SCHC Working Group
Internet-Draft
Intended status: Standards Track
Expires: 20 April 2026

Q. Lampin
Orange
A. Minaburo
Consultant
M. Tiloca
RISE AB
L. Toutain
IMT Atlantique

17 October 2025

Options representation in SCHC YANG Data Models
draft-ietf-schc-universal-option-01

Abstract

The idea of keeping option identifiers in SCHC Rules simplifies the interoperability and the evolution of SCHC compression, when the protocol introduces new options, that can be unknown from the current SCHC implementation. This document discuss the augmentation of the current YANG Data Model, in order to add in the Rule options identifiers used by the protocol.

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 20 April 2026.

Copyright Notice

Table of Contents

1. Introduction

This document aims to be included in a revision of [RFC9363] to replace the definition of identifiers for CoAP options.

Static Context Header Compression (SCHC) provides an essential mechanism for efficient communication in constrained networks by compressing protocol headers using predefined Rules. Originally developed for Low Power Wide Area Networks (LPWANs), SCHC has proven effective in scenarios with limited bandwidth, power constraints, and predictable traffic patterns.

The YANG Data Model defined in [RFC9363] was designed primarily for LPWAN technologies, focusing on highly constrained devices such as sensors and actuators that generate predictable traffic patterns, which allowed for static compression rules. This model also incorporates CoAP parameters defined in [RFC8824], representing CoAP options as specific Field Identifiers (FIDs) within SCHC Rules.

While this approach works well in controlled LPWAN environments, it presents interoperability challenges when SCHC is applied to more dynamic networks or when protocols evolve to incorporate new options. The primary issue arises from the disconnection between protocol option identifiers and SCHC Field Identifiers (FIDs), making it difficult to efficiently handle newly defined or private options without disrupting existing implementations.

This document proposes a more flexible approach to representing protocol options in SCHC YANG Data Models. By preserving original protocol option identifiers within SCHC Rules, we aim to:

* Improve interoperability between SCHC implementations

* Enable more graceful handling of protocol evolution

* Simplify Rule management between SCHC endpoints

* Support compression of newly defined or private options without requiring updates to the SCHC implementation

The following sections examine the current challenges in detail, explore potential solutions including both semantic and syntactic compression approaches, and propose extensions to the YANG Data Model that preserve protocol option identifiers while maintaining efficient compression capabilities.

2. Current Challenges

The fundamental challenge in SCHC compression for protocols with options stems from how options are represented in the SCHC Data Model. Unlike the relatively static headers of IPv6 or UDP, CoAP includes a flexible options mechanism that allows for protocol extension through new options.

2.1. Option Representation Problem

In the current SCHC Data Model defined in [RFC9363], CoAP options are represented as predefined Field Identifiers (FIDs) that are included in SCHC Rules. Each option is essentially abstracted away from its original protocol identifier and assigned a new SCHC-specific identifier. While this approach works adequately in static LPWAN environments where the set of options is known in advance and rarely changes, it creates significant challenges in more dynamic networks or when protocols evolve.

The core problem is that the mapping between protocol option identifiers (as defined in the protocol specification) and SCHC FIDs is not standardized or predictable. When a new CoAP option is defined after a SCHC implementation is deployed, there is no straightforward mechanism for the implementation to assign an appropriate FID to this option or to communicate this assignment to other SCHC implementations.

2.2. Rule Management Challenges

This limitation becomes particularly problematic in scenarios involving rule management between two SCHC endpoints. The following scenario (cf. Figure 1) illustrates this issue:

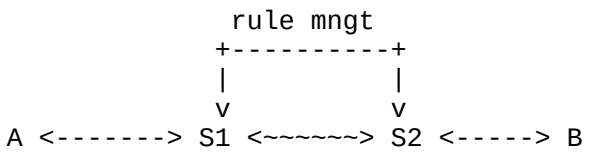


Figure 1: Rule Management between two SCHC end-points

In this scenario:

* Device A generates CoAP packets that may include various options, including newly defined ones.

* SCHC nodes S1 and S2 compress and decompress the traffic using Rules they share.

* When A uses a newly defined or private option, S1 can parse this option (as the CoAP header structure remains consistent) and could potentially derive a new Rule to optimize compression.

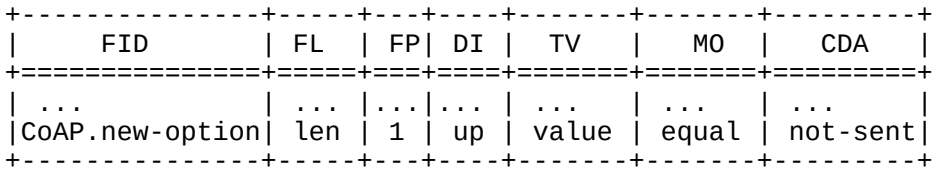
* However, S1 faces a critical problem: how to identify this new option in the Rule and communicate this identifier to S2 in a way that S2 can understand which option is involved and correctly reconstruct the header.

For example, suppose a Rule defines just a CoAP header, and S1 derives a more specific Rule including a URI-path. The entry shown in Figure 2 can be added to the derived Rule, and [RFC9363] defines appropriate identityref values (respectively fid:coap-option-uri-path, di:up, mo:equal and cda:not-sent) that can be used to communicate this Rule description to S2.

FID	FL	FP	DI	TV	MO	CDA
...	...	...	...	...	...	...
CoAP.Uri-path	len	1	up	value	equal	not-sent

Figure 2: New entry added by management

However, when A uses a recently defined option or a private option that was not known when the SCHC implementation was created, S1 cannot represent this option using existing FIDs. While S1 can still parse the CoAP header and identify the new option, it has no predefined FID to use in the Rule, as shown in Figure 3:



FID	FL	FP	DI	TV	MO	CDA
...	...	...	...	...	...	...
CoAP.new-option	len	1	up	value	equal	not-sent

Figure 3: New entry added by management

2.3. Interoperability Consequences

This disconnect between protocol option identifiers and SCHC FIDs creates several interoperability issues:

* Limited Protocol Evolution Support: New protocol options cannot be efficiently compressed without updates to the SCHC implementation.

* Incompatible Rule Exchanges: Different SCHC implementations may assign different FIDs to the same new option, making Rule exchange between them problematic.

* Implementation Complexity: SCHC implementations must maintain complex mappings between protocol options and FIDs, and these mappings may differ between implementations.

* Deployment Barriers: Deploying SCHC in environments with evolving protocols becomes difficult and requires frequent updates to SCHC implementations.

The fundamental issue is that the protocol option space and the SCHC FID space are disconnected, with no standardized or dynamic mapping between them. This disconnect seriously limits SCHC's applicability in dynamic environments and its ability to adapt to protocol evolution.

In the following sections, we will explore two approaches to address this challenge. The first approach, termed ‘syntactic,’ aims to closely align with CoAP’s native representation of options by defining three distinct fields: option delta type, option length, and option value. The second approach, called ‘semantic,’ abstracts away from the byte-level representation and introduces a generic option framework that eliminates the need for mapping between option values and Field IDs. This semantic approach focuses on the logical meaning rather than the protocol-specific encoding.

3. Syntactic compression

3.1. Overview of the Approach

SCHC compression typically uses a semantic approach where protocol fields are abstracted into generic representations with Field IDs (FIDs) that don’t directly correlate to the protocol’s encoding format. While effective for static fields, this creates challenges for dynamic protocol options as previously discussed.

Syntactic compression, as proposed in [I-D.lampin-lpwan-schc-considerations], takes a fundamentally different approach by aligning compression more closely with the protocol’s wire format. Instead of abstracting away from the protocol’s representation, syntactic compression preserves the original structure of protocol headers, including how options are encoded.

3.2. CoAP Option Encoding Background

To understand syntactic compression for CoAP options, it’s important to recall how CoAP encodes options in messages:

* Each CoAP option on the wire consists of three components:
  - Option Delta: The difference between the option number of the present option and the option number of the previous option (if any).
  - Option Length: The length of the option value in bytes.
  - Option Value: The actual option content.
* This encoding allows CoAP to efficiently represent options while maintaining extensibility.

3.3. Syntactic Representation of Options

In the syntactic approach, instead of representing a CoAP option as a single abstract Field ID, each option is decomposed into its three constituent parts as shown in Figure 4, where “x” in the Field Length of CoAP.value entry indicates the expected value:

FID	FL	FP	DI	TV	MO	CDA
CoAP.option	16	1	up	opt or delta	equal	not-sent
CoAP.length	16	1	up	value	equal	not-sent
CoAP.value	x	1	up	value	equal	not-sent

Figure 4: representation of an elided option with syntactic representation

In this representation:

* ‘CoAP.option’ can represent either the absolute CoAP option number or the delta value as encoded in the CoAP message. While this choice affects how the parser is implemented, it has minimal impact on the overall performance of the compression approach being examined in this document.

* “CoAP.length” represents the option length field.

* “CoAP.value” contains the actual option value, noted “x”.

This approach means that option identifiers remain in the CoAP numbering space rather than being converted to SCHC FIDs. This critical difference allows any option—existing, newly defined, or private—to be processed without requiring updates to the SCHC implementation’s FID mapping.

The syntactic approach offers several advantages. It provides robust support for protocol evolution, allowing new or private options to be compressed without requiring changes to SCHC implementations. It enhances interoperability since option identifiers remain in the protocol’s numbering space, enabling different SCHC implementations to exchange rules for any option. Additionally, it simplifies implementation by eliminating the need for complex mappings between protocol option identifiers and SCHC FIDs.

However, this approach also comes with notable disadvantages. It increases the number of entries required to describe each option by a factor of three, resulting in larger Rule representations. Furthermore, the syntactic approach may yield less efficient compression in certain scenarios, particularly when options must be sent rather than elided.

For instance, in a semantic approach, only the value and potentially the length is sent as residue (cf. Figure 5):

FID	FL	FP	DI	TV	MO	CDA
CoAP.new-option	var	1	up	value	equal	value-sent

Figure 5: representation of an option sent with semantic representation

The equivalent syntactic representation requires three entries, as shown in (cf. Figure 6):

FID	FL	FP	DI	TV	MO	CDA
CoAP.option	16	1	up	opt or delta	equal	not-sent
CoAP.length	16	1	up		ignore	value-sent
CoAP.value	var	1	up		ignore	value-sent

Figure 6: representation of an option sent with syntactic representation

In this case, both the option number and length (which may be encoded in just 4 bits in the CoAP message) are treated as 16-bit fields that must be sent as residue. Additionally, the option length is effectively sent twice: once in the CoAP.length field and again as part of the value's residue.

For example, a 4-byte option value would be encoded as follows in the syntactic approach:

* 0 bytes for the option number

* 2 bytes for the length field

* 0.5 bytes for the header of the value (in SCHC compressed format)

* 4 bytes for the actual value

This totals 6.5 bytes, compared to a more efficient representation in the semantic approach with 4.5 bytes.

One potential optimization would be to define a new length function that links the length of the value to the content of the CoAP.length field, avoiding the duplicate transmission of length information. However, this would add complexity to the compression mechanism.

Extending the Data Model to support the syntactic approach essentially involves creating three new Field Identifiers (FIDs) that correspond to the components of the option structure.

3.4. Conclusion

While syntactic compression offers a more generic approach that can handle protocol evolution without requiring implementation updates, it may lead to suboptimal compression efficiency and larger Rule representations. The tradeoff between flexibility and efficiency must be carefully considered based on the specific deployment scenario and requirements.

The syntactic approach demonstrates that staying too close to the byte-level representation of a protocol can compromise compression efficiency. This insight leads us to consider a hybrid approach that preserves the protocol' s option identifiers while maintaining the efficiency of semantic compression, as discussed in the next section.

4. Semantic compression

4.1. The Proposed Approach

Having examined the syntactic approach, which closely follows the byte-level representation of CoAP options, we now explore an alternative solution that maintains the efficiency of semantic compression while addressing the interoperability challenges. This approach preserves protocol option identifiers directly within SCHC Rules, eliminating the need for mapping between protocol-specific option numbers and SCHC Field Identifiers (FIDs).

The core idea is to incorporate the original protocol identifiers into the compression Rules. Since multiple protocols may reuse the same option identifier for different purposes (for example, option 8 refers to Location-Path in CoAP but Timestamp in TCP), this approach associates each option value with its protocol namespace to avoid ambiguity.

4.2. Technical Solution

The solution involves defining within SCHC an identity that references the protocol (creating a "protocol space"), followed by the specific option identifier used by that protocol. This preserves the protocol's native numbering scheme while allowing SCHC to differentiate between options from different protocols that might share the same option identifier.

Using this approach, a Rule that includes various CoAP options would directly reference the CoAP option numbers rather than abstract FIDs. For instance, the representation of a URI with two path elements (option 11) and two query elements (option 15) might look like (cf. Figure 7):

FID	FL	FP	DI	TV	MO	CDA
CoAP.option(11)	len	1	up	value	equal	not-sent
CoAP.option(11)	len	2	up		ignore	value-sent
CoAP.option(15)	len	1	up	value	equal	not-sent
CoAP.option(15)	len	2	up	value	equal	not-sent

Figure 7: Rule including options ID.

In this example, option identifiers 11 (URI-Path) and 15 (URI-Query) are directly specified, along with their position and direction, providing clear identification of which CoAP options are being compressed without requiring predefined FIDs for each option type.

5. Data Model Implementation Challenges

While this approach offers clear advantages for interoperability and protocol evolution, implementing it within the current YANG Data Model defined in [RFC9363] presents challenges. The current model defines Rule entries with a key composed of field-id, field-position, and direction-indicator, as shown in Figure 8:

```
+--:(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
      .
      .
      .
```

Figure 8: Rule entry defined by [RFC 9363].

The example shown in Figure 7 is not valid under this model because the combination of FID, position, and direction is repeated multiple times, which violates the key constraints. We cannot simply include the option value as part of the key in the existing structure.

Furthermore, it's not possible to augment the model defined in RFC 9363 to add a new leaf to the key of the list. Adding option identifiers to all entries would also be inefficient, as many fields (such as IPv6 or UDP fields) don't require this additional context.

5.1. Proposed YANG Data Model Extension

A more effective solution is to augment the current compression data model with a new list specifically designed for entries describing protocol options:

```
    +--rw schc-opt:entry-option-space* \
           [space-id option-id field-position direction-indicator]
      +--rw schc-opt:space-id                    space-type
      +--rw schc-opt:option-id                   uint32
      +--rw schc-opt:field-length                schc:fl-type
      +--rw schc-opt:field-position              uint8
      +--rw schc-opt:direction-indicator         schc:di-type
      .
      .
      .
```

Figure 9: Augmentation of SCHC Data Model to include options ID.

In this augmented model:

* The space-id defines the protocol namespace (e.g., CoAP, TCP), with values provided by the SCHC Working Group

* The option-id contains the actual option identifier as defined in the protocol's IANA registry

* The remaining elements (field-length, field-position, etc.) function as they do in the standard entry structure, but they will be diffently identified.

This approach maintains the semantic efficiency of SCHC while eliminating the need for protocol-to-FID mappings.

5.2. Implications for Residue Serialization

This design has implications for how residues are serialized. To ensure consistent interpretation, both SCHC endpoints must process entries in the same order. Therefore:

* Fields from the standard “entry” list MUST be serialized before those defined in the new “entry-option-space” list

*This constraint should be documented in [I-D.ietf-lpwan-architecture] to ensure interoperability

This approach preserves the guideline from [RFC8724] that Field Descriptors (entries) should be listed in the order they appear in the packet header.

5.3. Advantages of this Approach

The proposed approach offers several significant benefits compared to both the current SCHC model and the syntactic approach:

* It maintains the efficiency of semantic compression while eliminating the mapping problem between protocol options and SCHC FIDs

* It enables straightforward handling of newly defined or private options without requiring updates to SCHC implementations

* It allows for cleaner Rule exchange between SCHC endpoints, improving interoperability in heterogeneous environments

This approach represents a balanced solution that addresses the interoperability challenges while preserving the compression efficiency that makes SCHC valuable in constrained environments.

5.4. Impact on Current Standards

5.4.1. Compatibility with Existing Standards

The proposed extension to preserve protocol option identifiers in SCHC Rules has important implications for existing standards. Both [RFC9363] and [I-D.ietf-schc-8824-update] define specific Field Identifiers (FIDs) for CoAP options. These predefined FIDs and the proposed approach represent two different methods for identifying the same protocol elements, which creates potential compatibility issues.

5.4.2. Deprecation of Predefined CoAP Option FIDs

To maintain consistency and avoid confusion between the two approaches, the predefined CoAP option identifiers in the current standards should be deprecated in favor of the more flexible option identifier approach proposed in this document. This deprecation should be handled carefully to ensure backward compatibility with existing implementations while enabling a clear migration path to the new approach.

5.4.3. Entry Order Requirements

The proposed approach also highlights a constraint that was not explicitly included in the current Data Model. YANG does not impose a specific position for elements in a list, which has no impact on the compression process itself. However, the order becomes critical for the serialization of residues. When transmitting Rules from one endpoint to another, the order of Field Descriptors must be preserved to ensure consistent handling of residues. This requirement should be explicitly documented in [I-D.ietf-lpwan-architecture] to clarify that:

* The order of entries should not be changed when transmitted between endpoints This ordering preserves the guidance in [RFC8724] that Field Descriptors (entries) should be listed in the order they appear in the packet header

* This ordering requirement becomes particularly important with the introduction of the new “entry-option-space” list, as both types of entries (standard and option-space) must be processed in a consistent sequence across all implementations.

The statement “ordered-by user;” MUST be included in a revision of [RFC9363].

6. References

6.1. Normative References

[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>.

[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/info/rfc9363>.

6.2. Informative References

[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>.

[I-D.lampin-lpwan-schc-considerations] Lampin, Q., "SCHC design and implementation considerations", Work in Progress, Internet-Draft, draft-lampin-lpwan-schc-considerations-00, 10 November 2022, <https://datatracker.ietf.org/doc/html/draft-lampin-lpwan-schc-considerations-00>.

[I-D.ietf-schc-8824-update] Tiloca, M., Toutain, L., Martnez, I., and A. Minaburo, "Static Context Header Compression (SCHC) for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-schc-8824-update-05, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-schc-8824-update-05>.

[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>.

[GPC-SPE-207] GlobalPlatform, "Remote Application Management over CoAP Amendment M v1.0", n.d., <https://globalplatform.org/specs-library/amendment-m-remote-application-mgmt-over-coap/>.

Appendix A. YANG Data Model

This appendix defines the work in progress YANG Data Model to extend the Data Model defined in [RFC9363].

```
<CODE BEGINS> file "ietf-schc-opt@2024-12-19.yang"
module ietf-schc-opt {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-schc-opt";
  prefix schc-opt;

  import 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:p-wan@ietf.org>
     Editor:   Laurent Toutain
       <mailto:laurent.toutain@imt-atlantique.fr>
     Editor:   Ana Minaburo
       <mailto:ana@ackl.io>";
  description
     "
     Copyright (c) 2021 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 Simplified 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.

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

     This module extends the ietf-schc module to include the compound-ack
     behavior for Ack On Error as defined in RFC YYYY.
     It introduces a new leaf for Ack on Error defining the format of the
     SCHC Ack and add the possibility to send several bitmaps in a single
     answer.";
```

```
     revision 2024-12-19 {
       description
         "Initial version for RFC YYYY ";
       reference
         "RFC YYYY: OAM";
     }


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

     identity space-id-coap {
       base space-id-base-type;
       description
         "Field ID base type for IPv6 headers described in RFC 8200.";
       reference
         "RFC 8200 Internet Protocol, Version 6 (IPv6) Specification";
     }

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


    augment "/schc:schc/schc:rule/schc:nature/schc:compression" {
     list entry-option-space {
       key "space-id option-id field-position direction-indicator";
       leaf space-id {
         type space-type;
         mandatory true;
         description
           "";
       }
       leaf option-id {
         type uint32;
       }
       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 (bidirectional, only uplink, or only
             downlink).";
        }
        list target-value {
          key "index";
          uses schc:tv-struct;
          description
            "A list of values 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 schc: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, and 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 schc: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.";
        }
      }

      }
    }
    <CODE ENDS>
```

Appendix B. Examples

This appendix provides practical examples that compare different approaches to representing protocol options in SCHC. The purpose of these examples is to demonstrate the operational implications of each approach in terms of Rule size, compression efficiency, and query performance.

B.1. Test Scenario and Methodology

To provide a consistent basis for comparison, we use a common CoAP message containing various standard options along with one non-standard option (SCP82-Param). This example was chosen to illustrate how each approach handles both well-known and recently defined options.

The following CoAP message serves as our reference example:

```
0000  40 01 00 01 BD 01 61 63 63 65 6C 65 72 6F 6D 65  @.....accelerome
0010  74 65 72 73 07 6D 61 78 69 6D 75 6D 4A 64 61 74  ters.maximumJdat
0020  65 3D 74 6F 64 61 79 0A 75 6E 69 74 3D 6D 2F 73  e=today.unit=m/s
0030  5E 32 21 3C D1 E4 02 E3 05 F8 54 4C 56           ^2!<......TLV

CON  0x0001 GET
> Uri-path : b'accelerometers'
> Uri-path : b'maximum'
> Uri-query : b'date=today'
> Uri-query : b'unit=m/s^2'
> Accept : 60
> No-Response : 2
> SCP82-Param : b'TLV'
```

Figure 10: Example of a CoAP packet with options.

For each approach, we will evaluate three key metrics:

* Rule Size: The size of the CBOR-serialized compression rule in bytes

* Query Size: The size of the CORECONF query payload needed to access specific values in the rule

* Compressed Packet Size: The size of the resulting SCHC-compressed packet in bytes

In all examples, our compression rule will send residues for the Uri-Path, Uri-Query, and SCP82-Param options, while eliding the rest. This allows us to compare how different approaches handle both sent and not-sent options.

B.2. Approaches Compared

We evaluate the following approaches to option representation:

* Semantic Compression with RFC 9363: The current approach using predefined Field IDs for known options

* Universal Options: Our proposed approach preserving protocol option identifiers Merged Data Model: A combined approach that integrates both methods into a single data model

* Ordered SID Allocation: An optimized version with carefully arranged SID values

* Syntactic Compression: The approach using delta-type, length, and value fields

Each approach represents a different balance between compatibility, flexibility, and efficiency. The examples will demonstrate these trade-offs quantitatively. Let's examine each approach in detail:

Appendix C. Semantic compression

Semantic compression is the approach currently defined in [RFC8724], where each protocol field is assigned an abstract Field Identifier (FID) that represents its semantic meaning rather than its wire format. This section examines how this approach handles our example CoAP message.

C.1. Rule Definition

Using RFC 8724's informal notation, a rule matching our sample packet would be structured as follows:

RuleID 1/8							
Field	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP version	2	1	Bi	01	equal	not-sent	
CoAP Type	2	1	Dw	CON	equal	not-sent	
CoAP TKL	4	1	Bi	0	equal	not-sent	
CoAP Code	8	1	DW	0.01	equal	not-sent	
CoAP MID	16	1	Bi	0000	MSB(7)	LSB	MID
CoAP Uri-Path	var	1	Dw		ignore	value-sent	size+ value
CoAP Uri-Path	var	2	Dw		ignore	value-sent	size+ value
CoAP Uri-Query	var	1	Dw		ignore	value-sent	size+ value
CoAP Uri-Query	var	2	Dw		ignore	value-sent	size+ value
CoAP Accept	8	1	Dw	60	equal	not-sent	
CoAP No-Response	8	1	Dw	2	equal	not-sent	
CoAP SCP 82-Param	var	1	Dw		ignore	value-sent	size+ value

Figure 11: Target rule.

C.2. Implementation with RFC 9363

When implementing this rule using the RFC 9363 Data Model, we encounter a fundamental limitation: there is no identity reference (identityref) defined for the CoAP SCP82-Param option in the current standard. This illustrates the core problem addressed in this document the inability to represent new or protocol-specific options without updating the SCHC implementation. For the purpose of this comparison, we' ll assume that a theoretical extension to RFC 9363 has been created that defines identityrefs for the options in [GPC-SPE-207], with corresponding SID ranges: RFC 9363 SIDs starting at 5000 and [GPC-SPE-207] SIDs starting at 10000.

C.2.1. CBOR Serialization

With these assumptions, the rule can be serialized in CBOR format. The resulting message is 357 bytes long Figure 12.

b'a11913e7a10181a4048ca7061913bf070208010519139b091913db011913970d81a20100024101a7
061913be070208010519139a091913db011913970d81a20100024100a7061913bc070408010519139b
091913db011913970d81a20100024100a70619139f070808010519139a091913db011913970d81a201
00024101a8061913a207100801051913 9a091913de0a81a2010002410701191395 0d81a20100024100
a6061913b907d82d1913d508010519139b091913dc01191398a6061913b907d82d1913d50802051913
9b091913dc01191398a6061913bb07d82d1913d508010519139b091913dc01191398a6061913bb07d8
2d1913d508020519139b091913dc01191398a7061913a407080801051913 9b091913db011913970d81
a2010002413ca7061913ae07080801051913 9b091913db011913970d81a20100024102a60619271107
d82d1913d508010519139b091913dc011913981822081821081823 1913e0'

Figure 12: CBOR serialisation.

The diagnostic representation provides insight into the structure of this serialized rule:

```
Deltas in entry part:
- 6: field-id
- 7: field-length
- 8: field-position
- 5: direction-indicator
- 9: matching-operator
- 1: comp-decomp-action
- 10: matching-operator-value
- 13: target-value

Deltas in the rule part:
- 33: rule-id-length
- 34: rule-id-value
- 35: rule-nature

{5095: {1: [{4: [
  {6: 5055, 7: 2, 8: 1, 5: 5019, 9: 5083, 1: 5015, 13: [{1: 0, 2: h'01'}]},
  {6: 5054, 7: 2, 8: 1, 5: 5018, 9: 5083, 1: 5015, 13: [{1: 0, 2: h'00'}]},
  {6: 5052, 7: 4, 8: 1, 5: 5019, 9: 5083, 1: 5015, 13: [{1: 0, 2: h'00'}]},
  {6: 5023, 7: 8, 8: 1, 5: 5018, 9: 5083, 1: 5015, 13: [{1: 0, 2: h'01'}]},
  {6: 5026, 7: 16, 8: 1, 5: 5018, 9: 5086,
                10: [{1: 0, 2: h'07'}], 1: 5013, 13: [{1: 0, 2: h'00'}]},
  {6: 5049, 7: 45(5077), 8: 1, 5: 5019, 9: 5084, 1: 5016},
  {6: 5049, 7: 45(5077), 8: 2, 5: 5019, 9: 5084, 1: 5016},
  {6: 5051, 7: 45(5077), 8: 1, 5: 5019, 9: 5084, 1: 5016},
  {6: 5051, 7: 45(5077), 8: 2, 5: 5019, 9: 5084, 1: 5016},
  {6: 5028, 7: 8, 8: 1, 5: 5019, 9: 5083, 1: 5015, 13: [{1: 0, 2: h'3C'}]},
  {6: 5038, 7: 8, 8: 1, 5: 5019, 9: 5083, 1: 5015, 13: [{1: 0, 2: h'02'}]},
  {6: 10001, 7: 45(5077), 8: 1, 5: 5019, 9: 5084, 1: 5016}],
34: 8, 33: 8, 35: 5088}]}}
```

Figure 13: CBOR diagnostic notation.

Note that the encoding of rule-id-value, rule-id-length, and rule-nature is not optimal because the delta values are higher than 23, requiring 2 bytes each in the CBOR encoding.

C.2.2. CORECONF Query Example

To access the target value of the Accept option in this rule, a CORECONF query would be structured as follows. The size of the CoAP payload for this query is 14 bytes:

```
      REQ: FETCH </c>
              (Content-Format: application/yang-identifiers+cbor-seq)
         [5115,     / .../target-value/value
          1,        / rule-id-value
          8,        / rule-id-length
          5028,     / fid-coap-option-accept
          1,        / field-position
          5019,     / direction-indicator
          0]        / target-value/index
```

Figure 14: CORECONF query to Accept TV.

C.2.3. Compressed Packet

When applied to our sample CoAP message, the semantic compression approach produces a SCHC packet of 389 bits (49 bytes with byte alignment):

```
0800f30b1b1b2b632b937b6b2ba32b939bb6b0bc34b6bab6d3230ba329eba37b230bcd3ab734ba1eb697b9af1
91aa262b0/389
```

Figure 15: SCHC compressed packet.

C.2.4. Analysis

The semantic compression approach demonstrates both significant strengths and notable limitations. On the positive side, it achieves an efficient compressed packet size of only 49 bytes, making it suitable for constrained networks where bandwidth is at a premium. The query size is also relatively small at 14 bytes, which enables efficient rule management operations. Additionally, the semantic representation of fields provides clarity by abstracting protocol details into meaningful identifiers.

However, this approach faces important limitations that impact its flexibility. Most critically, it cannot handle new options without predefined Field Identifiers, which creates a dependency on standards updates. When new protocol options emerge, the approach requires formal extension of the standard to define appropriate identifiers, creating potential delays and compatibility issues. While the rule serialization size is moderate at 357 bytes, this still represents significant overhead in highly constrained environments. These limitations illustrate why semantic compression works well in static, predictable environments but presents challenges for protocol evolution and interoperability in more dynamic contexts where protocols frequently evolve.

C.3. Universal Options

The Universal Options approach preserves protocol option identifiers directly within SCHC Rules, eliminating the need for predefined Field Identifiers for each option type. This section examines how this approach handles our example CoAP message.

C.3.1. Implementation Approach

In this approach, we assign SIDs starting from 7000 to the YANG Data Model augmentation defined in this document. All CoAP options are represented by a combination of a space ID (indicating the protocol namespace, in this case CoAP) and the option identifier as used in the CoAP protocol. This allows the SCP82-Param option to be processed like any other option, regardless of whether it was defined when the SCHC implementation was created.

C.3.2. CBOR Serialization

The CBOR serialization of the rule using the Universal Options approach is 481 bytes long:

b’a11913e7a10181a4048ca7061913bf070208010519139b091913db011913970d81a20100024101a7
061913be070208010519139a091913db011913970d81a20100024100a7061913bc070408010519139b
091913db011913970d81a20100024100a70619139f070808010519139a091913db011913970d81a201
00024101a8061913a207100801051913 9a091913de0a81a2010002410701191395 0d81a20100024100
a719077c191b5a19077b0b190775d82d1913d5190776011907741913 9b1907771913dc190770191398
a719077c191b5a19077b0b190775d82d1913d5190776021907741913 9b1907771913dc190770191398
a719077c191b5a19077b0f190775d82d1913d5190776011907741913 9b1907771913dc190770191398
a719077c191b5a19077b0f190775d82d1913d5190776021907741913 9b1907771913dc190770191398
a819077c191b5a19077b1119077508190776011907741913 9b1907771913db1907701913971907 7d81
a2010002413ca819077c191b5a19077b190102190775d82d1913d5190776011907741913 9b19077719
13db1907701913971907 7d81a20100024102a719077c191b5a19077b190807190775d82d1913d51907
76011907741913 9b1907771913dc1907701913981822081821081823 1913e0’

Figure 16: CBOR serialisation.

The diagnostic representation of the CBOR message is the following:

```
Deltas in entry part:
- 6: field-id                      **1916: space-id
- 7: field-length                  **1915: option-id
- 8: field-position                **1909: field-length
- 5: direction-indicator           **1910: field-position
- 9: matching-operator             **1908: direction-indicator
- 1: comp-decomp-action            **1911: matching-operator
- 10: matching-operator-value      **1917: target-value
- 13: target-value

Deltas in the rule part:
* 33: rule-id-length
* 34: rule-id-value
* 35: rule-nature

{5095: {1: [{4: [
  {6: 5055, 7: 2, 8: 1, 5: 5019, 9: 5083, 1: 5015, 13: [{1: 0, 2: h'01'}]},
  {6: 5054, 7: 2, 8: 1, 5: 5018, 9: 5083, 1: 5015, 13: [{1: 0, 2: h'00'}]},
  {6: 5052, 7: 4, 8: 1, 5: 5019, 9: 5083, 1: 5015, 13: [{1: 0, 2: h'00'}]},
  {6: 5023, 7: 8, 8: 1, 5: 5018, 9: 5083, 1: 5015, 13: [{1: 0, 2: h'01'}]},
  {6: 5026, 7: 16, 8: 1, 5: 5018, 9: 5086,
                   10: [{1: 0, 2: h'07'}], 1: 5013, 13: [{1: 0, 2: h'00'}]},
  {1916: 7002, 1915: 11, 1909: 45(5077), 1910: 1, 1908: 5019, 1911: 5084, 1904: 5016},
  {1916: 7002, 1915: 11, 1909: 45(5077), 1910: 2, 1908: 5019, 1911: 5084, 1904: 5016},
  {1916: 7002, 1915: 15, 1909: 45(5077), 1910: 1, 1908: 5019, 1911: 5084, 1904: 5016},
  {1916: 7002, 1915: 15, 1909: 45(5077), 1910: 2, 1908: 5019, 1911: 5084, 1904: 5016},
  {1916: 7002, 1915: 17, 1909: 8, 1910: 1, 1908: 5019, 1911: 5083, 1904: 5015,
                                                  1917: [{1: 0, 2: h'3C'}]},
  {1916: 7002, 1915: 258, 1909: 45(5077), 1910: 1, 1908: 5019, 1911: 5083, 1904: 5015,
                                                  1917: [{1: 0, 2: h'02'}]},
  {1916: 7002, 1915: 2055, 1909: 45(5077), 1910: 1, 1908: 5019, 1911: 5084, 1904: 5016}],
34: 8, 33: 8, 35: 5088}]}}
```

Figure 17: CBOR serialisation.

It's important to note that in the CoAP options part, the delta values are very large and require 3 bytes each for CBOR encoding. This is because we've used a separate range of SIDs for the augmentation, following the standard allocation procedure where each YANG Data Model has its own SID range.

C.3.3. CORECONF Query Example

A CORECONF query to access the target value of the Accept option in this approach would be structured as follows. The size of the CoAP payload is 15 bytes:

```
REQ: FETCH </c>
        (Content-Format: application/yang-identifiers+cbor-seq)
   [7019,     / .../target-value/value
    1,        / rule-id-value
    8,        / rule-id-length
    7002,     / space-id-value
    17,       / option-id
    1,        / field-position
    5019,     / direction-indicator
    0]        / target-value/index
```

Figure 18: CORECONF query to Accept TV.

C.3.4. Compressed Packet

When applied to our sample CoAP message, the Universal Options approach produces a SCHC packet with the same size as the semantic approach: 49 bytes with byte alignment.

C.3.5. Analysis

The Universal Options approach demonstrates several important characteristics when compared to the semantic approach.

The primary advantage of this approach is its flexibility and future-proofing. By preserving the protocol's native option identifiers, it can handle any option—including newly defined or private options—without requiring updates to the SCHC implementation. This is particularly evident in how it handles the SCP82-Param option (2055) without requiring predefined Field Identifiers. The compressed packet size remains efficient at 49 bytes, identical to the semantic approach, demonstrating that flexibility doesn't come at the cost of compression efficiency at the packet level.

However, there are trade-offs. The CBOR serialization of the rule is significantly larger at 481 bytes compared to 357 bytes for the semantic approach. This increased size is primarily due to the additional information needed to represent both protocol space identifiers and option values, along with less efficient delta encoding due to SID allocation in separate ranges. The query size is also slightly larger (15 bytes vs. 14 bytes), though this difference is minimal.

Overall, the Universal Options approach provides significantly improved flexibility and interoperability at the cost of larger rule serialization size. This trade-off may be acceptable in many applications, particularly where protocol evolution and interoperability are more critical than rule storage efficiency.

C.4. Merged Data Model Approach

The Merged approach combines elements from both the semantic compression and Universal Options approaches into a single, unified YANG Data Model. This section examines how this integrated approach handles our example CoAP message.

C.4.1. Implementation Approach

Instead of maintaining two separate Data Models (RFC 9363 and Universal Options), this approach merges them into a single model, which we' ll refer to as "9363bis." The SID allocation process remains unchanged, with SIDs allocated automatically based on alphabetical ordering of nodes in the YANG model. The key advantage of this approach is that it provides a unified framework that can represent both predefined fields using semantic compression and protocol-specific options using the Universal Options approach.

C.4.2. CBOR Serialization

The CBOR serialization of the rule using the Merged approach is 400 bytes in size:

b'a11913e9a10181a4048ca7171913bf1818021819011619139b181a1913db12191397181e81a20100
024101a7171913be1818021819011619139a181a1913db12191397181e81a20100024100a7171913bc
1818041819011619139b181a1913db12191397181e81a20100024100a71719139f1818081819011619
139a181a1913db12191397181e81a20100024101a8171913a2181810181901161913 9a181a1913de18
1b81a2010002410712191395181e81a20100024100a70e1913e60d0b07d82d1913d508010619139b09
1913dc02191398a70e1913e60d0b07d82d1913d508020619139b091913dc02191398a70e1913e60d0f
07d82d1913d508010619139b091913dc02191398a70e1913e60d0f07d82d1913d508020619139b0919
13dc02191398a80e1913e60d11070808010619139b091913db021913970f81a2010002413ca80e1913
e60d19010207d82d1913d508010619139b091913db021913970f81a20100024102a70e1913e60d1908
0707d82d1913d508010619139b091913dc021913981833081832081834 1913e0'

Figure 19: CBOR serialisation.

The diagnostic representation reveals the structure of this serialized rule:

```
Deltas in entry part:
- 23: field-id                    -14: space-id
* 24: field-length                -11: option-id
* 25: field-position              -7: field-length
- 22: direction-indicator         -8: field-position
* 26: matching-operator           -6: direction-indicator
- 18: comp-decomp-action          -9: matching-operator
* 27: matching-operator-value     -15: target-value
* 30: target-value

Deltas in the rule part:
* 50: rule-id-length
* 51: rule-id-value
* 52: rule-nature

{5097: {1: [{4: [
  {23: 5055, 24: 2, 25: 1, 22: 5019, 26: 5083, 18: 5015, 30: [{1: 0, 2: h'01'}]},
  {23: 5054, 24: 2, 25: 1, 22: 5018, 26: 5083, 18: 5015, 30: [{1: 0, 2: h'00'}]},
  {23: 5052, 24: 4, 25: 1, 22: 5019, 26: 5083, 18: 5015, 30: [{1: 0, 2: h'00'}]},
  {23: 5023, 24: 8, 25: 1, 22: 5018, 26: 5083, 18: 5015, 30: [{1: 0, 2: h'01'}]},
  {23: 5026, 24: 16, 25: 1, 22: 5018, 26: 5086,
                       27: [{1: 0, 2: h'07'}], 18: 5013, 30: [{1: 0, 2: h'00'}]},
  {14: 5094, 13: 11, 7: 45(5077), 8: 1, 6: 5019, 9: 5084, 2: 5016},
  {14: 5094, 13: 11, 7: 45(5077), 8: 2, 6: 5019, 9: 5084, 2: 5016},
  {14: 5094, 13: 15, 7: 45(5077), 8: 1, 6: 5019, 9: 5084, 2: 5016},
  {14: 5094, 13: 15, 7: 45(5077), 8: 2, 6: 5019, 9: 5084, 2: 5016},
  {14: 5094, 13: 17, 7: 8, 8: 1, 6: 5019, 9: 5083, 2: 5015,
                                                    15: [{1: 0, 2: h'3C'}]},
  {14: 5094, 13: 258, 7: 45(5077), 8: 1, 6: 5019, 9: 5083, 2: 5015,
                                                    15: [{1: 0, 2: h'02'}]},
  {14: 5094, 13: 2055, 7: 45(5077), 8: 1, 6: 5019, 9: 5084, 2: 5016}],
51: 8, 50: 8, 52: 5088}]}}
```

Figure 20: CBOR serialisation.

It can be observed that some delta values are higher than 23, requiring 2 bytes for their encoding in CBOR. This is a result of the automatic SID allocation based on alphabetical ordering, which doesn' t optimize for efficient deltas.

C.4.3. CORECONF Query and Compressed Packet

The CORECONF query and compressed packet sizes remain consistent with previous approaches. The query size is 15 bytes (similar to the Universal Options approach), and the compressed packet size remains at 49 bytes.

C.4.4. Analysis

The Merged Data Model approach offers an interesting middle ground between the semantic and Universal Options approaches. By combining both approaches in a single model, it provides the benefits of compatibility with existing semantic compression while adding the flexibility to handle new or protocol-specific options.

The CBOR serialization size (400 bytes) falls between the semantic approach (357 bytes) and the Universal Options approach (481 bytes), reflecting its hybrid nature. This represents a reasonable compromise that balances compatibility, flexibility, and efficiency.

A key advantage of this approach is its unified framework, which eliminates the need to choose between different compression approaches. Implementations can leverage semantic compression for well-known fields while still maintaining the ability to handle any new protocol options through the Universal Options mechanism.

However, the automatic SID allocation based on alphabetical ordering leads to some inefficiency in delta encoding. As shown in the diagnostic representation, some delta values require 2 bytes for encoding, which increases the serialization size. This suggests that further optimization of SID allocation could improve efficiency, which is explored in the next approach.

Overall, the Merged approach offers a pragmatic solution that balances the strengths of both semantic and Universal Options approaches while mitigating some of their respective limitations. It provides a path forward that maintains backward compatibility while enabling future extensibility.

C.5. Ordered SID Allocation Approach

The Ordered approach represents a further optimization of the Merged approach through strategic manual allocation of SIDs. While it uses the exact same YANG Data Model as the Merged approach, it differs in how SIDs are assigned to the nodes in the model.

C.5.1. Implementation Approach

In standard YANG Data Models, SIDs are typically allocated automatically based on alphabetical ordering of nodes or through sequential assignment. This automatic allocation, while convenient, often produces suboptimal delta values when serializing CBOR-encoded rules. The Ordered approach intervenes in this process by manually editing the SID file to optimize delta values specifically for CBOR encoding efficiency. By carefully arranging SIDs to ensure that

frequently used nodes have closely related values, this approach minimizes the number of bytes required to encode the deltas between adjacent SIDs in the CBOR representation.

C.5.2. CBOR Serialization

The CBOR serialization of the rule using the Ordered approach is 376 bytes in size:

```
b’a119139fa11781a4178ca7171914222802270116191 3fe2619143e121913fa2281a20100024101a7
17191421280227011619 13fd2619143e121913fa2281a20100024100a71719141f28042701161913fe
2619143e121913fa2281a20100024100a7171914022808270116191 3fd2619143e121913fa2281a201
00024101a81719140528102701161913fd261914412581a2010002410712 1913f82281a20100024100
a70e1914490d0b07d82d1914380801061913fe0919143f021913fba70e1914490d0b07d82d19143808
02061913fe0919143f021913fba70e1914490d0f07d82d1914380801061913fe0919143f021913fba7
0e1914490d0f07d82d1914380802061913fe0919143f021913fba80e1914490d1107080801061913fe
0919143e021913fa0f81a2010002413ca80e1914490d19010207d82d1914380801061913fe0919143e
021913fa0f81a20100024102a70e1914490d19080707d82d1914380801061913fe0919143f021913fb
2a0829082b191443
```

Figure 21: CBOR serialisation.

The diagnostic representation shows how this approach affects the delta values:

```
Deltas in entry part:
- 23: field-id                    -14: space-id
* -9: field-length                -11: option-id
* -8: field-position              -7: field-length
- 22: direction-indicator         -8: field-position
* -7: matching-operator           -6: direction-indicator
- 18: comp-decomp-action          -9: matching-operator
* -6: matching-operator-value     -15: target-value
* -3: target-value

Deltas in the rule part:
* -11: rule-id-length
* -10: rule-id-value
* -12: rule-nature

{5023: {23: [{23: [
  {23: 5154, -9: 2, -8: 1, 22: 5118, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'01'}]},
  {23: 5153, -9: 2, -8: 1, 22: 5117, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'00'}]},
  {23: 5151, -9: 4, -8: 1, 22: 5118, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'00'}]},
  {23: 5122, -9: 8, -8: 1, 22: 5117, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'01'}]},
  {23: 5125, -9: 16, -8: 1, 22: 5117, -7: 5185,
                       -6: [{1: 0, 2: h'07'}], 18: 5112, -3: [{1: 0, 2: h'00'}]},
  {14: 5193, 13: 11, 7: 45(5176), 8: 1, 6: 5118, 9: 5183, 2: 5115},
  {14: 5193, 13: 11, 7: 45(5176), 8: 2, 6: 5118, 9: 5183, 2: 5115},
  {14: 5193, 13: 15, 7: 45(5176), 8: 1, 6: 5118, 9: 5183, 2: 5115},
  {14: 5193, 13: 15, 7: 45(5176), 8: 2, 6: 5118, 9: 5183, 2: 5115},
  {14: 5193, 13: 17, 7: 8, 8: 1, 6: 5118, 9: 5182, 2: 5114,
                                                    15: [{1: 0, 2: h'3C'}]},
  {14: 5193, 13: 258, 7: 45(5176), 8: 1, 6: 5118, 9: 5182, 2: 5114,
                                                    15: [{1: 0, 2: h'02'}]},
  {14: 5193, 13: 2055, 7: 45(5176), 8: 1, 6: 5118, 9: 5183, 2: 5115}],
-11: 8, -10: 8, -12: 5187}]}}
```

Figure 22: CBOR serialisation.

The key innovation in this approach is visible in the diagnostic representation: by carefully arranging SIDs, many of the delta values are now negative numbers with small absolute values. In CBOR, small integers (both positive and negative) can be encoded in a single byte, leading to significant space savings compared to the larger deltas in the Merged approach.

C.5.3. SID Allocation Strategy

The SID allocation file was manually edited to optimize delta values, as shown in the excerpt below:

```
5023;data;/ietf-schc:schc
...
5030;data;/ietf-schc:schc/rule/window-size
5031;data;/ietf-schc:schc/rule/w-size
5032;data;/ietf-schc:schc/rule/tile-size
5033;data;/ietf-schc:schc/rule/tile-in-all-1
5034;data;/ietf-schc:schc/rule/rule-nature
5035;data;/ietf-schc:schc/rule/rule-id-value
5036;data;/ietf-schc:schc/rule/rule-id-length
5037;data;/ietf-schc:schc/rule/retransmission-timer/ticks-numbers
5038;data;/ietf-schc:schc/rule/retransmission-timer/ticks-duration
5039;data;/ietf-schc:schc/rule/retransmission-timer
5040;data;/ietf-schc:schc/rule/rcs-algorithm
5041;data;/ietf-schc:schc/rule/maximum-packet-size
5042;data;/ietf-schc:schc/rule/max-interleaved-frames
5043;data;/ietf-schc:schc/rule/dtag-size
5044;data;/ietf-schc:schc/rule/direction
5045;data;/ietf-schc:schc/rule/ack-behavior
5046;data;/ietf-schc:schc/rule
5047;data;/ietf-schc:schc/rule/fcn-size
5048;data;/ietf-schc:schc/rule/fragmentation-mode
5049;data;/ietf-schc:schc/rule/inactivity-timer
5050;data;/ietf-schc:schc/rule/inactivity-timer/ticks-duration
5051;data;/ietf-schc:schc/rule/inactivity-timer/ticks-numbers
5052;data;/ietf-schc:schc/rule/l2-word-size
5053;data;/ietf-schc:schc/rule/max-ack-requests
...
5060;data;/ietf-schc:schc/rule/entry/field-length
5061;data;/ietf-schc:schc/rule/entry/field-position
5062;data;/ietf-schc:schc/rule/entry/matching-operator
5063;data;/ietf-schc:schc/rule/entry/matching-operator-value
5064;data;/ietf-schc:schc/rule/entry/matching-operator-value/index
5065;data;/ietf-schc:schc/rule/entry/matching-operator-value/value
5066;data;/ietf-schc:schc/rule/entry/target-value
5067;data;/ietf-schc:schc/rule/entry/target-value/index
5068;data;/ietf-schc:schc/rule/entry/target-value/value
5069;data;/ietf-schc:schc/rule/entry
5070;data;/ietf-schc:schc/rule/entry-option-space
5071;data;/ietf-schc:schc/rule/entry-option-space/comp-decomp-action
5072;data;/ietf-schc:schc/rule/entry-option-space/comp-decomp-action-value
5073;data;/ietf-schc:schc/rule/entry-option-space/comp-decomp-action-value/index
5074;data;/ietf-schc:schc/rule/entry-option-space/comp-decomp-action-value/value
5075;data;/ietf-schc:schc/rule/entry-option-space/direction-indicator
5076;data;/ietf-schc:schc/rule/entry-option-space/field-length
5077;data;/ietf-schc:schc/rule/entry-option-space/field-position
5078;data;/ietf-schc:schc/rule/entry-option-space/matching-operator
5079;data;/ietf-schc:schc/rule/entry-option-space/matching-operator-value
5080;data;/ietf-schc:schc/rule/entry-option-space/matching-operator-value/index
```

```
5081;data;/ietf-schc:schc/rule/entry-option-space/matching-operator-value/value
5082;data;/ietf-schc:schc/rule/entry-option-space/option-id
5083;data;/ietf-schc:schc/rule/entry-option-space/space-id
5084;data;/ietf-schc:schc/rule/entry-option-space/target-value
5085;data;/ietf-schc:schc/rule/entry-option-space/target-value/index
5086;data;/ietf-schc:schc/rule/entry-option-space/target-value/value
5087;data;/ietf-schc:schc/rule/entry/comp-decomp-action
5088;data;/ietf-schc:schc/rule/entry/comp-decomp-action-value
5089;data;/ietf-schc:schc/rule/entry/comp-decomp-action-value/index
5090;data;/ietf-schc:schc/rule/entry/comp-decomp-action-value/value
5091;data;/ietf-schc:schc/rule/entry/direction-indicator
5092;data;/ietf-schc:schc/rule/entry/field-id
```

Figure 23: CBOR serialisation.

The allocation strategy focuses on several key principles:

* Grouping related nodes with consecutive SIDs to minimize delta values

* Positioning frequently used nodes strategically to ensure small deltas

* Using both positive and negative deltas to maximize single-byte encoding opportunities

* Arranging rule-related fields to minimize the encoding size of rule metadata

C.5.4. CORECONF Query and Compressed Packet

As with previous approaches, the CORECONF query size (15 bytes) and compressed packet size (49 bytes) remain consistent. The optimization affects only the rule serialization size, not the operational efficiency of queries or the compression ratio of actual packets.

C.5.5. Analysis

The Ordered SID Allocation approach demonstrates the significant impact that strategic SID assignment can have on rule serialization efficiency. By reducing the CBOR serialization size from 400 bytes in the Merged approach to 376 bytes, it achieves a 6% reduction in size while maintaining identical functionality and compatibility.

This approach is particularly noteworthy because it achieves this efficiency gain without any changes to the YANG Data Model itself—only the SID allocation is modified. This makes it a non-invasive optimization that can be applied to existing models without affecting their structure or semantics.

The resulting serialization size (376 bytes) is only slightly larger than the original semantic approach (357 bytes), while maintaining all the flexibility benefits of the Universal Options approach. This represents an excellent compromise that nearly eliminates the size penalty of the more flexible approach.

The Ordered approach demonstrates that thoughtful SID allocation can significantly improve encoding efficiency for CBOR-serialized SCHC Rules. This optimization technique could be valuable in constrained environments where rule transmission and storage efficiency are critical concerns.

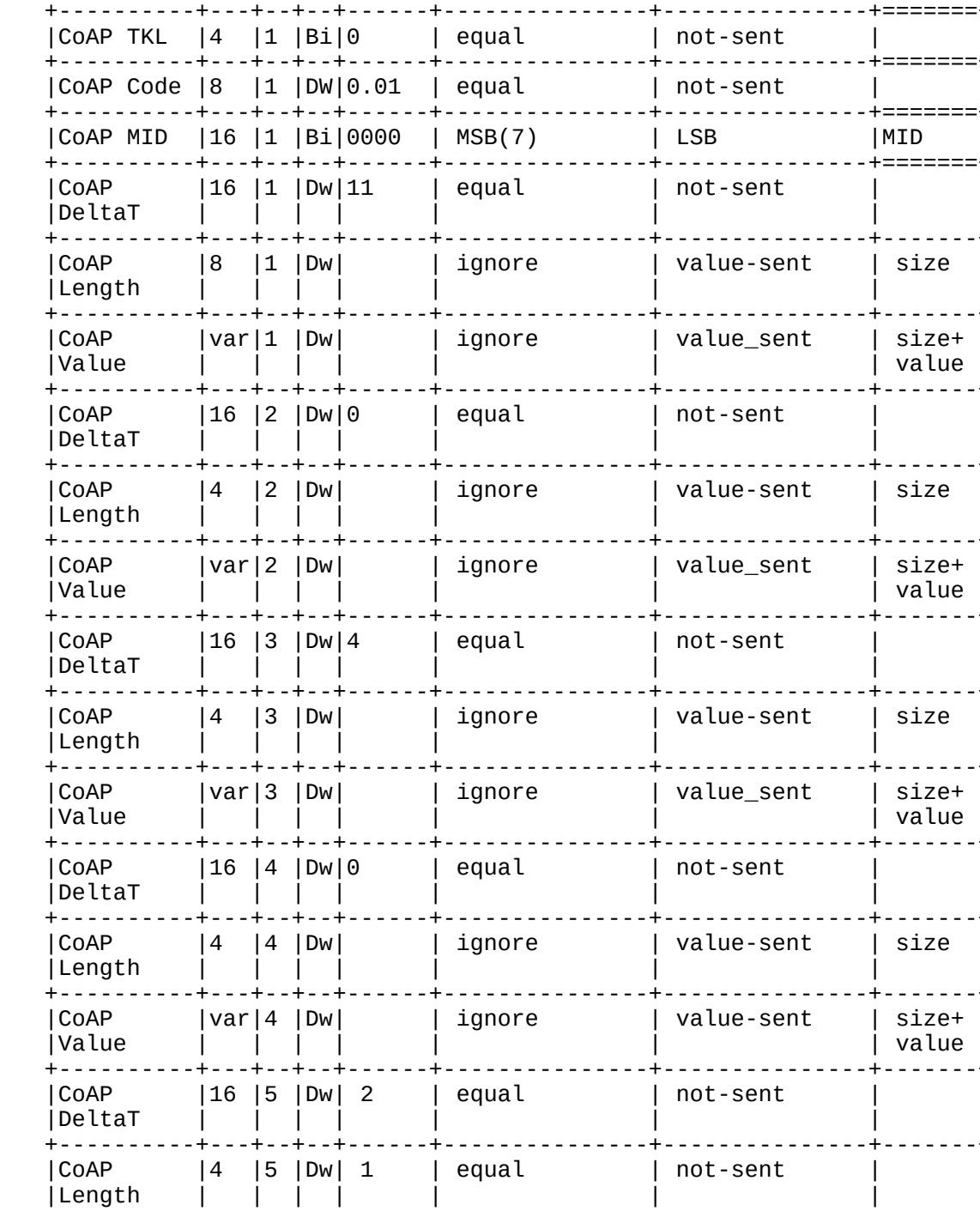
Appendix D. Syntatic compression

The syntactic approach processes all options uniformly by decomposing each CoAP option into its constituent components. Rather than treating an entire option as a single field, this approach treats the delta type, length, and value components that make up a CoAP option as separate fields. While the core compression algorithm remains unchanged, the parsing phase must be modified to accommodate this decomposed representation of the header. From a Data Model perspective, three new Field Identifiers (FIDs) need to be defined. The specific SID values assigned have minimal impact since they function purely as identifiers without CORECONF delta encoding considerations.

D.1. Rule specification

As shown in the transition from Figure 11 to Figure 24, the Field Position (FP) parameter plays a crucial role in maintaining the proper ordering of these option components.

RuleID 9/8							
Field	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP version	2	1	Bi	01	equal	not-sent	
CoAP Type	2	1	Dw	CON	equal	not-sent	



CoAP TKL	4	1	Bi	0	equal	not-sent	
CoAP Code	8	1	DW	0.01	equal	not-sent	
CoAP MID	16	1	Bi	0000	MSB(7)	LSB	MID
CoAP DeltaT	16	1	Dw	11	equal	not-sent	
CoAP Length	8	1	Dw		ignore	value-sent	size
CoAP Value	var	1	Dw		ignore	value_sent	size+ value
CoAP DeltaT	16	2	Dw	0	equal	not-sent	
CoAP Length	4	2	Dw		ignore	value-sent	size
CoAP Value	var	2	Dw		ignore	value_sent	size+ value
CoAP DeltaT	16	3	Dw	4	equal	not-sent	
CoAP Length	4	3	Dw		ignore	value-sent	size
CoAP Value	var	3	Dw		ignore	value_sent	size+ value
CoAP DeltaT	16	4	Dw	0	equal	not-sent	
CoAP Length	4	4	Dw		ignore	value-sent	size
CoAP Value	var	4	Dw		ignore	value-sent	size+ value
CoAP DeltaT	16	5	Dw	2	equal	not-sent	
CoAP Length	4	5	Dw	1	equal	not-sent	

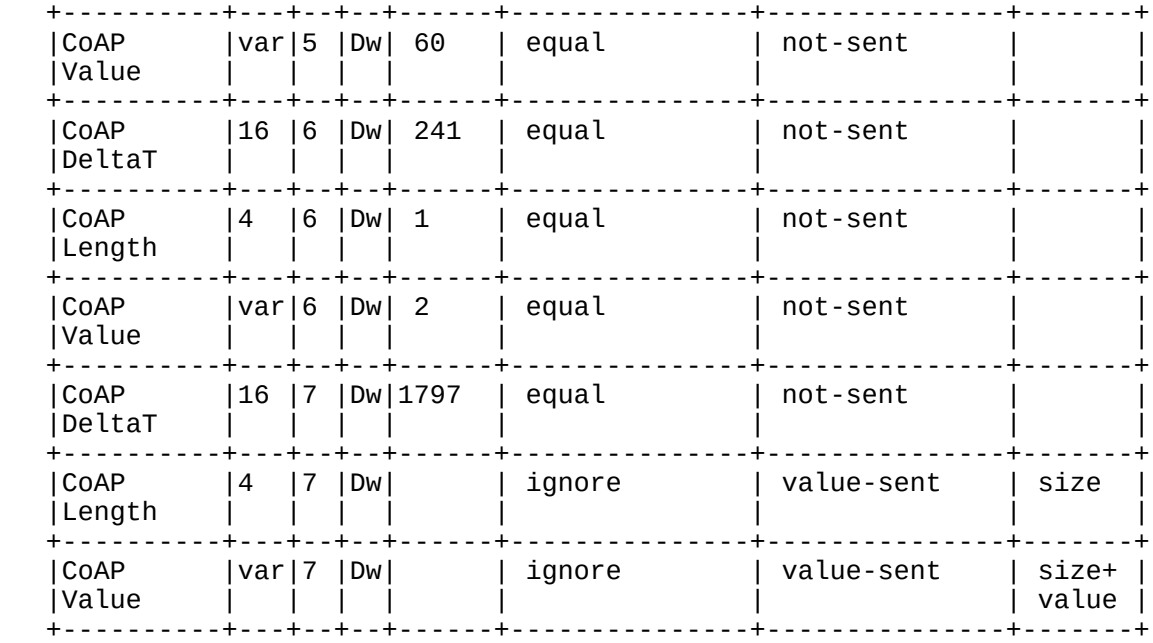


Figure 24: Target rule.

A notable limitation of this approach concerns the SCHC Field Length of the 'CoAP Length' fields, which restricts the rule's applicability. For example, the first Uri-path option requires 8 bits, instead of 4, for its Field Length because the value 'accelerometers' is 14 bytes long, necessitating the escape value 0xD to encode the length on an additional byte. Consequently, if shorter values need to be handled, separate rules would be required.

D.2. Compressed packet

The compression residue with the rule ID is 409 bit-long or 52 byte-long with the alignment (see Figure 25).

```
090087730b1b1b2b632b937b6b2ba32b939bbb6b0bc34b6bab6d53230ba329eba37b230bcd
53ab734ba1eb697b9af191aa262b00
```

Figure 25: SCHC compressed packet.

The increased size of the syntactic approach is primarily due to redundant transmission of option lengths. The length information must be sent twice: first to reconstruct the ‘CoAP Length’ field in the option header, and again to specify the size of the ‘CoAP Value’ residue. This duplication contributes significantly to the lower compression efficiency compared to other approaches.

D.3. CORECONF Query Example

A CORECONF query to access the target value of the Accept with is in fith position. The size of the CoAP payload is also 14 bytes:

```
REQ: FETCH </c>
        (Content-Format: application/yang-identifiers+cbor-seq)
   [5115,     / .../target-value/value
    9,        / rule-id-value
    8,        / rule-id-length
    8003,     / fid-coap-value (from another YANG DM)
    5,        / field-position
    5019,     / direction-indicator
    0]        / target-value/index
```

Figure 26: CORECONF query to Accept TV.

D.4. CBOR Serialization

The serialization uses the manually assigned sid to minize the representation. The result is 718 byte-long.

Deltas in entry part:

- 23: field-id
* -9: field-length
* -8: field-position
- 22: direction-indicator
* -7: matching-operator
- 18: comp-decomp-action
* -6: matching-operator-value
* -3: target-value
-14: space-id
-11: option-id
-7: field-length
-8: field-position
-6: direction-indicator
-9: matching-operator
-15: target-value

Deltas in the rule part:

* -11: rule-id-length
* -10: rule-id-value
* -12: rule-nature

```
{5023: {23: [
  {23: [{23: 5154, -9: 2, -8: 1, 22: 5118, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'01'}]},
  {23: 5153, -9: 2, -8: 1, 22: 5117, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'00'}]},
  {23: 5151, -9: 4, -8: 1, 22: 5118, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'00'}]},
  {23: 5122, -9: 8, -8: 1, 22: 5117, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'01'}]},
  {23: 5125, -9: 16, -8: 1, 22: 5117, -7: 5185, -6: [{1: 0, 2: h'07'}],
                                                    18: 5112, -3: [{1: 0, 2: h'00'}]},
  {23: 8001, -9: 4, -8: 1, 22: 5118, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'0B'}]},
  {23: 8002, -9: 12, -8: 1, 22: 5118, -7: 5183, 18: 5115},
  {23: 8003, -9: 45(5176), -8: 1, 22: 5118, -7: 5183, 18: 5115},
  {23: 8001, -9: 4, -8: 2, 22: 5118, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'00'}]},
  {23: 8002, -9: 4, -8: 2, 22: 5118, -7: 5183, 18: 5115},
  {23: 8003, -9: 45(5176), -8: 2, 22: 5118, -7: 5183, 18: 5115},
  {23: 8001, -9: 4, -8: 3, 22: 5118, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'04'}]},
  {23: 8002, -9: 4, -8: 3, 22: 5118, -7: 5183, 18: 5115},
  {23: 8003, -9: 45(5176), -8: 3, 22: 5118, -7: 5183, 18: 5115},
  {23: 8001, -9: 4, -8: 4, 22: 5118, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'00'}]},
  {23: 8002, -9: 4, -8: 4, 22: 5118, -7: 5183, 18: 5115},
  {23: 8003, -9: 45(5176), -8: 4, 22: 5118, -7: 5183, 18: 5115},
  {23: 8001, -9: 4, -8: 5, 22: 5118, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'02'}]},
  {23: 8002, -9: 4, -8: 5, 22: 5118, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'01'}]},
  {23: 8003, -9: 8, -8: 5, 22: 5118, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'3C'}]},
  {23: 8001, -9: 4, -8: 6, 22: 5118, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'F1'}]},
  {23: 8002, -9: 4, -8: 6, 22: 5118, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'01'}]},
  {23: 8003, -9: 8, -8: 6, 22: 5118, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'02'}]},
  {23: 8001, -9: 8, -8: 7, 22: 5118, -7: 5182, 18: 5114, -3: [{1: 0, 2: h'0705'}]},
  {23: 8002, -9: 4, -8: 7, 22: 5118, -7: 5183, 18: 5115},
  {23: 8003, -9: 8, -8: 7, 22: 5118, -7: 5183, 18: 5115}], -11: 9, -10: 8, -12: 5187}]}}
```

Appendix E. Summary

	RFC9363	Univ Opt	merged	ordered	Syntactic	Revised
CORECONF	357	481	400	376	718	
Query	14	15	15	15	14	
SCHC pkt	49	49	49	49	52	

Appendix F. 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

Quentin Lampin
Orange
Email: quentin.lampin@orange.com

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

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-16440 Kista
Sweden
Email: marco.tiloca@ri.se

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