     +--rw schc-opt:entry-option-space* \
         [space-id option-value field-position direction-indicator]
        +--rw schc-opt:space-id                    space-type
        +--rw schc-opt:option-value                uint32
        +--rw schc-opt:field-length                union
        +--rw schc-opt:field-position              uint8
        +--rw schc-opt:direction-indicator         schc:di-type
        +--rw schc-opt:target-value* [index]
        |  +--rw schc-opt:index    uint16
        |  +--rw schc-opt:value?   binary
        +--rw schc-opt:matching-operator           schc:mo-type
        +--rw schc-opt:matching-operator-value* [index]
        |  +--rw schc-opt:index    uint16
        |  +--rw schc-opt:value?   binary
        +--rw schc-opt:comp-decomp-action          schc:cda-type
        +--rw schc-opt:comp-decomp-action-value* [index]
           +--rw schc-opt:index    uint16
           +--rw schc-opt:value?   binary
```

In the CORECONF representation, even if the name are similar in the structure, the SID values are different. The key for an entry contains 4 elements.

```
REQ: FETCH </c>
        (Content-Format: application/yang-identifiers+cbor-seq)
   ["schc-opt:matching-operator", 8, 3, "schc-opt:space-id-coap", 11, 1, "di-up"]
```

3.5. RPC

Represented as a tree:

```
     rpcs:
       +---x duplicate-rule
          +---w input
          |  +---w from
          |  |  +---w rule-id-value?    uint32
          |  |  +---w rule-id-length?   uint8
          |  +---w to
          |  |  +---w rule-id-value?    uint32
          |  |  +---w rule-id-length?   uint8
          |  +---w ipatch-sequence?   binary
          +--ro output
             +--ro status?   string
```

After duplication, the new rule stays in a candidate state until the new values are set.

4. Protocol Stack

The management inside the instance has its own IPv6 stack totally independent of the rest of the system. The goal is to implement IPv6/UDP/CoAP to allow the implementation of the CORECONF interface. No other kind of traffic is allowed.

The end-point acting as a Device has the IPv6 address fe80::1/64 and the other end fe80::2/64.

Both implements CoAP client and server capabilities. The server uses port 5683 and the client 3865.

4.1. Compression Rules

Two rules are required for management functionality. The first rule (RuleID M1) defines packets containing application payloads that include a CoAP Content-Format field. Depending on the direction (Up or Down), this rule manages Confirmable FETCH/iPATCH requests or Non-Confirmable Content responses accordingly. Therefore, the second rule (RuleID M2) is used to compress packets which do not include application payload, basically response packets in downlink.

RuleID M1						
FID	FL	FP	DI	TV	MO	CDA
IPv6 Version	4	1	Bi	6	equal	not-sent
IPv6 Traffic Class	8	1	Bi	1	equal	not-sent
IPv6 Flow Label	20	1	Bi	144470	equal	not-sent
IPv6 Length	16	1	Bi		ignore	compute-*
IPv6 Next Header	8	1	Bi	17	equal	not-sent
IPv6 Hop Limit	8	1	Bi	64	equal	not-sent
IPv6 DevPrefix	64	1	Bi	fe80::/64	equal	not-sent
IPv6 DevIID	64	1	Bi	::2	equal	not-sent
IPv6 AppPrefix	64	1	Bi	fe80::/64	equal	not-sent
IPv6 AppIID	64	1	Bi	::1	equal	not-sent
UDP DevPort	16	1	Bi	3865	equal	not-sent
UDP AppPort	16	1	Bi	5683	equal	not-sent
UDP Length	16	1	Bi		ignore	compute-*
UDP Checksum	16	1	Bi		ignore	compute-*
CoAP Version	2	1	Bi	1	equal	not-sent
CoAP Type	2	1	Dw	2	equal	not-sent
CoAP Type	2	1	Up	0	equal	not-sent
CoAP TKL	4	1	Bi	0	equal	not-sent
CoAP Code	8	1	Up	[5, 7]	match-mapping	mapping-sent
CoAP Code	8	1	Dw	69	equal	not-sent
CoAP MID	16	1	Bi	0	MSB(9)	LSB
CoAP Uri-Path	8	1	Bi	c	equal	not-sent
CoAP Content-Format	8	1	Bi	application	equal	not-sent
	8	1	Bi	/yang-ident		
	8	1	Bi	fiers+cbor-		
	8	1	Bi	seq		

Figure 5: Management Rule 1

RuleID M2						
FID	FL	FP	DI	TV	MO	CDA
IPv6 Version	4	1	Bi	6	equal	not-sent
IPv6 Traffic Class	8	1	Bi	1	equal	not-sent
IPv6 Flow Label	20	1	Bi	144470	equal	not-sent
IPv6 Length	16	1	Bi		ignore	compute-*
IPv6 Next Header	8	1	Bi	17	equal	not-sent
IPv6 Hop Limit	8	1	Bi	64	equal	not-sent
IPv6 DevPrefix	64	1	Bi	fe80::/64	equal	not-sent
IPv6 DevIID	64	1	Bi	::2	equal	not-sent
IPv6 AppPrefix	64	1	Bi	fe80::/64	equal	not-sent
IPv6 AppIID	64	1	Bi	::1	equal	not-sent
UDP DevPort	16	1	Bi	3865	equal	not-sent
UDP AppPort	16	1	Bi	5683	equal	not-sent
UDP Length	16	1	Bi		ignore	compute-*
UDP Checksum	16	1	Bi		ignore	compute-*
CoAP Version	2	1	Bi	1	equal	not-sent
CoAP Type	2	1	Dw	2	equal	not-sent
CoAP TKL	4	1	Bi	0	equal	not-sent
CoAP Code	8	1	Dw	[68, 128, 132]	match-mapping	mapping-sent
CoAP MID	16	1	Bi	0	MSB(9)	LSB

Figure 6: Management Rule 2

5. OSCORE

5.1. Compression Rules

6. DTLS

6.1. Compression Rules

7. Example CORECONF usage in Python

7.1. Deletion cases

* Delete root element:

```
YANG REQ: iPATCH /c
{
  (’/ietf-schc:schc’): None
}

CORECONF REQ: iPATCH /c
{
  (5100): None
}

REQ: iPATCH /c
    (Content-Format: application/yang-identifiers+cbor-seq)
a1811913ecf6

RES: 2.04 Changed
```

* Delete a specific rule:

```
YANG REQ: iPATCH /c
{
  (’/ietf-schc:schc/rule’, 0, 3): None
}

CORECONF REQ: iPATCH /c
{
  (5101, 0, 3): None
}

REQ: iPATCH /c
    (Content-Format: application/yang-identifiers+cbor-seq)
a1831913ed0003f6

RES: 2.04 Changed
```

* Delete a specific entry:

```
YANG REQ: iPATCH /c
{
  (’/ietf-schc:schc/rule/entry’, 0, 3, ’fid-ipv6-version’, 1, ’di-bidirectional’): None
}

CORECONF REQ: iPATCH /c
{
  (5105, 0, 3, 5068, 1, 5018): None
}

REQ: iPATCH /c
    (Content-Format: application/yang-identifiers+cbor-seq)
a1861913f100031913cc0119139af6

RES: 2.04 Changed
```

* Delete a basic key:

```
YANG REQ: iPATCH /c
{
  (’/ietf-schc:schc/rule/rule-status’, 0, 3): None
}

CORECONF REQ: iPATCH /c
{
  (5137, 0, 3): None
}

REQ: iPATCH /c
    (Content-Format: application/yang-identifiers+cbor-seq)
a1831914110003f6

RES: 2.04 Changed
```

* Delete a leaf-list single:

```
YANG REQ: iPATCH /c
{
  (’/ietf-schc:schc/rule/entry/target-value/value’, 0, 3, ’fid-ipv6-version’, 1, ’di-bidi
rectional’, 0): None
}

CORECONF REQ: iPATCH /c
{
  (5120, 0, 3, 5068, 1, 5018, 0): None
}

REQ: iPATCH /c
    (Content-Format: application/yang-identifiers+cbor-seq)
a187191400000319 13cc0119139a00f6

RES: 2.04 Changed
```

* Delete a leaf-list several:

```
YANG REQ: iPATCH /c
{
  (’/ietf-schc:schc/rule/entry/target-value/value’, 0, 3, ’fid-ipv6-trafficclass’, 1, ’di
-bidirectional’, 1): None
}

CORECONF REQ: iPATCH /c
{
  (5120, 0, 3, 5065, 1, 5018, 1): None
}

REQ: iPATCH /c
    (Content-Format: application/yang-identifiers+cbor-seq)
a187191400000319 13c90119139a01f6

RES: 2.04 Changed
```

* Delete an unknown entry:

```
YANG REQ: iPATCH /c
{
  (’/ietf-schc:schc/rule/entry’, 2, 3, ’fid-ipv6-version’, 1, ’di-bidirectional’): None
}

CORECONF REQ: iPATCH /c
{
  (5105, 2, 3, 5068, 1, 5018): None
}

REQ: iPATCH /c
    (Content-Format: application/yang-identifiers+cbor-seq)
a1861913f102031913cc0119139af6

RES: 4.00 Bad Request
```

* Delete a protected key:

```
YANG REQ: iPATCH /c
{
  (’/ietf-schc:schc/rule/rule-id-value’, 0, 3): None
}

CORECONF REQ: iPATCH /c
{
  (5135, 0, 3): None
}

REQ: iPATCH /c
    (Content-Format: application/yang-identifiers+cbor-seq)
a18319140f0003f6

RES: 4.00 Bad Request
```

7.2. Update cases

* Update protected key:

```
YANG REQ: iPATCH /c
{
  (’/ietf-schc:schc/rule/rule-id-value’, 0, 3): 5
}

CORECONF REQ: iPATCH /c
{
  (5135, 0, 3): 5
}

REQ: iPATCH /c
    (Content-Format: application/yang-identifiers+cbor-seq)
a18319140f000305

RES: 2.04 Changed
```

* Update basic key:

```
YANG REQ: iPATCH /c
{
  (’/ietf-schc:schc/rule/rule-status’, 0, 3): ’status-candidate’
}

CORECONF REQ: iPATCH /c
{
  (5137, 0, 3): 5096
}

REQ: iPATCH /c
    (Content-Format: application/yang-identifiers+cbor-seq)
a18319141100031913e8

RES: 2.04 Changed
```

7.3. Addition cases

* Add a new entry:

```
YANG REQ: iPATCH /c
{
  (’/ietf-schc:schc/rule/entry’, 0, 3, ’fid-ipv6-appprefix’, 1, ’di-bidirectional’): {
      ’field-length’: 64,
      ’target-value’: [{’index’: 0, ’value’: ’/oAAAAAAAAA=’}],
      ’matching-operator’: ’ietf-schc:mo-equal’,
      ’comp-decomp-action’: ’ietf-schc:cda-not-sent’
  }
}

CORECONF REQ: iPATCH /c
{
  (5105, 0, 3, 5057, 1, 5018): {
      7: 64,
      13: [{1: 0, 2: b’\xfe\x80\x00\x00\x00\x00\x00\x00’}],
      9: 5083,
      1: 5015
  }
}

REQ: iPATCH /c
    (Content-Format: application/yang-identifiers+cbor-seq)
a1861913f100031913c10119139aa40718400d81a201000248fe8000000000000091913db01191397

RES: 2.04 Changed

*  Add leaf-list incremental:

YANG REQ: iPATCH /c
{
  (’/ietf-schc:schc/rule/entry/target-value’, 0, 3, ’fid-ipv6-flowlabel’, 1, ’di-bidirect
ional’): {
      ’index’: 4, ’value’: ’vLw=’
  }
}

CORECONF REQ: iPATCH /c
{
  (5118, 0, 3, 5061, 1, 5018): {
      1: 4, 2: b’\xbc\xbc’
  }
}

REQ: iPATCH /c
    (Content-Format: application/yang-identifiers+cbor-seq)
a1861913fe00031913c50119139aa201040242bcbc

RES: 2.04 Changed
```

* Add leaf-list non-incremental:

```
YANG REQ: iPATCH /c
{
  (’/ietf-schc:schc/rule/entry/target-value’, 0, 3, ’fid-ipv6-flowlabel’, 1, ’di-bidirect
ional’): {
      ’index’: 7, ’value’: ’vLw=’
  }
}

CORECONF REQ: iPATCH /c
{
  (5118, 0, 3, 5061, 1, 5018): {
      1: 7, 2: b’\xbc\xbc’
  }
}

REQ: iPATCH /c
    (Content-Format: application/yang-identifiers+cbor-seq)
a1861913fe00031913c50119139aa201070242bcbc

RES: 2.04 Changed
```

* Add new key:value:

```
YANG REQ: iPATCH /c
{
  (’/ietf-schc:schc/rule/entry/target-value’, 0, 3, ’fid-ipv6-payload-length’, 1, ’di-bid
irectional’): [
      {’index’: 0, ’value’: ’UA==’},
      {’index’: 1, ’value’: ’VQ==’}
  ]
}

CORECONF REQ: iPATCH /c
{
  (5118, 0, 3, 5064, 1, 5018): [
      {1: 0, 2: b’\x50’}, {1: 1, 2: b’\x55’}
  ]
}

REQ: iPATCH /c
    (Content-Format: application/yang-identifiers+cbor-seq)
a1861913fe00031913c80119139a82a20100024150a20101024155

RES: 2.04 Changed
```

* Add new rule:

```
   YANG REQ: iPATCH /c
   {
     (’/ietf-schc:schc/rule’, 5, 3): {
         ’rule-status’: ’ietf-schc:status-active’,
         ’rule-id-value’: 10,
         ’rule-id-length’: 5,
         ’rule-nature’: ’ietf-schc:nature-compression’
     }
   }

   CORECONF REQ: iPATCH /c
   {
     (5101, 5, 3): {36: 5094, 34: 10, 33: 5, 35: 5088}
   }

   REQ: iPATCH /c
       (Content-Format: application/yang-identifiers+cbor-seq)
   a1831913ed0503a418241913e618220a18210518231913e0

   RES: 2.04 Changed
```

* Add entry into unknown rule:

```
YANG REQ: iPATCH /c
{
  (’/ietf-schc:schc/rule/entry’, 250, 8, ’fid-ipv6-payload-length’, 1, ’di-bidirectional’
): {
      ’field-length’: 16,
      ’matching-operator’: ’ietf-schc:mo-ignore’,
      ’comp-decomp-action’: ’ietf-schc:cda-value-sent’
  }
}

CORECONF REQ: iPATCH /c
{
  (5105, 250, 8, 5064, 1, 5018): {7: 16, 9: 5084, 1: 5016}
}

REQ: iPATCH /c
    (Content-Format: application/yang-identifiers+cbor-seq)
a1861913f118fa081913c80119139aa30710091913dc01191398

RES: 4.00 Bad Request
```

8. Normative References


```
   [I-D.ietf-core-comi]
              Veillette, M., Van der Stok, P., Pelov, A., Bierman, A.,
              and C. Bormann, "CoAP Management Interface (CORECONF)",
```

Work in Progress, Internet-Draft, draft-ietf-core-comi-20, 6 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-comi-20>.

[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://datatracker.ietf.org/doc/html/draft-ietf-lpwan-architecture-02>.

[I-D.toutain-schc-sid-allocation]
Minaburo, A. and L. Toutain, "SCHC Sid Allocation", Work in Progress, Internet-Draft, draft-toutain-schc-sid-allocation-01, 7 July 2023, <https://datatracker.ietf.org/doc/html/draft-toutain-schc-sid-allocation-01>.

[I-D.toutain-schc-universal-option]
Lampin, Q., Minaburo, A., Tiloca, M., and L. Toutain, "Options representation in SCHC YANG Data Models", Work in Progress, Internet-Draft, draft-toutain-schc-universal-option-01, 14 April 2025, <https://datatracker.ietf.org/doc/html/draft-toutain-schc-universal-option-01>.

[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/rfc/rfc8724>.

[RFC9254] Veillette, M., Ed., Petrov, I., Ed., Pelov, A., Bormann, C., and M. Richardson, "Encoding of Data Modeled with YANG in the Concise Binary Object Representation (CBOR)", RFC 9254, DOI 10.17487/RFC9254, July 2022, <https://www.rfc-editor.org/rfc/rfc9254>.

[RFC9363] Minaburo, A. and L. Toutain, "A YANG Data Model for Static Context Header Compression (SCHC)", RFC 9363, DOI 10.17487/RFC9363, March 2023, <https://www.rfc-editor.org/rfc/rfc9363>.

Appendix A. YANG DM

```
rpc duplicate-rule { input { container from { uses ietf-schc:rule-id-
type; } container to { uses ietf-schc:rule-id-type; } } output { leaf
status { type string; } } }
```

Acknowledgments

The authors sincerely thank

This work was supported by the Sweden’s Innovation Agency VINNOVA within the EUREKA CELTIC-NEXT project CYPRESS.

Authors’ Addresses

Ana Minaburo
Consultant
Rue de Rennes
35510 Cesson-Sevigne
France
Email: anaminaburo@gmail.com

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

Corentin Banier
IMT Atlantique
CS 17607, 2 rue de la Chataigneraie
35576 Cesson-Sevigne Cedex
France
Email: corentin.banier@imt-atlantique.fr

Marion Dumay
Orange
Email: marion.dumay@orange.com