SCHC Working Group
Internet-Draft
Intended status: Informational
Expires: 6 December 2026

Q. Lampin
Orange

4 June 2026

VOICI
draft-lampin-voici-00

Abstract

The Static Context Header Compression (SCHC) framework identified the need for a minimal transport encapsulation that provides Session multiplexing when extrinsic Discriminators are insufficient. This document specifies a Link Multiplexer (VOICI) that addresses those SCHC-driven requirements while remaining general enough to accommodate other compression mechanisms and uncompressed payloads. The encapsulation is designed for minimal overhead, reducing to 2 bytes in the common case, while supporting optional integrity protection and original EtherType/port recovery.

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 6 December 2026.

Table of Contents

1. Introduction

The SCHC framework [SCHC] provides header compression and optional fragmentation based on static contexts shared between Endpoints. In the common deployment -- a single Instance per Endpoint over a single link -- the mapping between the link and the Instance is trivial: all Datagrams on the link belong to that one Instance, and no multiplexing mechanism is needed.

However, two deployment scenarios require a mechanism to distinguish multiple Sessions over a shared link:

* An Endpoint hosts multiple Instances serving different Domains or tenants.

* Multiple Sessions share an Ethernet segment or IPv6 link.

These requirements were first identified by the SCHC architecture [SCHC-ARCH] for the case of SCHC-compressed Datagrams. But the need is broader than SCHC alone. Operator and industrial deployments often carry a mix of traffic types on the same constrained link: SCHC-compressed Datagrams from devices that use static Contexts; Datagrams from other mechanisms; and uncompressed management or diagnostic traffic that bypasses compression. In all of these cases, transport-level multiplexing, and optional integrity are desirable.

This document specifies a Link Multiplexer (VOICI) that satisfies the requirements identified for SCHC while remaining general enough for other compression mechanisms. The VOICI header carries a Session ID for multiplexing, a Content Identifier for dispatching the Datagram to the correct handler, and optional integrity protection. The encapsulation is designed for minimal overhead, reducing to 2 bytes in the common case (1-byte flag + 1-byte Session ID for values less than 128).

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 BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Requirements

The requirements below are organized into two groups. Requirements 1-3 were first identified by the SCHC architecture [SCHC-ARCH] for the specific case of SCHC-compressed Datagrams. Requirements 4-5 were added when the scope was broadened to encompass other compression mechanisms and uncompressed payloads.

2.1. Requirements driven by SCHC

1. *Session identification*: A mechanism to distinguish Sessions and route Datagrams to the correct processing handler (for example, a SCHC Instance). The identifier (Session ID) is locally significant to the link.

2. *Original EtherType/port recovery (optional)*: A mechanism to carry the original EtherType or UDP port number when the carrier uses the SCHC EtherType or SCHC UDP port. This is needed when the payload is decompressed so that the receiver can restore the original framing layer after decompression.

3. *Integrity protection (optional)*: A mechanism to detect corruption of the Datagram, including the Session ID and the compressed residue.

2.2. Requirements driven by multi-mechanism and uncompressed payloads

1. *Content identification*: A mechanism to identify how the Datagram payload is encoded when the link carries Datagrams from multiple mechanisms (for example, SCHC, uncompressed). This allows the receiver to dispatch the Datagram to the correct decompressor without inspecting its contents.

2. *Layer independence*: The encapsulation MUST operate over any link layer that carries compressed traffic, whether identified by an Ethertype, IP Protocol Number, or UDP port [SCHC-PROTO-NUMS].

3. Gap Analysis

Several existing mechanisms can provide multiplexing or labeling. This section analyzes their suitability for SCHC and identifies the gap that VOICI fills.

3.1. MPLS

MPLS labels provide efficient multiplexing and are widely deployed in operator networks. However:

* MPLS adds 4 bytes per label, which may be excessive for highly constrained deployments.

* MPLS is not available on all link types relevant to SCHC (LPWAN, PPP, low-speed serial links).

* MPLS provides no integrity protection.

3.2. UDP Encapsulation

UDP is commonly used for Internet traversal and NAT traversal. The UDP source port can carry a Session ID:

* The UDP header is 8 bytes.

* Using the UDP source port as Session ID is fragile in the presence of NAT (port remapping) and port exhaustion (65535 limit shared with other applications).

* UDP is only available above IP.

3.3. IP Protocol Number and SCHC Ethertype

The SCHC IP Protocol Number and Ethertype [SCHC-PROTO-NUMS] identify SCHC traffic at the respective layers but do not provide:

* Session multiplexing (one protocol number or Ethertype per link, not per Session).

* Integrity protection.

They are necessary to identify SCHC traffic but insufficient for multiplexing.

3.4. Summary

Mechanism	Multiplexing	Integrity	Overhead	Link Coverage
MPLS	Yes	No	4+ bytes	Limited
UDP (src port)	Yes	No	8 bytes	IP only
IP Protocol Num	No	No	0 bytes	IP only

Ethertype	No	No	0 bytes	IEEE 802 only
VOICI	*Yes*	*Opt.*	*2 B*	*Any*

Table 1: Comparison of multiplexing mechanisms

VOICI fills the gap by providing multiplexing, integrity, content mechanism identification, and original EtherType/port recovery with minimal overhead. The comparison is summarized in Table 1.

3.5. Encoding Within SCHC Datagram

Encoding session or version information inside the SCHC rules or rule results would couple transport-layer concerns (multiplexing, version negotiation) to compression-layer concerns (what to compress, how to parse the residue). A separate encapsulation keeps the SCHC datagram focused on compression results and allows the transport header to be added or removed without modifying the compression strategy or the Context/Rules.

Furthermore, when multiple compression mechanisms share the same link, an inner-field approach would require every mechanism to reserve space for the same routing metadata, reducing compression efficiency. VOICI places this metadata in a single, mechanism-agnostic header.

4. Integration within SCHC framework

VOICI integrates at the carrier layer using the SCHC Ethertype and IP/UDP protocol numbers defined in [SCHC-PROTO-NUMS]. When multiplexing is required, these values identify VOICI traffic on the wire. On deployments where explicit multiplexing is not needed, i.e., provided by the supporting lower layers, VOICI is optional. The use of VOICI is part of the Endpoint configuration.

On the sender side, the VOICI module prepends its header to the Datagram and replaces the original EtherType, IP Protocol Number, or UDP port number with the corresponding SCHC Ethertype or IP/UDP protocol number. If the original framing information must be preserved for later restoration, the Original EtherType/Port flag (O) is set and the field is populated.

On the receiver side, packets identified by the SCHC Ethertype or IP/UDP protocol number are handed to the VOICI dispatcher. The VOICI module parses the header, uses the Session ID and CI field to route the Datagram to the correct processing handler, strips its own

header, and optionally restores the original EtherType, IP Protocol Number, or UDP port number before passing the reconstituted frame to upper layers.

5. Header Format

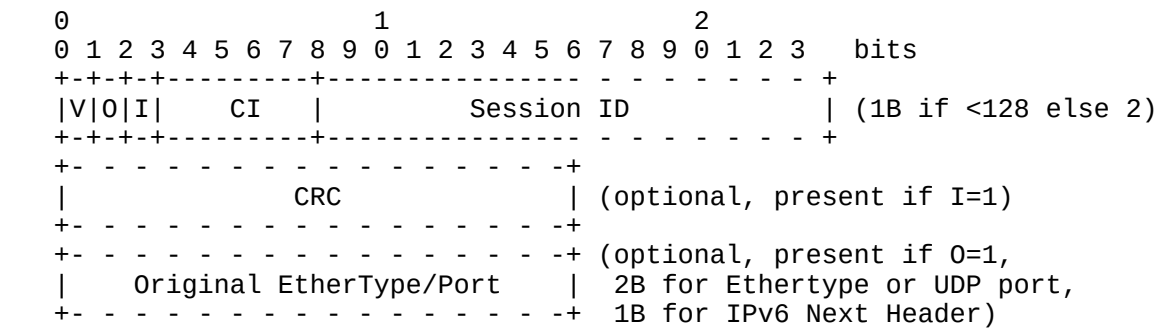

Figure 1: VOICI Header

The V-O-I flags (3 bits), CI field (5 bits), and the Session ID (1-2 bytes) are always present. The CRC (2 bytes) is present when I=1. The Original EtherType/Port (1-2 bytes) is present when O=1.

The Datagram payload follows immediately after the last header field.

Parsing order:

1. Read byte 0; extract V, O, I, CI.

2. Read Session ID (LEB128, 1-2 bytes).

3. If I=1, read 2-byte CRC and compute expected CRC over the flag byte, Session ID, Original EtherType/Port (if O=1), and the Datagram payload; drop frame if CRC is invalid.

4. If O=1, read Original EtherType/Port field (2 bytes for Ethernet/UDP, 1 byte for IPv6 Next Header).

5. Pass remaining buffer to the identified handler and recover original/content.

6. If O=1, restore original Ethertype or Port number and return processed frame to original handler.

5.1. Fields

* *V (1 bit):* VOICI header format version. V=0 for this draft. V=1 for future VOICI revisions.

* *O (1 bit):* Original EtherType/Port present. When set, the Original EtherType/Port field is present, carrying the EtherType, IP Next Header, or UDP port number that was replaced by the VOICI EtherType, VOICI IP Protocol Number, or VOICI UDP port. The field is interpreted as an EtherType when VOICI is carried over a link-layer transport (for example, IEEE 802 Ethertype), as a Next Header if carried over IP, and as a UDP port when VOICI is carried in a UDP payload. This restoration is an VOICI responsibility; the Content Mechanism does not need to manage framing recovery and dispatching to original handler.

* *I (1 bit):* Integrity flag. When set, a CRC-16 field is present and covers the Session ID through the end of the datagram. When clear, no integrity check is carried.

* *CI (5 bits):* Content Identifier. Identifies the mechanism used for the datagram payload. The mechanism profile defines the interpretation of each CI value. VOICI profiles register new CI values as needed.

  The initial CI assignments are:

CI	Content Mechanism
0	Unprocessed / raw -- Datagram requiring no reconstruction; used for minimalistic multiplexing only
1	SCHC -- standard SCHC compressed residue
2-31	Reserved for future mechanisms

Table 2: Initial CI assignments

Profiles that register a new CI value MUST specify the mechanism and its parameters.

* *Session ID (variable length, LEB128):* Identifies the logical session that owns this Datagram. When a mechanism is registered with VOICI, the mechanism profile assigns Session IDs and registers them with the VOICI instance. The receiver VOICI uses the Session ID to dispatch the Datagram to the correct handler --

for SCHC (CI=1), the handler is an SCHC Instance; for other mechanisms, the handler is defined by the mechanism profile. The Session ID space (0-65535) is local to the link over which VOICI is carried and to the Content Mechanism.

Values up to 127 fit in 1 byte; larger values use 2 bytes. The Session ID is encoded as a LEB128 variable-length integer [DWARF]:

- If the value is less than 128, a single byte is used (MSB = 0).
- If the value is 128 or greater, two bytes are used (first byte MSB = 1).
- No values larger than 16 bits (65535) are supported.

The receiver reads the Session ID by inspecting the most significant bit of each byte: if the MSB is 1, the next byte is part of the value; if 0, the byte is the last.

* *Original EtherType/Port (1-2 bytes, optional):* Present when O=1. Carries the EtherType or UDP port number that was replaced by the VOICI carrier. The field is interpreted as an EtherType when VOICI is carried over a link-layer transport (for example, IEEE 802 Ethertype) and as a UDP port when VOICI is carried in a UDP payload.

* *CRC (16 bits, optional):* Present when I=1. CRC-16/CCITT-FALSE over the flag byte (V-O-I-CI), the Session ID, the Original EtherType/Port field (if O=1), and the entire Datagram payload.

5.2. Minimal Header

When no optional fields are needed (V=0, O=0, I=0), the VOICI header reduces to 2-3 bytes (flag byte + 1-2 byte Session ID):

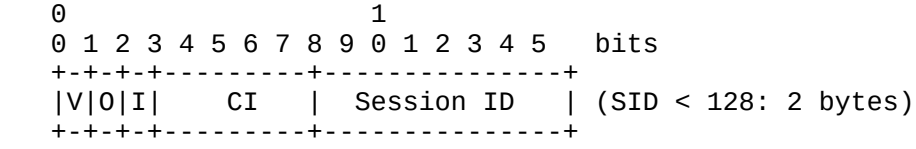

Figure 2: Minimal VOICI Header (2-3 bytes)

5.3. Header Size Summary

VOICI header sizes for various configurations (SID < 128 vs SID >= 128):

Configuration	V	O	I	SID < 128	SID >= 128
Session ID only	0	0	0	2 B	3 B
+ CRC	0	0	1	4 B	5 B
+ Orig. EtherType/Port	0	1	0	4 B	5 B
All fields	0	1	1	6 B	7 B

Table 3: VOICI header size summary

5.4. Header Field Reference

Field	Size	Description
V	1 bit	VOICI header version
O	1 bit	Original EtherType/Port presence
I	1 bit	CRC presence
CI	5 bits	Content Identifier
Session ID	1-2 B	Session identifier (LEB128)
Original ET/Port	1-2 B	EtherType, Next Header, or UDP port (if O=1)
CRC	2 B	Integrity check (if I=1)

Table 4: VOICI header field summary

6. Session ID Allocation

The Session ID is locally significant to the link. Allocation strategies depend on the deployment topology:

6.1. P2P Deployments

Session IDs MAY be negotiated between peers during Session establishment, or assigned by the Domain Manager during provisioning. Session ID 0 is a valid Session ID (no reserved values).

6.2. Star Topologies

The Network Gateway assigns Session IDs and communicates them to Devices during provisioning. The Gateway maintains the Session ID to handler mapping.

6.3. Mesh and Other Topologies

Session IDs MAY be assigned by a Network or Domain Manager, or negotiated between peers.

6.4. Relay Remapping

A relay or gateway translating between links MAY remap Session IDs. The Session ID space is local to each link segment; there is no requirement for global uniqueness.

7. Content Mechanism Identification

The CI field provides content mechanism identification. VOICI at the receiver uses the CI and Session ID values to dispatch the Datagram to the correct handler without inspecting the Datagram contents.

This is needed when a link carries Datagrams from multiple mechanisms simultaneously. Common scenarios include:

* A gateway that receives both SCHC-compressed Datagrams and Management and diagnostic traffic that bypasses compression entirely.

* Future registrations of additional mechanisms via new CI values.

7.1. Registration of New Mechanisms

Profiles that register a new CI value MUST specify the mechanism and its parameters. Implementations that encounter a CI value they do not recognize MUST drop the Datagram.

8. Integrity Protection

The I flag and CRC field provide optional integrity protection for the Datagram.

8.1. CRC Scope

The CRC covers the VOICI header and the Datagram payload, excluding the CRC field itself. Specifically, the CRC is computed over the flag byte (V-O-I-CI), the Session ID, the Original EtherType/Port field (if O=1), and the entire Datagram payload.

8.2. CRC Algorithm

CRC-16/CCITT-FALSE (polynomial 0x1021, initial value 0xFFFF, no reflection, no final XOR) is used. This is the same algorithm used in many constrained network protocols (for example, Bluetooth, CAN bus).

8.3. Relationship to ULP Checksums

Some compression strategies elide Upper Layer Protocol (ULP) checksums (for example, UDP checksum) to reduce residue size. On links where the underlying transport does not guarantee datagram integrity, this makes the VOICI CRC the sole integrity mechanism. Profiles MUST specify whether ULP checksum elision is permitted and, if so, whether the VOICI CRC is mandatory to compensate.

8.4. Limitations

The CRC provides integrity (corruption detection) but NOT authentication. An attacker can compute a valid CRC for a forged Datagram. Authentication must be provided by the underlying transport or a higher-layer security mechanism.

9. Interaction with Protocol Numbers

The protocol numbers defined in [SCHC-PROTO-NUMS] identify VOICI traffic on the wire. The VOICI header follows the carrier header and provides Session multiplexing, Content Mechanism dispatch, and optional integrity protection.

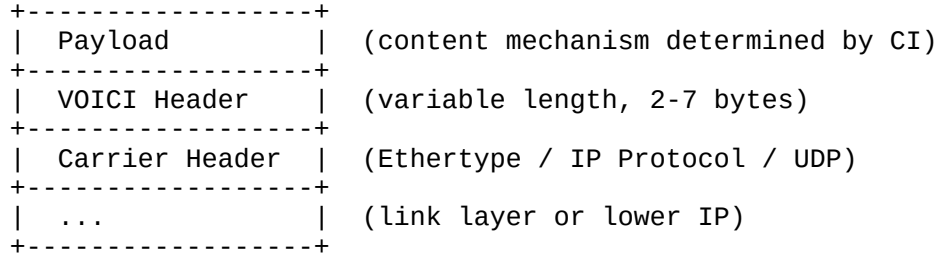

Figure 3: VOICI Layer Stack

The CI field identifies the mechanism (CI=0: unprocessed; CI=1: SCHC; other values: future registrations).

9.1. Over Ethertype

The SCHC Ethertype identifies VOICI-encapsulated traffic. When O=1, the Original EtherType/Port field carries the replaced EtherType value.

9.2. Over IP Protocol Number

The SCHC IP Protocol Number identifies VOICI-encapsulated traffic. The VOICI header follows the IPv6 header (or the IPv6 extension containing the protocol number). When O=1, the Original EtherType/Port field carries the replaced IPv6 Next Header value.

9.3. Over UDP

The SCHC UDP port identifies VOICI-encapsulated traffic carried in the UDP payload. The UDP header provides its own checksum, which may make the VOICI CRC redundant. When O=1, the Original EtherType/Port field carries the replaced UDP destination port number.

10. Security Considerations

10.1. Session Hijacking

If Session IDs are predictable, an attacker could inject Datagrams with a forged Session ID to redirect traffic to a different handler. Session IDs SHOULD be randomly generated or derived from a secure key exchange. In star topologies where the Domain Manager assigns Session IDs, the assigned values SHOULD be cryptographically random rather than sequential or otherwise predictable.

10.2. Integrity Limitations

The CRC provides corruption detection but not authentication. An attacker with link access can forge Datagrams with valid CRCs. Authentication must be provided by the underlying transport (for example, IPsec, TLS) or a higher-layer mechanism (for example, OSCORE).

10.3. Flag Bit Manipulation

When the I flag is set, the CRC covers the flag byte, making flag bit flipping detectable at the cost of a CRC failure. When I=0, flipping the O flag (0 to 1) would cause the receiver to consume payload bytes as an Original EtherType/Port field. Higher-layer authentication is recommended for adversarial environments.

10.4. CI Manipulation

When the I flag is set, the CRC covers the CI field, making manipulation detectable. When I=0, an attacker can flip the CI field to dispatch the Datagram to a wrong handler, causing decompression errors or potential information leakage if the wrong decompressor produces interpretable output.

10.5. Denial of Service

An attacker could inject Datagrams with invalid Session IDs, causing the receiver to waste resources on lookup failures. Implementations SHOULD rate-limit Session ID lookup failures.

10.6. Replay Attacks

VOICI carries no sequence number or timestamp. An attacker with link access could replay previously captured Datagrams. For SCHC's primary use cases (sensor telemetry, periodic reporting), replayed Datagrams carry stale data that is not harmful. Deployments requiring replay protection SHOULD use a higher-layer mechanism (for example, OSCORE, DTLS) or the underlying transport.

11. IANA Considerations

11.1. Content Identifier Registry

This document requests the creation of a "Content Identifier (CI)" registry. The initial entries are:

Value	Content Mechanism	Reference
0	Unprocessed / raw	This document
1	SCHC [SCHC]	[SCHC]
2-31	Reserved	--

Table 5: Initial CI registry entries

New CI values are assigned per [RFC8126] "Specification Required" policy.

11.2. Session ID Space

The Session ID space is locally significant to the link and Content Mechanism. No IANA assignment is required.

11.3. Future Extensions

The VOICI header allows for future extensions. New flags or fields would be introduced through a subsequent revision of this document, with IANA registry updates. Existing implementations that encounter unrecognized flag combinations MUST treat the unrecognized flags as zero and process the header according to their supported flags. For the CI field, implementations that encounter a CI value they do not recognize MUST drop the Datagram.

12. References

12.1. Normative References

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

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

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

[SCHC] 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>.

12.2. Informative References

[DWARF] Dwarf Standards Committee, "DWARF Debugging Information Format", Web https://dwarfstd.org/documentation/, <https://dwarfstd.org/documentation/>.

[SCHC-ARCH] Pelov, A., Thubert, P., and A. Minaburo, "Static Context Header Compression (SCHC) Architecture", Work in Progress, Internet-Draft, draft-ietf-schc-architecture-05, 17 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-schc-architecture-05>.

[SCHC-PROTO-NUMS] Moskowitz, R., Thubert, P., Gomez, C., Minaburo, A., and M. Blanchet, "Protocol Numbers for SCHC", Work in Progress, Internet-Draft, draft-ietf-schc-protocol-numbers-06, 23 December 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-schc-protocol-numbers-06>.

Author’s Address

Quentin Lampin
Orange
Orange 3 Massifs - 22 Chemin du Vieux Chene
38240 Meylan
France
Email: quentin.lampin@orange.com